Adam Pieprzycki

Wybrane algorytmy optymalizacji inspirowane zjawiskami przyrodniczymi

z zastosowaniami w telekomunikacji bezprzewodowej oraz przykładami w środowisku Matlab

Selected nature-inspired optimisation algorithms (NIOA)

with applications in wireless telecommunications and examples in the Matlab environment

Wydawnictwa Akademii Tarnowskiej

Wybrane algorytmy optymalizacji inspirowane zjawiskami przyrodniczymi

z zastosowaniami w telekomunikacji bezprzewodowej oraz przykładami w środowisku Matlab

Selected nature-inspired optimisation algorithms (NIOA)

with applications in wireless telecommunications and examples in the Matlab environment

Adam Pieprzycki

Wybrane algorytmy optymalizacji inspirowane zjawiskami przyrodniczymi

z zastosowaniami w telekomunikacji bezprzewodowej oraz przykładami w środowisku Matlab

Selected nature-inspired optimisation algorithms (NIOA)

with applications in wireless telecommunications and examples in the Matlab environment

Wydawnictwa Akademii Tarnowskiej
Tarnów 2025

Publikacja ukazała się nakładem Akademii Tarnowskiej

Recenzent
Witold Byrski

Grafika na okładce: yorpiz.com, grafika połączona ze zdjęciem [429]

Wydawca
Wydawnictwa Akademii Tarnowskiej
u. Mickiewicza 8, 33-100 Tarnów, www.atar.edu.pl
tel. +48 14 631 65 67m e-mail: wydawnictwa@atar.edu.pl

ISBN 978-83-976333-9-1

Projekt okładki i skład
Teresa Stańczyk

Podziękowania

Szczególne podziękowania chciałbym złożyć żonie Małgorzacie, córce Ricie oraz rodzicom. Podziękowania chciałbym także wyrazić dla dra hab. inż. Wiesława Ludwina prof. AT, prof. dra hab. inż. Bogusława Filipowicza za cenne uwagi dotyczące tematyki telekomunikacji oraz optymalizacji, dr inż. Edycie Gawin i dr. Tomaszowi Beberokowi za matematyczną korektę pracy.

Spis treści

Streszczenie

W pierwszej części niniejszej monografii rozważono cały szereg nowoczesnych algorytmów optymalizacji, które zaimplementowano i następnie oceniono w różnych zadaniach optymalizacji. Szczególną uwagę zwrócono na algorytmy inspirowane przyrodą, a zwłaszcza najnowsze algorytmy rojowe, które stanowią efektywne narzędzie w optymalizacji globalnej różnych zadań optymalizacji.

W drugiej części monografii, za pomocą odpowiednich narzędzi matematycznych oraz wybranych metod badań operacyjnych (*operations research*) podjęto próbę wyznaczenia rozwiązania optymalnego postawionego w monografii problemu.

W planowaniu sieci WLAN (*Wireless Local Area Network*) z użyciem funkcji kryterialnej, optymalna liczba punktów dostępu AP, ich wzajemne położenie, a także wybór parametrów pracy tych punktów są wynikiem rozwiązania zadania optymalizacji z użyciem jednej lub kilku funkcji kryterialnych.

Pierwszym krokiem w sformułowaniu zadania optymalizacji, umożliwiającego w zadanym środowisku radiokomunikacyjnym optymalne zaplanowanie sieci WLAN standardu IEEE 802.11, w tym w szczególności w środowisku wewnątrzbudynkowym, był wybór odpowiedniej funkcji kryterialnej.

Obok wyznaczenia zasięgu radiowego, a zatem i obszaru działania sieci WLAN, innym bardzo ważnym zagadnieniem, koniecznym do znalezienia właściwego rozwiązania zadania optymalizacji opartego na zadanej funkcji kryterialnej, było oszacowanie wartości przepustowości analizowanej sieci WLAN.

W monografii modelowano i analizowano sieci WLAN standardu IEEE 802.11 pracujące w paśmie ISM 2.4 GHz. Dla sieci pracujących w tym paśmie przedstawiono szereg wzorów i zależności umożliwiających obliczanie maksymalnej przepustowości sieci WLAN zarówno przy wykorzystaniu modeli prostych, takich jak TUL (*Throughput Upper Limit*), jak również przy zastosowaniu modeli bardziej rozbudowanych i dokładnych, w tym

w szczególności opartych na łańcuchach Markowa i uwzględniających, na przykład statyczny wybór trybu pracy (*multi-rate*) stacji ST.

W celu ograniczenia wpływu technik sterowania przepływem i retransmisją pakietów na przepustowość sieci, w monografii rozważano tylko i wyłącznie transmisję danych z protokołem UDP. Tym samym otrzymane przy wykorzystaniu tego protokołu wartości przepustowości sieci WLAN były wartościami największymi z możliwych do osiągnięcia w tego typu sieciach i określały kres górny przepustowości tych sieci WLAN w zadanym środowisku wewnątrzbudynkowym.

Obecność stacji ST o znacząco małych szybkościach transmisji M_{TR} prowadzi coraz częściej we współczesnych instalacjach sieci WLAN z infrastrukturą do istotnego pogorszenia ich przepustowości. Stacje ST pracujące z małymi szybkościami transmisji, czego przyczyną jest przede wszystkim stosowanie przez te stacje starszych wersji standardu IEEE 802.11, skutecznie ograniczają możliwości transmisyjne wszystkich pozostałych stacji ST, które mogłyby nadawać z dużo większymi szybkościami. W kontekście tego zjawiska, nazywanego powszechnie anomalią wydajności (*Performance Anomaly*) sieci Wi-Fi, coraz większego znaczenia nabiera staranne planowanie sieci WLAN pod kątem dostępnego na rynku sprzętu Wi-Fi (*Wireless Fidelity*) oraz wyboru trybu pracy stacji ST. I właśnie taki rodzaj planowania sieci WLAN z infrastrukturą pracujących w paśmie ISM 2.4 GHz jest możliwy do przeprowadzenia za pomocą funkcji kryterialnych zaprezentowanych w rozdziale drugim tej monografii.

Szacowanie przepustowości sieci WLAN, obok szacowania jej zasięgu jest istotnym elementem zadania optymalizacji, gdyż daje możliwość bardziej pełnego określenia wydajności sieci bezprzewodowej.

Ponieważ rozwiązanie zadania optymalizacji, w przypadku zastosowania nieliniowych funkcji kryterialnych może nastręczać wiele problemów i to nie tylko natury obliczeniowej, dlatego w niniejszej monografii rozważono cały szereg nowoczesnych algorytmów optymalizacji, które zaimplementowano i następnie oceniono w różnych zadaniach optymalizacji.

W niniejszej monografii dokonano analizy komparatywnej rozwiązań uzyskanych z użyciem wybranych algorytmów optymalizacji w zadaniach opartych na funkcjach kryterialnych zdefiniowanych w rozdziale drugim.

Wykorzystane w publikacji wybrane modele propagacyjne, a także model matematyczny przepustowości, system operacyjny czy rodzaj transmisji danych miały przede wszystkim na celu opracowanie i budowę takiego

modelu matematycznego sieci WLAN IEEE 802.11 z infrastrukturą, który byłby jak najbliższy rzeczywistości.

W monografii do znajdywania najlepszego rozwiązania zadania optymalizacji i planowania sieci WLAN z infrastrukturą zastosowano zarówno optymalizację jednokryterialną SOO (*Single Objective Optimisation*), jak i wielokryterialną MOO (*Multi Objective Optimisation*), która daje możliwość uwzględnienia w zadaniu optymalizacji wielu funkcji kryterialnych oraz otrzymania zbioru rozwiązań niezdominowanych. Trudnością w zastosowaniu optymalizacji wielokryterialnej MOO jest brak znajomości frontu Pareto (*P*) oraz duża liczba rozwiązań Pareto-optymalnych, które trzeba poddać bardzo szczegółowej analizie zanim wybierze się jedno z nich.

Dwa analizowane zestawy funkcji MOO porównano z wynikami uzyskanymi za pomocą optymalizacji jednokryterialnych SOO opartych na różnych wybranych funkcjach celu. Z otrzymanych wyników utworzono front Pareto (*P*) i po zastosowaniu metody MUZ (Metoda Unitaryzacji Zerowanej) najlepsze wyniki poddano analizie wskaźnikiem efektywności *PM*. Dodatkowo otrzymane rozwiązania zweryfikowano empirycznie, a także budując na podstawie $\overline{S^E}$, $\overline{PM^E}$ i $\overline{JFI^E}$ ranking GQ_i^{best} (rozdział 4.3).

Przeprowadzono szereg analiz porównawczych wybranych – znanych z literatury – najważniejszych algorytmów optymalizacji OPA (*Optimisation Algorithm*), umożliwiających nie tylko wybór liczby punktów dostępu AP sieci WLAN z infrastrukturą oraz ich optymalne rozmieszczenie we wnętrzu budynku objętego zasięgiem radiowym sieci, ale także pozwalających wyznaczyć moce wyjściowe nadajników i częstotliwości kanałów radiowych w poszczególnych jej punktach dostępu AP. Dla tak sformułowanego zbioru zasad i metod optymalnego, z punktu widzenia przyjętej funkcji kryterialnej, planowania sieci WLAN z infrastrukturą oraz dla wybranych algorytmów optymalizacji OPA zaprojektowano, w konkretnym środowisku wewnątrzbudynkowym przykładową lokalną bezprzewodową sieć komputerową standardu IEEE 802.11 z infrastrukturą, którą następnie poddano szczegółowej analizie.

Abstract

The primary aim of this monography was compiling a set of principles and ways to plan local networks of IEEE 802.11 standard with infrastructure based upon DSSS and OFDM techniques. A procedure devised – for the purposes of designing, construction, and use – of planning wireless computer networks was, first of all, to guarantee, considering the accepted Optimisation criteria, a reliable, and efficient operation of such networks in indoor environment.

In WLAN network planning assisted with objective function the optimal number of access points (AP), their placement in relation to one another, and selection of operation parameters comprise a result of solving a problem using one or many objective functions.

In Chapters 3 and 4 of this monography applications of appropriate mathematical instruments and selected operations research methods has enabled an attempt to obtain an optimal solution to the problem posited in this book.

The first step in formulation of Optimisation problem, enabling the optimal planning of IEEE 802.11 WLAN network in a given radio–communications environment (indoor setting in particular), was selection on appropriate objective function (Chapter 2).

In the next step of local network planning a propagation model had to be selected, one which would reflect in the best possible manner transmission characteristics of wireless links both in ST (wireless station) to AP (access point) relationship, as well as AP to ST, in indoor environment in particular.

Apart from establishing the radio range, and thus WLAN operation area, the other vital issue, necessary for establishing the correct solution to Optimisation problem based upon a given objective function, was estimation of throughput values of the analysed WLAN (Chapter 2).

In the book IEEE 802.11 standard WLAN networks were modelled and analysed, operating within ISM 2.4 GHz radio bands. For those networks an array of patterns and relationships enabling calculation of maximal throughput were presented. These included both simple models (such as TUL – Throughput Upper Limit), as well as more complex and exact ones,

based upon Markov chains in particular, involving, for instance, static selection of operation modes (multi-rate) of ST.

So as to limit the impact of traffic and retransmission management techniques onto the network throughput in the dissertation only UDP protocol-based data transmission has been taken into consideration. Thus, the WLAN throughput values obtained were the highest possible for such types on networks.

Presence of ST with significantly low transmission rates M_{TR} leads to worsening of throughput of WLAN networks. Wireless stations (ST) operating in low transmission rates, which is primarily caused by usage the older versions of IEEE 802.11 standard, significantly limit transmission rates of all other wireless stations (ST) which would otherwise be able to transmit with greater rates. In the context of this phenomenon, known as Wi-Fi Performance Anomaly, careful planning of WLAN network acquires even greater importance – available Wi-Fi hardware, and ST operation modes must be taken into consideration. Such planning of WLAN networks with infrastructure operating in 2.4 GHz is possible using objective functions, as presented in Chapter 2 of this book.

Estimation on WLAN throughput next to evaluation of its range is an important part of Optimisation problem, as it offers a possibility of a fuller assessment of wireless network's efficiency.

As solving the Optimisation problem using non-linear objective functions may present many problems – not only computational in nature – in this dissertation a selection of modern Optimisation algorithms were taken into consideration, which were implemented, and subsequently evaluated in various Optimisation problems. A particular portion of attention was given to nature-inspired algorithms, swarm algorithms in particular, which constitute an effective tool for global Optimisation of various Optimisation problems.

In Chapters 3 and 4 of this book a comparative analysis of results obtained with selected Optimisation algorithms in objective function-based problems (defined in Chapter 2) was achieved.

The selected propagation models applied in this dissertation, as well as mathematical throughput model, packets generator, operation system or type of data transmission were intended to facilitate devising and construction of mathematical model of WLAN IEEE 802.11 network with infrastructure which would approximate reality to the highest degree.

In the book for the purpose of finding the best solution of Optimisation problem and WLAN network planning along with its infrastructure multi-objective Optimisation (MOO) was used, which allows for including many objective functions in the Optimisation problem, as well as obtaining a set of non-dominated solutions. The difficulty in using MOO is lack of known Pareto frontier (P) as well as large number of Pareto-optimal solutions which have to undergo thorough analysis before one of them can be selected.

Two MOO functions sets which underwent analysis were compared with results obtained via single objective Optimisation (SOO) based upon various selected objective functions. The obtained results allowed for creation of Pareto frontier, and, after application of Zero Unitarization Method (MUZ), the best results were further analysed with performance metric (PM). Additionally, the results obtained were verified empirically, and a ranking GQ_{i}^{best} was constructed upon $\overline{S^E}$, $\overline{PM^E}$, $\overline{JFI^E}$ estimated number of AP and assumed number of ST stations. (Chapter 4.3).

In the book a number of comparative analyses of selected, known from scientific publications, and the most important Optimisation algorithms (OPA) was conducted. The algorithms allow for choice of number of AP in WLAN with infrastructure, and their optimal placement within indoor area covered by radio reach of the network, but also allow establishing of initial power of transmitter, and radio frequencies in its particular access points. Using thus formulated set of rules and methods of optimal (from the objective function perspective) planning of WLAN network with infrastructure and using selected Optimisation algorithms a test local wireless network with infrastructure, compliant with IEEE 802.11 standard was implemented in indoor environment.

Wykaz oznaczeń i symboli

a_U	pomocniczy parametr w określeniu $U\left(P_S^{thr}\right)$,
A_e	powierzchnia skuteczna anteny,
B	*Bandwidth* – szerokość kanału radiowego,
BER	*Bit Error Rate* – bitowa stopa błędów,
b_U	pomocniczy parametr w określeniu $U\left(P_S^{thr}\right)$,
$b(x,y)$	stany systemu w modelu (z użyciem łańcuchów) Markowa,
$b_{-1,0}$	stan braku aktywności (*idle state*) stacji abonenckiej ST,
$b_{-2,0}$	stan nasycenia,
B	szerokość kanału radiowego,
c	prędkość światła,
c_{1no}	najmniejsza wartość nominanty metody MUZ,
c_{2no}	największa wartość nominanty metody MUZ,
ch_{nr}	numer kanału radiowego WLAN standardu IEEE 802.11 dla pasma ISM 2.4 GHz,
C	współczynnik (pojemnościowy) sprawności punktu dostępu AP,
CFI	*Capacity Fairness Index* pojemnościowy indeks sprawiedliwego przydziału zasobów Jaina,
CFI^E	empiryczna wartość pojemnościowego indeksu sprawiedliwego przydziału zasobów Jaina,
CW_{min}	minimalny rozmiar okna rywalizacji,
CW_{max}	maksymalny rozmiar okna rywalizacji,
d	odległość między antenami nadajnika i odbiornika,
d, φ, θ	współrzędne punktu w układzie sferycznym,
d_i	długość drogi dla promienia (i),
d_{ji}	odległość między punktem dostępu AP (j) a stacją ST (i) połączoną z tym punktem,
d_{jk}	odległość między najdalszą stacją ST należącą do klasy ruchu (k) a punktem dostępu AP (j),
d_0	odległość odniesienia dla modeli propagacyjnych,
D_r	współczynnik dyfrakcji promienia,
D_t	liczba kierunków transmisji,

$\overrightarrow{E_i}(\theta, \varphi, d_i)$	wektor natężenia pola elektrycznego dla promienia *i,*
E_0	natężenie pola elektrycznego odniesienia,
$E[L_{DATA\_payload}]$	średni rozmiar pola informacyjnego (ładunku) pakietu,
$E[T]$	średni czas trwania wszystkich rozważanych stanów w kanale radiowym,
f_c	częstotliwość środkowa kanału radiowego w pasmie ISM 2.4 GHz,
F_c	funkcja kryterialna,
$\mathbb{F}\mathbb{c}^*$	współrzędne wektora określającego cel poszukiwań optymalizacji wielokryterialnej,
GQ_i	zmienna agregowana (syntetyczna) (*Global Quality*) metody MUZ,
GQ_i^{best}	najlepsze rozwiązania metody MUZ,
G_t	maksymalny zysk energetyczny anteny nadawczej,
G_r	maksymalny zysk energetyczny anteny odbiorczej,
I_t	moc szumów interferencyjnych,
JFI	*Jain's Fairness Index* indeks sprawiedliwego przydziału zasobów Jaina,
JFI^E	empiryczny indeks sprawiedliwego przydziału zasobów Jaina,
k_B	stała Boltzmanna,
K	liczba wszystkich rodzajów przeszkód na drodze propagacji sygnału,
l_r	straty sygnału w przewodzie i w złączach anteny odbiornika,
l_t	straty sygnału w przewodzie i w złączach anteny nadajnika,
L	tłumienie sygnału na w torze nadawania,
L_{epo}	rozmiar „epoki" – liczba powtórzeń uruchomienia algorytmu symulowanego wyżarzania SA dla tej samej wartości parametru – temperatura,
L_{f_j}	tłumienie sygnału radiowego wnoszone przez strop typu (j),
L_i^j	tłumienie sygnału między stacją ST (i), a punktem dostępu AP (j),
$L_{i,max}$	maksymalne dopuszczalne tłumienie sygnału między anteną stacji abonenckiej ST (i), a anteną punktu dostępu AP (j),
L_{o_k}	tłumienie przeszkody typu k,
L_r	limit ponowień transmisji ramek danych,

L_{ser}/L_{tail}	rozmiar pól SERVICE/TAIL warstwy PHY OFDM,
L_{sig}	tłumienie sygnału,
$L_w L_w$	tłumienie ściany,
L_{w_i}	tłumienie sygnału dla przeszkód (np. ścian) typu (i),
L_{ACK}	rozmiar ramki typu ACK,
L_{DbpS}	liczba zakodowanych bitów (*DATA*) na jeden symbol,
L_{FSL}	tłumienie sygnału w wolnej przestrzeni FSL,
L_{DATA}	rozmiar ramki danych (*DATA*),
$L_{DATA\_payload}$	rozmiar ładunku ramki danych MSDU,
L_{ITU}	tłumienie medianowe sygnału według modelu *ITU-R P.1238,*
L_{LAM}	medianowe tłumienie sygnału według modelu liniowego tłumienia *LAM*,
L_{MK}	medianowe tłumienie sygnału według modelu *Motleya-Keenana*,
L_{MWM}	medianowe tłumienie sygnału według modelu *MWM*,
$L_{MWMcost}$	medianowe tłumienie sygnału według modelu *MWM* dla modelu COST 231,
L_{TC}	liczba klas ruchu wynikająca z szybkości transmisji,
L_{UDP}/ L_{IP}	rozmiar nagłówka UDP/ IP,
L_0	tłumienie sygnału dla odległości odniesienia,
L_{1SM}	tłumienie medianowe sygnału według modelu *One Slope*,
L_{50}	tłumienie medianowe sygnału na drodze propagacji,
m	maksymalna liczba ponowień procedury *backoff DCF,*
M_{CR}	szybkość transmisji danych sterujących (*control rate*),
M_{TR}	szybkość transmisji danych,
M_{TR_i}	szybkość transmisji danych stacji abonenckiej ST (i),
M_{TR}^{max}	maksymalna szybkość transmisji danych,
M_{TR}^{min}	minimalna szybkość transmisji danych,
$M_{TR_{ij}}^{max}, M_{TR_{ij}}^{min}$	maksymalna i minimalna szybkość transmisji danych dla stacji abonenckiej ST (i) współpracującą z punktem dostępu AP (j),

n	wymiar przestrzeni optymalizacji,
n_f	liczba stropów na drodze propagacji sygnału,
n_{ji}	liczba stacji abonenckiej ST dla danej klasy ruchu (i) przydzielonych do punktu dostępu AP (j),
n_{prop}	współczynnik propagacji sygnału dla modelu *One-slope*,
n_{o_k}	liczba przeszkód typu (k) na drodze propagacji sygnału,
n_w	liczba ścian na drodze propagacji fali radiowej,
n_{w_i}	liczba ścian typu (i),
N	współczynnik tłumienia w modelu ITU-R P.1238,
N_c	liczba podnośnych informacyjnych,
N_{meas}	liczba pomiarów,
N_{power}	mocy szumów termicznych,
N_{psc}	liczba podnośnych pilota,
N_{ray}	liczba promieni składowych sygnału,
N_{sc}	liczba podnośnych,
N_{AP}	liczba punktów dostępu AP,
N_{DbpC}	liczba bitów danych na nośną,
N_{DbpS}	liczba bitów na symbol,
N_{ST}	liczba stacji abonenckich ST rywalizujących o dostęp do kanału radiowego,
$N_{ST}^{AP_n}$	liczba stacji ST przydzielonych poszczególnym punktom AP,
N_{ST}^{i}	liczba stacji abonenckich ST realizujących ruch klasy (i),
N_{ST_j}	liczba stacji abonenckich ST połączonych z punktem dostępu AP (j),
NF	współczynnik szumów odbiornika,
q	prawdopodobieństwo przebywania, co najmniej jednego pakietu w kolejce do nadania,
p	prawdopodobieństwo,
$p_{BER}^{M_{TR}}(L_{DATA})$	prawdopodobieństwo błędu w pakiecie danych,

$p_{BER}^{M_{CR}}(L_{ACK})$	prawdopodobieństwo wystąpienia błędu w pakiecie potwierdzenia odbioru,
p_{cap}	prawdopodobieństwo wystąpienia efektu przechwytywania sygnału (*capture effect*),
p_e	prawdopodobieństwo błędnej transmisji,
p_{idle}	prawdopodobieństwo stan: wolny kanał,
p_s	prawdopodobieństwo udanej transmisji,
p_{s_i}	prawdopodobieństwo udanej transmisji dla i-tej klasy ruchu L_{TC},
p_{str_i}	prawdopodobieństwo wystąpienia tylko jednej transmisji typu (i) w kanale radiowym,
p_{tr}	prawdopodobieństwo zajętego kanału radiowego – wystąpienie transmisji w rozważanej szczelinie czasowej,
P_t	poziom mocy wyjściowej nadajnika,
r	współczynnik korelacji liniowej Pearsona,
r_t	współczynnik redukcji temperatury (schładzania) w algorytmie symulowanego wyżarzania (SA),
P_S^{thr}	czułość progowa odbiornika,
PM	wartość wskaźnika *Performance Metric* obliczona na podstawie modelu matematycznego,
PM^E	empiryczna wartość wskaźnika *Performance Metric* obliczona na podstawie pomiaru przepustowości,
RSL	poziom mocy sygnału na wejściu odbiornika stacji abonenckiej ST,
R	współczynnik odbicia,
S	przepustowość,
S^E	przepustowość określona empirycznie,
S_{thr}	poziom progowej przepustowości,
S_{TUL}	górna granica przepustowości dla punktu AP,
$SINR$	*Signal to Interference and Noise Ratio* – stosunek poziomu mocy sygnału do poziomu mocy zakłóceń interferencyjnych i mocy szumów własnych odbiornika,

T	wartość testu kolejności par Wilcoxona,
T_{emp}	temperatura,
T_C	średni czas zajętości kanału dla stacji abonenckiej ST uczestniczących w kolizji,
T_D	opóźnienie pakietu,
T^*_{DATA}	średni czas transmisji pakietu danych,
T_{DIFS}	czas trwania odstępu DIFS,
$T_{PHY_{header}}$	czas trwania nagłówka PHY,
T_{PRE}	czas transmisji preambuły PHY,
T_{PHY}	czas transmisji nagłówka PHY,
T_{SIFS}	czas trwania odstępu SIFS,
T_{opt}	czas rozwiązania zadania optymalizacji,
T_{slot}	czas trwania pojedynczej szczeliny czasowej,
T_{s_i}	średni czas udanej transmisji pakietu strumienia ruchu (i),
T_{symbol}	czas transmisji jednego symbolu OFDM,
$U(P_S^{thr})$	prawdopodobieństwo wystąpienia poziomu mocy sygnału na rozważanym obszarze (*Area Location Probability)*,
w_i	waga (dla funkcji kryterialnej F_{c_2}),
W	parametr testu Shapiro-Wilka badającego normalność rozkładu,
x	wartość analizowanego parametru np. przepustowości,
$X_{1_{AP}}$	współrzędna X_1 wyróżnionego punktu dostępu AP,
$X_{2_{AP}}$	współrzędna X_2 (Y) wyróżnionego punktu dostępu AP,
$X_{3_{AP}}$	współrzędna X_3 (Z) wyróżnionego punktu dostępu AP,
$X_\sigma(0,\sigma)$	zmienna losowa o rozkładzie Gaussa o zerowej średniej i wariancji σ^2,
$\mathbb{Z}_0$	impedancja falowa próżni,
α	poziom istotności,
α_{LAM}	współczynnik tłumienia dla modelu LAM,
ε_r	względna przenikalność elektryczna,
Θ	kąt padania promienia,

$\vec{\xi}$	wersor promienia,
μ_p	współczynnik kary (dla funkcji kryterialnej F_{c_2}),
δ	czas propagacji fali EM między punktem dostępu AP a stacją abonencką ST,
τ_i	prawdopodobieństwo udanej transmisji ramki,
λ	intensywność generowania pakietów,
σ	odchylenie standardowe,
σ_m	konduktywność,
$\sigma_{\overline{N_{ST}}}$	odchylenie standardowe liczby stacji ST przydzielonych poszczególnym punktom AP,
$\sigma_{\bar{S}}$	estymator niepewności średniej,
η_{MRE}	średni błąd względny MRE *Mean Relative Error*,
ν	współczynnik aktywności użytkownika stacji abonenckiej ST,
γ	zmienna swobodna optymalizacji wielokryterialnej,
Ψ	waga sterująca typem funkcji kryterialnej: F_{c_2}, F_{c_5}, F_{c_6},
$\Lambda(\gamma)$	obszar rozwiązań dopuszczalnych optymalizacji wielokryterialnej.

Wykaz skrótów

ABC	*Artificial Bee Colony*
ACK	*Acknowledgment*
AP	*Access Point*
BeA	*Bees Algorithm*
BA	*Bat Algorithm*
BSS	*Basic Service Set*
CBR	*Constant Bit Rate*
CCK	*Complementary Code Keying*
COST	*European Cooperation in Science and Technology*
CR	*Code Rate*
CSMA/CA	*Carrier Sense Multiple Access/Collision Avoidance*,
CTS	*Clear To Send*
CW	*Contention Window*
DBPSK	*Differential Binary Phase Shift Keying*
DCF	*Distributed Coordination Function*
DIFS	*DCF Inter Frame Space*
D–ITG	*Distributed Internet Traffic Generator*
DQPSK	*Differential Quadrature Phase Shift Keying*
DSSS	*Direct Sequence Spread Spectrum*
DUT	*Device Under Test*
EIFS	*Extended Inter Frame Space*
EIRP	*Effective/Equivalent Isotropic Radiated Power*
EM	*Electromagnetic*
ERP–OFDM	*Extended rate PHY*
ESS	*Extended Service Set*
FA	*Firefly Algorithm*
FAF	*Floor Attenuation Factor*
FER	*Frame Error Rate*
FM	*Fading Margin*
FSL	*Free Space Loss*
GA	*Genetic Algorithm*
GDOP	*Geometric Dilution of Precision*
HDOP	*Horizontal Dilution of Precision*
IBSS	*Independent Basic Service Set*
ISM	*Industry, Science, Medicine*
ITU	*International Telecommunication Union*

LOS	*Line Of Sight*
LWAPP	*Lightweight Access Point Protocol*
MAC	*Media Access Control*
ME	*Mean Error*
MWF	*Multi Wall Floor*
LAM	*Linear Attenuation Model*
MOO	*Multi Objective Optimisation*
MRE	*Mean Relative Error*
MSDU	*MAC layer Service Data Unit*
MTU	*Maximum Transmission Unit*
MWM	*Multi Wall Model*
NAV	*Network Allocation Vector*
NF	*Noise Figure*
OFDM	*Orthogonal Frequency Division Multiplexing*
OPA	*Optimisation Algorithms*
PER	*Packet Error Rate*
PHY	*Physical layer*
PM	*Performance Metric*
PPDU	*PHY Protocol Data Unit*
PSO	*Particle Swarm Optimisation*
QAM	*Quadrature Amplitude Modulation*
RBW	*Resolution Bandwidth*
RBO	*Random Back Off*
RF	*Radio Frequency*
RTS	*Request To Send*
RSL	*Received Signal Level*
RSSI	*Received Signal Strength Indicator*
RBW	*Resolution Bandwidth*
SA	*Simulated Annealing*
SIFS	*Short Inter Frame Space*
SOO	*Single Objective Optimisation*
ST	*Station*
TDOA	*Time-Difference-Of-Arrival*
TG	*Traffic Generator*
TUL	*Throughput Upper Limit*
UDP	*User Datagram Protoco*
UCNLNav	*Underwater Communication and Navigation Labolatory*
VBW	*Video Bandwidth*
WAF	*Wall Attenuation Factor*
WLAN	*Wireless Local Area Network*

1. WYBRANE ZAGADNIENIA OPTYMALIZACJI

Optymalizacja jest procesem, w którym – w ramach zbioru rozwiązań dopuszczalnych – poszukuje się w sposób uporządkowany takiego rozwiązania, dla którego funkcja kryterialna przyjmuje wartość najlepszą z możliwych, czyli optymalną (łac. *optimus* – najlepszy) [1], [2].

Początków zadań optymalizacji możemy odnaleźć u Wergiliusza (70–19 p.n.e.), który w poemacie *Eneida*, w której sprytna królowa Dydona poleciła pociąć skórę byka na wąskie rzemienie i opasać nimi obrys murów, czyli wyznaczyć przestrzeń wymaganą do założenia Kartaginy: „kupiwszy ziemi tyle, ile się zamyka byrsą" Wergiliusz, *Eneida* I, 367–368.

Rozwiązywanie zadania optymalizacji może nie być zagadnieniem łatwym, bowiem zazwyczaj przeszukiwany zbiór dopuszczalnych rozwiązań może być zbiorem bardzo dużym. Przeszukanie takiego zbioru, w celu odnalezienia rozwiązania optymalnego, może okazać się zadaniem niewykonalnym w z góry zadanym i akceptowalnym dla szukającego czasie. Ponieważ w analizie większości problemów technicznych trudno jest przewidzieć rozwiązanie znajdujące się w pobliżu rozwiązania optymalnego (suboptymalne), dlatego zastosowanie znalazły tu głównie metody optymalizacji globalnej, w tym w szczególności metody oparte na algorytmach heurystycznych (gr. *heuriskein* – znaleźć, odkryć) oraz rozwiązań hybrydowych, łączących w sobie różne techniki optymalizacji – także metody deterministyczne.

Rozwiązanie optymalne jest takim rozwiązaniem dopuszczalnym, które spełnia w sposób najlepszy z możliwych zadaną relację lub relacje wynikające z jednego lub kilku, tzw. wskaźników efektywności [3]. W literaturze, dotyczącej zagadnień optymalizacji, przez relację rozumie się zarówno funkcję celu, wskaźnik jakości, funkcjonał jakości, kryterium optymalizacji, kryterium jakości, jak i funkcję kryterialną (F_c).

W procesie optymalizacji poszukiwane są należące do zbioru rozwiązań dopuszczalnych tylko takie, dla których funkcja kryterialna przyjmuje wartość optymalną, tzn. minimalną lub maksymalną [2], [4]. Z problemem

optymalizacyjnym możemy mieć także do czynienia w sytuacji, gdy występuje rozbieżność między istniejącym a założonym (pożądanym) stanem układu/systemu. W takim przypadku, rozwiązanie zadania optymalizacji polega na takim przydziale dostępnych w układzie/systemie zasobów, który zminimalizuje ową rozbieżność, czyli obserwowaną różnicę.

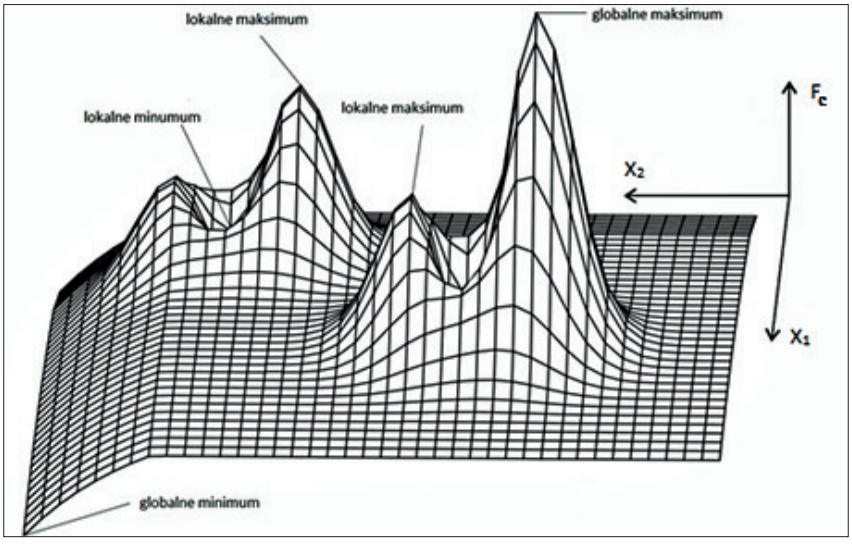
W sieciach WLAN takimi zasobami, podlegającymi optymalnemu zagospodarowaniu mogą być, w szczególności, liczba i rozmieszczenie oraz parametry konfiguracyjne punktów dostępu AP (np. *EIRP*, numer kanału radiowego ISM ch_{nr}, parametry metody DCF), jak również i inne parametry sieciowe (np. rozmiar pakietu czy jednostki *MTU*).

1.1. Ogólna charakterystyka metod i algorytmów optymalizacji

Wybór metody, za pomocą której poszukuje się rozwiązania konkretnego zadania optymalizacji [2] zależy od:

- rodzaju tego zadania – np. metoda programowania liniowego lub metody optymalizacji nieliniowej;
- występowania lub braku ograniczeń – np. metody z ograniczeniami lub bez ograniczeń;
- wymiaru problemu – np. metody jednowymiarowe lub wielowymiarowe;
- liczby kryteriów optymalizacji – np. metody jednokryterialne lub wielokryterialne.

Oceniając przydatność oraz wartość takiej czy innej metody, opartej na konkretnych algorytmach optymalizacyjnych, należy zwrócić uwagę także na efektywność procesu poszukiwania za jej pomocą rozwiązania zadania optymalizacji [4].



Rysunek 1.1. Lokalne i globalne optima dla dwuwymiarowej funkcji kryterialnej [5]

Wprawdzie algorytmy deterministyczne bardzo często znajdują optima lokalne, a nie globalne [6], to jednak, uruchamiając je z różnych punktów startowych, można je z powodzeniem wykorzystać do odnajdywania optimum globalnego zadanej funkcji kryterialnej F_c [7] (rys. 1.1).

Dokładność uzyskanego, przy użyciu wybranej metody optymalizacji, rozwiązania χ [8] określa, tzw. skuteczność konkretnego algorytmu optymalizacyjnego i ocenia się ją, np. za pomocą:

- odległości euklidesowej znalezionego rozwiązania χ od poszukiwanego optimum χ_{opt}:

$$\|x_{opt} - x\|_2, \tag{1.1}$$

- różnicy między wartościami funkcji kryterialnej wyznaczonymi dla rozwiązania χ i poszukiwanego optimum χ_{opt}:

$$\|F_c(x_{opt}) - F_c(x)\|_2. \tag{1.2}$$

Stosowane są także inne kryteria oceny dokładności wyznaczonego rozwiązania, lecz mogą być one niemożliwe do wykorzystania w sytuacji braku wystarczającej wiedzy o właściwościach funkcji kryterialnej [8].

Ponieważ dla funkcji kryterialnych (F_c), przedstawionych w dalszej części monografii, ich optima globalne nie są na ogół znane, stąd zaproponowane kryteria oceny dokładności (1.1) i (1.2) znalezionego – za pomocą wybranej metody optymalizacji – rozwiązania mogą być trudne, a nawet wręcz niemożliwe do zastosowania. Dlatego też, weryfikację wyników – rozwiązań zadań optymalizacji – uzyskanych z zastosowaniem modeli matematycznych można przeprowadzić za pomocą dobranego stanowiska pomiarowego – modelu empirycznego [9], [10].

W teorii optymalizacji bardzo ważną grupą algorytmów – niezależnych od konkretnych zadań optymalizacji – są, tzw. algorytmy metaheurystyczne. Algorytmy te stanowią swego rodzaju ramy algorytmiczne dostarczające zestawu wytycznych i strategii działania, wspomagających znajdywanie optimum zadanej funkcji kryterialnej F_c.

Na podstawie różnych topologii metaheurystycznych, wyróżniono trzy klasy algorytmów metaheurystycznych:

- algorytmy lokalnego przeszukiwania z ró¿nymi strategiami przeszukiwania, takimi jak: wspinaczkowe (*hill-climbers*), z wielostartem (*multi-start*), ze zmiennym sąsiedztwem VNS (*Variable Neighborhood Search*), z zakazem TS (*Tabu Search*) czy wreszcie algorytm symulowanego wyżarzania SA (*Simulated Annealing*);
- algorytmy konstrukcyjne, w tym, np. algorytm z dynamiczną zmianą funkcji kryterialnej GRASP (*Greedy Randomized Adaptive Search Procedure*);

- algorytmy populacyjne, w tym, np. ewolucyjne, rozproszonego przeszukiwania (*scatter search*), sztuczne systemy immunologiczne oraz algorytmy stadne.

Algorytmy optymalizacyjne można podzielić także, ze względu na liczbę modyfikowalnych rozwiązań, na [11]:

- algorytmy, tzw. „samotnego poszukiwania", operujące na jednym rozwiązaniu;
- algorytmy populacyjne, realizujące proces poszukiwania optymalnego rozwiązania na podstawie ewolucji zbioru rozwiązań w przestrzeni poszukiwań.

Do grupy algorytmów tak zwanego „samotnego poszukiwania" należą takie algorytmy, jak: symulowanego wyżarzania SA, poszukiwania z zakazem TS czy ze zmiennym sąsiedztwem VNS.

Do algorytmów populacyjnych zalicza się algorytmy rojowe, w tym przede wszystkim: sztucznej kolonii pszczół ABC (*Artificial Bee Colony*), pszczeli BeA (*Bee Algorithm*), kukułki CS (*Cuckoo Search*), świetlika FA (*Firefly Algorithm*), optymalizacji rojem cząstek (ptasi) PSO (*Particle Swarm Optimisation*), nietoperza BA (*Bat Algorithm*) i inne [12].

1.2. Wybrane algorytmy optymalizacji

Przedstawione w następnym rozdziale niniejszej pracy funkcje kryterialne są funkcjami nieliniowymi wielu zmiennych, a zatem poszukiwanie ich ekstremów wymaga zastosowania efektywnych metod optymalizacji.

Ze względu na specyfikę planowania i optymalizacji sieci WLAN z infrastrukturą, w celu rozwiązania zadania optymalizacji, rozważono nieliniową, wielowymiarową optymalizację z pojedynczą funkcją kryterialną (F_c), a także optymalizację wielokryterialną.

W szczególności, dla różnych scenariuszy, przedstawionych w dalszej części monografii, rozważono możliwość zastosowania – w procesie poszukiwania ekstremów funkcji kryterialnych (F_c) – takich algorytmów optymalizacyjnych OPA (*Optimisation Algorithms*) jak:

– algorytm symulowanego wyżarzania SA (*Simulated Annealing*) [13], [14];
– algorytm genetyczny GA (*Genetic Algorithm)*;

- algorytm hybrydowy genetyczny GA, połączony z algorytmem deterministycznym NM (*Neldera-Meada*) – GA/NM;
- algorytmy inspirowane zjawiskami przyrodniczymi;
  algorytmy rojowe:
  - sztucznej kolonii pszczół ABC (*Artificial Bee Colony*),
  - nietoperza BA (*Bat Algorithm*),
  - pszczele BeA (*Bee Algorithm*),
  - kukułki CS (*Cuckoo Serach*),
  - świetlika FA (*Firefly Algorithm*),
  - optymalizacji rojem cząstek PSO (*Particle Swarm Optimisation*), nazywanym też algorytmem ptasim.

Po przyjrzeniu się właściwościom tych algorytmów, wymaganiom dotyczącym procedur obliczeniowych oraz wynikom tych obliczeń, szczególną uwagę – w przypadku optymalizacji jednokryterialnej – zwrócono na algorytm symulowanego wyżarzania SA oraz algorytmy rojowe.

Natomiast dla potrzeb optymalizacji wielokryterialnej (zastosowanych w dalszej części monografii) użyto algorytmy: MOGA (*MultiObjective Genetic Algorithm*) w wersji NSGAIII [15], wielokryterialny algorytm ptasi MOPSO (*MultiObjective Particle Swarm Optimisation*) oraz wielokryterialny algorytm kukułki MOCS (*MultiObjective Cuckoo Search).*

1.3. Wybór środowiska obliczeniowego

Funkcjonuje duża liczba programów komputerowych z zakresu optymalizacji. Do najbardziej popularnych programów (pakietów/języków programowania) stosowanych w tym celu możemy zaliczyć m.in.: Mathwork*s* (Matlab) (z dodatkowymi pakietami: Optimisation Toolbox, Global Optimisation Toolbox, CVX, Robust Control Toolbox/LMI Control Toolbox [16]), Mathematica, WolframAlpha, Wolfram/Lipschitz Global Optimizer (LGO), *C++* (biblioteki *alglib*, *stats++*, Dlib, LEMON, *ceres-solver*, OptimLib, CasADi), Python, FSQP (dodatek dla języka Python [17]), IMSL Library (biblioteka dla środowiska Python), Octave (biblioteka GLPK [18], [19]), Scilab, AEM Design (FSQP), Ap-Tech (GAUSS/CML), ARKI (CONOPT), CCLRC (Fortran/GALAHAD), Frontline Solvers (dla Excel a także dla innych środowisk obliczeniowych), Grabitech (Multisimplex), Lindo systems,

Mosek ApS, NAG Numerical Algorithms Group, Pi Blue (OptWorks/Excel), AMPL/LOQO, Quantum Leap Innovations/Aoe-Adaptive Optimisation engine, SOPT, SAS Institute (OR, PROC NLP, PROC OPTMODEL), Klaus Schittkowski/NLPQL, DONLP2, Stanford Optimisation Library [20], Vanderplaats (VisualDOC), KNITRO, COIN-OR [21, 22] i inne [23], [24], [25].

Przykładowo, w popularnym programie Excel dostępny jest wspomniany wcześniej dodatek Solver, który pozwala rozwiązywać problemy programowania liniowego oraz całkowitoliczbowego [26].

Funkcjonują różne kryteria wyboru metody optymalizacji. Mogą nimi być:

- typ problemu optymalizacji (liniowy, nieliniowy);
- rozmiar problemu, liczba zmiennych decyzyjnych oraz liczba ograniczeń;
- liczba kryteriów;
- rodzaj zmiennych decyzyjnych (zmienne ciągłe albo dyskretne);
- typ ograniczeń (równościowe, nierównościowe);
- konieczność obliczania pochodnych lub ograniczeń;
- założona dokładność optymalizacji;
- wymagana niezawodność wyznaczania ekstremum globalnego;
- dostępność funkcjonujących rozwiązań informatycznych – (programów/bibliotek programistycznych);
- łatwość adaptacji dostępnego oprogramowania do rozważanego zadania;
- informacja o użyteczności danego algorytmu w analogicznych problemach;
- akceptowalna funkcjonalność we wprowadzaniu danych oraz interpretowaniu wyników (zaimplementowany interfejs graficzny).

W dalszej części pracy zastosowano pakiet obliczeniowy Matlab z zainstalowanymi pakietami: *Optimisation Toolbox* (tab. 1.1) i *Global Optimisation Toolbox* [27], a także skorzystano z kodów programów dostępnych w projekcie Yarpiz [28] oraz plików projektów udostępnionych w serwisie Matlab File Center [29].

Tabela 1.1. Funkcje (programy) [30] *Matlab Optimisation Toolbox [31]*

Rodzaj zadania optymalizacji	**Opis matematyczny problemu**		**Funkcja/program**
Liniowa	$\min_{x \in R^{n+}} f^T(x) \triangleq c^T x$; Ax≤b; A_{eq}x=b_{eq}; x≥0, x_{lb}≤x≤x_{ub};	(1.3)	linprog(…) [32]
Mieszane programowanie całkowitoliczbowe *Mixed-integer linear programming* (MILP)	$\min_{x \in R^{+}} f^T x$; Ax≤b; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub}; $x \in \mathbb{Z}$;	(1.4)	intlinprog(…) [33]
Nieliniowa jednej zmiennej	$\min_{x \in R^{n+}} f(x)$; x_{lb}≤x≤x_{ub};	(1.5)	fminbnd(…) [34]
Kwadratowa	$\min_{x \in R^{+}} \frac{1}{2} x^T H_q x + f^T x$; Ax≤b; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub};	(1.6)	quadprog(…) [35]
Nieliniowa wielu zmiennych bez ograniczeń	$\min_{x \in R^{n+}} f(x)$;	(1.7)	fminunc(…) [36] fminsearch(…) [37]
Nieliniowa wielu zmiennych z ograniczeniami	$\min_{x \in R^{+}} f(x)$; $c(x) \leq 0, c_{eq}(x) = 0$; (ograniczenia, mogą być nieliniowymi funkcjami) Ax≤b; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub};	(1.8)	fmincon(…) [38]
Programowanie Minimaksowe	$\min_{x} \max_{i} F_i(x)$; $c(x) \leq 0, c_{eq}(x) = 0$; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub};	(1.9)	fminimax(…) [39]
Nieliniowe z ograniczeniami parametryzowanymi	$\min_{x \in R^{+}} f(x)$; $K_i(x, w_i) \leq 0;\ 1 \leq i \leq n$; $c(x) \leq 0$; $c_{eq}(x) = 0$; Ax≤b; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub};	(1.10)	fseminf(…) [40]
Pierwiastek funkcji nieliniowej	$f(x) = 0$;	(1.11)	fzero(…) [41]
Układ równań nieliniowych wielu zmiennych	$F(x) = 0$; (n–równań, n–niewiadomych)	(1.12)	fsolve(…) [42]
Nieliniowe dopasowanie krzywej	$\min_{x,\gamma} \|F(x, xdata) - ydata\|$; x_{lb}≤x≤x_{ub};	(1.13)	lsqcurvefit [43]

Rozwiązywanie problemów liniowych z ograniczeniami metodą najmniejszych kwadratów	$\min_{x\in R^+} \frac{1}{2}\|C\cdot x-d\|_2^2$; Ax≤b; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub};	(1.14)	lsqlin(…) [44]
Rozwiązywanie problemów nieliniowych metodą najmniejszych kwadratów	$\min_x \|f(x)\|_2^2 =$ $= \min_x (f_1(x)^2 + f_2(x)^2 + \ \ldots \ + f_n(x)^2)$; Ax≤b; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub}; $c(x) \le 0$; $c_{eq}(x) = 0$;	(1.15)	lsqnonlin(…) [45]
Rozwiązywanie problemów liniowych metodą najmniejszych kwadratów dla nieujemnych wartości zmiennych	$\min_x \frac{1}{2}\|C\cdot x-d\|_2^2$; x≥0;	(1.16)	lsqnonneg(…) [46]
Programowanie celowe	$\min_{x,\gamma} \gamma$; $F(x) - w\gamma \le cel$; $c(x) \le 0$, $c_{eq}(x) = 0$; Ax≤b; A_{eq}x=b_{eq}; x_{lb}≤x≤x_{ub}.	(1.17)	fgoalattain(…) [47]

Zbiór *Global Optimisation Toolbox* obejmuje pięć funkcji (algorytmów obliczeniowych) [27]:

– patternsearch;
– ga (algorytm genetyczny *Genetic Algorithm* GA);
– particleswarm (algorytm optymalizacji rojem cząstek *Particle Swarm Optimisation* PSO);
– surogateopt;
– GlobalSearch.

1.4. Algorytmy klasyczne

Każdy program liniowy można zapisać w postaci macierzowej (1.3), oraz można przyjąć uproszczenie, że macierz ograniczeń A ma pełny rząd wierszowy:

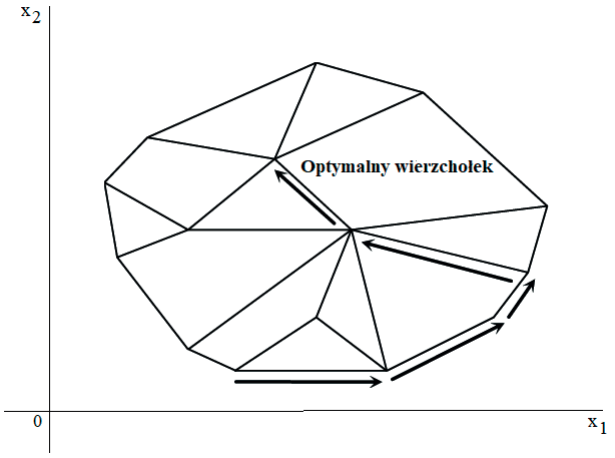
$$rank\ A = m \tag{1.3}$$

Zadanie optymalizacji, w której funkcjonał (funkcja kryterialna) $F_c(\chi)$ oraz ograniczenia typu nierównościowego zadane są przez operatory liniowe, nazywa się zadaniem programowania liniowego [19]. Do rozwiązywania zadań optymalizacji liniowej wykorzystuje się algorytm sympleks [48], [49]. Nazwa metody pochodzi od sympleksu czyli figury wypukłej będącej uogólnieniem m.in. trójkąta na większą liczbę wymiarów (sympleks *n* – wymiarowy to wielokomórka, której ścianami jest *n+1* sympleksów *n–1* wymiarowych).

Algorytm simpleks jest uniwersalną metodą polegającą na badaniu kolejnych rozwiązań bazowych, które stanowią wierzchołki zbioru rozwiązań dopuszczalnych problemu liniowego o postaci kanonicznej.

Wykorzystywane są różne podejścia w przejściach do kolejnych tablic sympleksowych, w których wykorzystywane są przekształcenia algebraiczne [3], [25], [26].

W tej metodzie, optimum jest poszukiwane przez odpowiednie deterministyczne (uporządkowane) przemieszczanie się po punktach wierzchołkowych zbioru punktów dopuszczalnych χ_0 (tworzące obszar dopuszczalny).



Rysunek 1.2. Przykładowe działanie algorytmu sympleks [50]

Podczas działania algorytmu, generowane są kolejne punkty o wartościach funkcji celu nie gorszej niż w poprzednim punkcie [25].

Zadanie programowania liniowego może mieć: jedno rozwiązanie (punkt ekstremalny zbioru rozwiązań dopuszczalnych), nieskończenie wiele rozwiązań (brzeg obszaru rozwiązań dopuszczalnych), ale również może nie mieć rozwiązania [16].

W standardowej postaci rozważa się minimalizację funkcji kryterialnej (funkcji celu). W przypadku rozwiazywania zadania maksymalizacji wykorzystuje się własność:

$$\max_x f_c(x) = -\min_x\big(-f_c(x)\big). \tag{1.19}$$

Najprostszą metodą rozwiazywania zadań optymalizacji liniowej jest metoda graficzna, a uniwersalnym sposobem jest macierzowa metoda sympleks [51].

Tabela 1.2. Zdefiniowanie przykładu 1 – optymalizacja liniowa z liniowymi ograniczeniami nierównościowymi

Rozwiąż zadanie:

$$\max_x f_c(x) = 11x_1 + 7x_2, \tag{1.20}$$

przy ograniczeniach:

$$3x_1 + 10x_2 \leq 60, \tag{1.21}$$

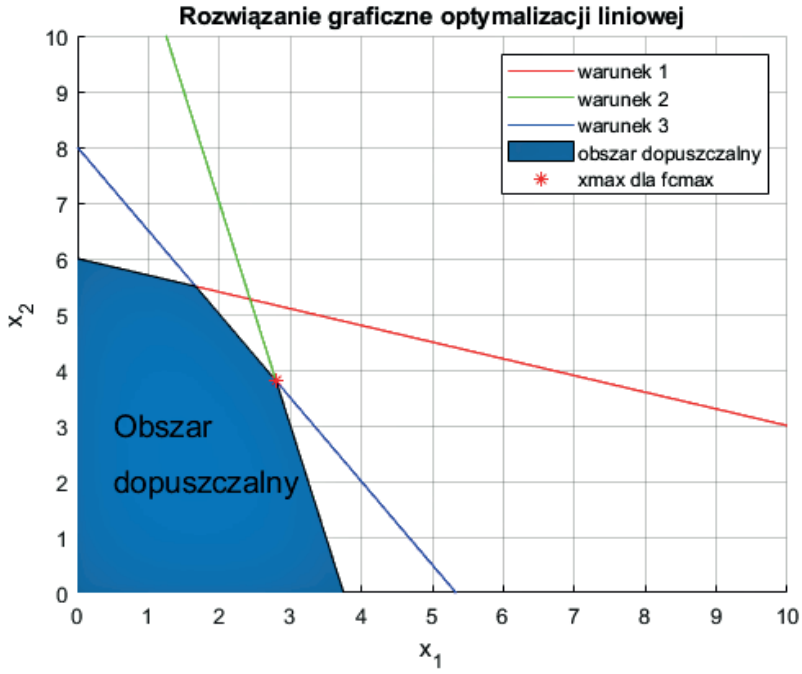
$$4x_1 + x_2 \leq 15, \tag{1.22}$$

$$3x_1 + 2x_2 \leq 16, \tag{1.23}$$

oraz:

$$x_1, x_2 \geq 0.$$

Procedurę rozwiązania zadania optymalizacji liniowej szczegółowo przedstawiono w publikacji [49]. Zastosowanie funkcji **linprog** [52] w rozwiązaniu zadania optymalizacji liniowej z liniowymi ograniczeniami nierównościowymi z przykładu 1 (tabela 1.2) przedstawiono w tabeli 1.3.



Rysunek 1.3. Wizualizacja graficznego rozwiązania optymalizacji liniowej dla przykładu 1 (prezentacja ograniczona dla dodatnich wartości zmiennych)

Tabela 1.3. Rozwiązanie zadania programowania liniowego z przykładu 1

```
clear; close all; clc;
% parametry optymalizacji – opisanie zadania – przykład 1
fc= [11;7]; %współczynniki c ze wzoru (1.3) oraz (1.20) – funkcji
% kryterialnej
A=[3 10; 4 1;3 2 ]; % współczynniki ograniczeń wzory (1.21–1.23)
b=[60;15;16]; % współczynniki nierównościowe ograniczeń –
% wzory (1.21–1.23)
figure(1);
dimx=10;
axes('Xlim',[0 dimx],'Ylim',[0 dimx]);
axis([0 dimx 0 dimx]);
hold on;
xlabel('x_1');
ylabel('x_2');
grid on
color=['r','g','b'];
x1= linspace(0,dimx,dimx^2);
for i=1:size(A,1)
    x2=(b(i)-A(i,1)*x1)/A(i,2);
    line(x1,x2,'Color',color(i));
end
xa=[0 b(2)/A(2,1)];
ya=[b(1)/A(1,2) 0 ];
x3=A(2:3,:)\b(2:3,1);
x4=A(1:2:3,:)\b(1:2:3,1);
xa=[0 b(2)/A(2,1) x3(1) x4(1)];
ya=[b(1)/A(1,2) 0 x3(2) x4(2)];
area(xa, ya);
title('Rozwiązanie graficzne optymalizacji liniowej');
text(0.5,3,'Obszar','Color','black','FontSize',14);
text(0.5,2,'dopuszczalny','Color','black','FontSize',14);
optim= optimset('Display','iter');
[xmax,fval,exitflag,output,lambda]=linprog(-fc,A,b,[],[],[],[], optim);
fcmax=-fval;
plot(xmax(1),xmax(2),'*r');
legend('warunek 1','warunek 2','warunek 3','obszar dopuszczalny','xmax dla
fcmax');
```

Rozwiązaniem liniowego zadnia optymalizacji z przykładu 1 jest wektor χ_{max} (tab. 1.3) z wartościami $\chi_{max}=\begin{bmatrix}2,8\\3,8\end{bmatrix}$, dla których osiągnięto maksymalną wartość funkcji kryterialnej $f_c^{max}=57,4$.

Programowanie liniowe całkowitoliczbowe

Optymalizacja całkowitoliczbowa, dotyczy tych problemów, w których na wartość zmiennych decyzyjnych nałożony jest warunek całkowitoliczbowości, a relacje występujące między wielkościami są liniowe [51].

Tabela 1.4. Porównanie optymalizacji liniowej oraz całkowitoliczbowej [51]

```
clear; close all; clc;
f=[2; 3]; % maksymalizowana funkcja kryterialna
A=[195 273; 4 40];
b=[1365; 140];
lb= [0 0];
ub=[ 4 inf];
% optymalizacja liniowa – zastosowanie funkcji linprog (1.3)
[x fval]=linprog(-f,A,b,[],[],lb,ub);
% optymalizacja całkowitoliczbowa – obie zmienne intcon=[1, 2] –
% zastosowanie funkcji intlinprog (1.4)
intcon=[1, 2];
[xc fvalc] = intlinprog(-f,intcon,A,b,[],[],lb,ub);
```

Analizując rozwiązania uzyskane z zastosowaniem programu przedstawionego w tabeli 1.4 dotyczące zadania:

$$\max_x f_c(x) = 2x_1 + 3x_2, \tag{1.24}$$

przy ograniczeniach:

$$195x_1 + 273x_2 \leq 1365, \tag{1.25}$$

$$4x_1 + 40x_2 \leq 140, \tag{1.26}$$

$$x_1, x_2 \geq 0, x_1 \leq 4, \tag{1.27}$$

można zauważyć, że optymalne rozwiązanie nie jest zaokrągleniem rozwiązania optymalizacji liniowej, gdyż uzyskano rozwiązania: $x = \begin{bmatrix} 2{,}44 \\ 3{,}26 \end{bmatrix}$ dla optymalizacji liniowej oraz $x_c = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$ dla optymalizacji całkowitoliczbowej.

1.4.1. Nieliniowe zagadnienia optymalizacyjne

Optymalizacją nieliniową nazywa się zadanie postaci [26]:

$$f(x) = f(x_1, \ldots, x_n) \rightarrow \text{minimum (lub maksimum)}, \tag{1.28}$$

$$g(x) = g_i(x_1, \ldots, x_n) \leq 0 \text{ (lub } \geq 0) \ (i = 1, \ldots, r), \tag{1.29}$$

$$x_1, \ldots, x_n \geq 0. \tag{1.30}$$

Przy czym przynajmniej jedna z funkcji *f(x)* lub $g(x)$ nie jest funkcją liniową, przy czym zakłada się, że funkcje *f(x)* oraz $g(x)$ są ciągłe.

1.4.2. Metoda mnożników Lagrange'a

Problem programowania nieliniowego z ograniczeniami, można sprowadzić do problemu programowania nieliniowego bez ograniczeń przez wprowadzenie dodatkowych zmiennych bilansujących oraz zdefiniowania nowej funkcji celu tzw. Lagrangianu [53].

W optymalizacji nieliniowej, w zadaniu przedstawionych w postaci kanonicznej, metodę mnożników Lagrange'a można podzielić na dwa etapy.

Pierwszym jest sprawdzenie ekstremum bezwarunkowego oraz spełnienie warunków ograniczających. W takim przypadku ekstremum bezwarunkowe jest także ekstremum warunkowym funkcji celu. Warunkiem wystarczającym na istnienie minimum funkcji *f(x)* w punkcie $x_0 = (x_1, x_2, \dots, x_n)$ jest dodatnia wartość wyznacznika hesjanu (macierzy Hessego) $H(x)$ pochodnych cząstkowych drugiego rzędu z tej funkcji (1.31), pod warunkiem, że *x* jest punktem krytycznym funkcji *f(x)*, która jest klasy C^2. Dodatkowo wartości wszystkich **minorów głównych** tej macierzy w punktach stacjonarnych (podejrzanych o ekstremum) były dodatnie. Jeśli ekstremum bezwarunkowe spełnia dodatkowo warunki ograniczające, to zadanie posiada rozwiązanie.

$$H(\boldsymbol{x}) = \begin{vmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{vmatrix} > 0. \tag{1.31}$$

Jeśli ekstremum bezwarunkowe nie spełnia ograniczeń, to wprowadza się funkcję, w której uwzględnione zostanie zbiór rozwiązań dopuszczalnych.

Dla odwzorowania: $f: \mathbb{R}^n \to \mathbb{R},$ (1.32)

z ograniczeniami: $g_j(x) = b_j, \ \text{dla}\ j = 1,2,\dots,r,$ (1.33)

otrzymujemy funkcję Lagrange'a:

$$L(x,\lambda) = f(x) + \lambda^T \left[\overbrace{b - g(x)}^{g_t(x)} \right]^T. \tag{1.34}$$

Jeżeli funkcja Lagrange'a $L(x, \lambda)$ ma ciągłe pochodne cząstkowe, a $x^* \in \mathbb{R}^n$ jest rozwiązaniem optymalnym problemu z ograniczeniami, to istnieje wektor λ^* taki, że pochodne cząstkowe funkcji Lagrange'a $L(x, \lambda)$ w punkcie (x^*, λ^*) są równe zeru.

Jeżeli funkcja celu jest wypukła, a ograniczenia $g(x)$ są liniowe (1.33), to znalezione rozwiązanie stanowi minimum globalne.

Dzięki tej metodzie, problem optymalizacji z ograniczeniami, za cenę wzrostu liczby zmiennych optymalizacji, zostaje zamieniony na odpowiedni problem optymizacji bez ograniczeń.

W celu sprawdzenia otrzymanego rozwiązania, dla każdego obliczonego punktu stacjonarnego („podejrzanego" o ekstremum) oblicza się wyznacznik hesjanu.

Dla ekstremów lokalnych funkcji dwóch zmiennych – bez ograniczenia, hesjan przyjmuje postać:

$$H(x_1, x_2) = det \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_2^2} \end{bmatrix}. \tag{1.35}$$

Jeżeli:

$H(x_1, x_2) > 0$ oraz $\frac{\partial^2 f(x)}{\partial x_1^2} > 0$ – otrzymano minimum w punkcie (x_1, x_2),

$H(x_1, x_2) > 0$ oraz $\frac{\partial^2 f(x)}{\partial x_1^2} < 0$ – otrzymano maksimum w punkcie (x_1, x_2),

$H(x_1, x_2) < 0$ – nie występuje ekstremum w punkcie (x_1, x_2),

$H(x_1, x_2) = 0$ – nie można stwierdzić występowania ekstremum w punkcie (x_1, x_2)

Przedstawione testowanie przeprowadza się dla wszystkich punktów stacjonarnych. Każde otrzymane rozwiązanie układu równań (pochodne z $L(x, \lambda)$ jest „podejrzane" o to, że istnieje w nim lokalne ekstremum warunkowe.

Sprawdzenie warunku koniecznego polega na policzeniu dla każdego punktu stacjonarnego – wyznacznika hesjanu obrzeżonego (H_k):

$$H_k = det\begin{bmatrix} & & & & -\frac{\partial g_{t1}(x)}{\partial x_1} & \dots & -\frac{\partial g_{t1}(x)}{\partial x_k} \\ 0 & \dots & 0 & & \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots & & -\frac{\partial g_{tr}(x)}{\partial x_1} & \dots & -\frac{\partial g_{tr}(x)}{\partial x_k} \\ 0 & \dots & 0 & & & & \\ -\frac{\partial g_{t1}(x)}{\partial x_1} & \dots & -\frac{\partial g_{tr}(x)}{\partial x_1} & \frac{\partial^2 L(x,\lambda)}{\partial x_1^2} & \dots & \frac{\partial^2 L(x,\lambda)}{\partial x_1 \partial x_k} & \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \\ -\frac{\partial g_{t1}(x)}{\partial x_k} & \dots & -\frac{\partial g_{tr}(x)}{\partial x_k} & \frac{\partial^2 L(x,\lambda)}{\partial x_k \partial x_1} & \dots & \frac{\partial^2 L(x,\lambda)}{\partial x_k^2} & \end{bmatrix}, \quad (1.36)$$

$$k = 2, 3, \dots, n.$$

Jeśli w danym punkcie $x_0 = (x_1, x_2, \dots, x_n)$ spełniony jest warunek konieczny istnienia ekstremum warunkowego:

$$\forall_{i=1,2,\dots,n} \frac{\partial L(x,\lambda)}{\partial x_i} = 0 \ \wedge \frac{\partial L(x,\lambda)}{\partial \lambda} = 0, \quad (1.37$$

Prawdziwe są twierdzenia:

- jeśli $\forall_{k=2,\dots,n} H_k(x_0, \lambda) < 0$, to funkcja *f(x)* przyjmuje minimum warunkowe w punkcie x_0 ,
- jeśli $\forall_{k=2,\dots,n} (-1)^{k+1} H_k(x_0, \lambda) < 0$, to funkcja *f(x)* przyjmuje maksimum warunkowe w punkcie x_0 ,
- sytuacja nie jest rozstrzygnięta, gdy $H_k(x_0, \lambda) = 0$.

Istnienie ekstremum należy analizować innymi metodami (np. bada się hesjan na wektorach przestrzeni stycznej) [54].

Dla funkcji dwóch zmiennych n=2 (oraz jednego ograniczenia r=1) wyznacznik hesjanu obrzeżonego zdefiniowano jako:

$$H_2 = det\begin{bmatrix} 0 & -\frac{\partial g_{t1}(x)}{\partial x_1} & -\frac{\partial g_{t1}(x)}{\partial x_2} \\ -\frac{\partial g_{t1}(x)}{\partial x_1} & \frac{\partial^2 L(x,\lambda)}{\partial x_1^2} & \frac{\partial^2 L(x,\lambda)}{\partial x_1 \partial x_2} \\ -\frac{\partial g_{t1}(x)}{\partial x_2} & \frac{\partial^2 L(x,\lambda)}{\partial x_1 \partial x_2} & \frac{\partial^2 L(x,\lambda)}{\partial x_2^2} \end{bmatrix}. \quad (1.38)$$

Jeżeli otrzymana wartość wyznacznika hesjanu obrzeżonego (1.38) jest mniejsza od 0 to osiągnięto minimum w punkcie stacjonarnym. Gdy wyznacznik hesjanu obrzeżonego ma wartość większą od 0, oznacza to, że w rozważanym punkcie osiągnięto maksimum.

Programowanie nieliniowe z zastosowaniem metody mnożników Lagrange'a przedstawiono w przykładzie 2 (tab. 1.5) [26].

Tabela 1.5. Problem oraz rozwiązanie przykładu 2 [26]

Z elektrociepłowni energia przesyłana jest do dwóch zużywających ją zakładów produkcyjnych. Funkcja kosztów *f(x)* przesyłania energii do tych zakładów w zależności od wielkości przesyłu – odpowiednio x_1, x_2 do pierwszego i drugiego zakładu wynosi:

$$f(x) = 5x_1^2 - 8x_1x_2 + 7x_2^2 - 12x_1 - 4x_2 + 81. \tag{1.39}$$

Należy rozdzielić dzienną produkcję energii wynoszącą 16 MWh między dwa zakłady, tak aby minimalizować koszty związane z przesłaniem energii.

Dla tego zadania nałożone ograniczenie można zapisać jako:

$$g(x) = x_1 + x_2 = 16, x_j \geq 0; j \in [1,2]. \tag{1.40}$$

Sprawdzenie ekstremum bezwarunkowego:

$$\frac{\partial f(x)}{\partial x_1} = 10x_1 - 8x_2 - 12 = 0, \tag{1.41}$$

$$\frac{\partial f(x)}{\partial x_2} = -8x_1 + 14x_2 - 4 = 0, \tag{1.42}$$

Rozwiązaniem jest: $x = \left(2\frac{12}{19}, 1\frac{15}{19}\right)$. Funkcja (1.39) może posiadać ekstremum bezwarunkowe tylko w tym punkcie.

$$|H(x_1, x_2)| = \det\begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_2^2} \end{bmatrix} = \det\begin{bmatrix} 10 & -8 \\ -8 & 14 \end{bmatrix} = 76 > 0. \tag{1.43}$$

Spełniony jest warunek wystarczający na istnienie minimum funkcji *f(x)* w punkcie $x = x_{min} = \left(2\frac{12}{19}, 1\frac{15}{19}\right)$, gdyż otrzymano dodatnią wartość wyznacznika macierzy z pochodnych cząstkowych drugiego rzędu H(x_1, x_2)z tej funkcji (1.43). Dodatkowo minor główny macierzy H(x_1, x_2) czyli $\frac{\partial^2 f(x)}{\partial x_1^2} = 10 > 0$ ma wartość dodatnią. Minor główny to minor (wyznacznik macierzy), w którym przy wykreślaniu pozostawiono wiersze i kolumny o równych indeksach.

Otrzymane rozwiązanie jest mniejsze od ograniczenia równościowego (1.40), więc analizie poddano rozwiązanie z ograniczeniem.

Lagrangian można zapisać jako:

$$L(x,\lambda_1) = 5x_1^2 - 8x_1x_2 + 7x_2^2 - 12x_1 - 4x_2 + 81 + \lambda_1 \overbrace{(16 - x_1 - x_2)}^{g_{t1}(x)}, \quad (1.44)$$

oraz pochodne:

$$\frac{\partial L(x,\lambda_1)}{\partial x_1} = 10x_1 - 8x_2 - 12 + \lambda_1 = 0, \quad (1.45)$$

$$\frac{\partial L(x,\lambda_1)}{\partial x_2} = -8x_1 + 14x_2 - 4 + \lambda_1 = 0, \quad (1.46)$$

$$\frac{\partial L(x,\lambda_1)}{\partial \lambda_1} = 16 - x_1 - x_2 = 0. \quad (1.47)$$

Prowadzi to do rozwiązania układu równań liniowych z trzema niewiadomymi (x_1, x_2, λ_1) = (9,7–22), a wartość funkcji w tym punkcie wynosi $f(x_1, x_2)$ =189 jednostek pieniężnych. Rozwiązanie układu trzech równań pochodne z $L(x_1, \lambda_1)$ (1.45–1.44) „podejrzane" jest o lokalne ekstremum warunkowe.

Sprawdzenie warunku koniecznego polega na policzeniu dla każdego punktu stacjonarnego – wyznacznika hesjanu obrzeżonego opisanego równaniem 1.38.

Analiza przykładu 2 prowadzi do wyniku:

$$H_2 = det\begin{bmatrix} 0 & 1 & 1 \\ 1 & 10 & -8 \\ 1 & -8 & 14 \end{bmatrix} = -40 < 0 \quad (1.48)$$

Otrzymana wartość H_2 jest mniejsza od 0, oznacza to, że osiągnięto minimum (z ograniczeniem 1.40) w punkcie stacjonarnym $(x_1^{minogr}; x_2^{minogr}) = \begin{bmatrix} 9 \\ 7 \end{bmatrix}$. Gdyby wyznacznik hesjanu obrzeżonego H_2 był większy od 0, oznaczałoby osiągnięcie maksimum w tym punkcie.

Tabela 1.6. Rozwiązanie zadania programowania nieliniowego (kwadratowego) dla przykładu 2

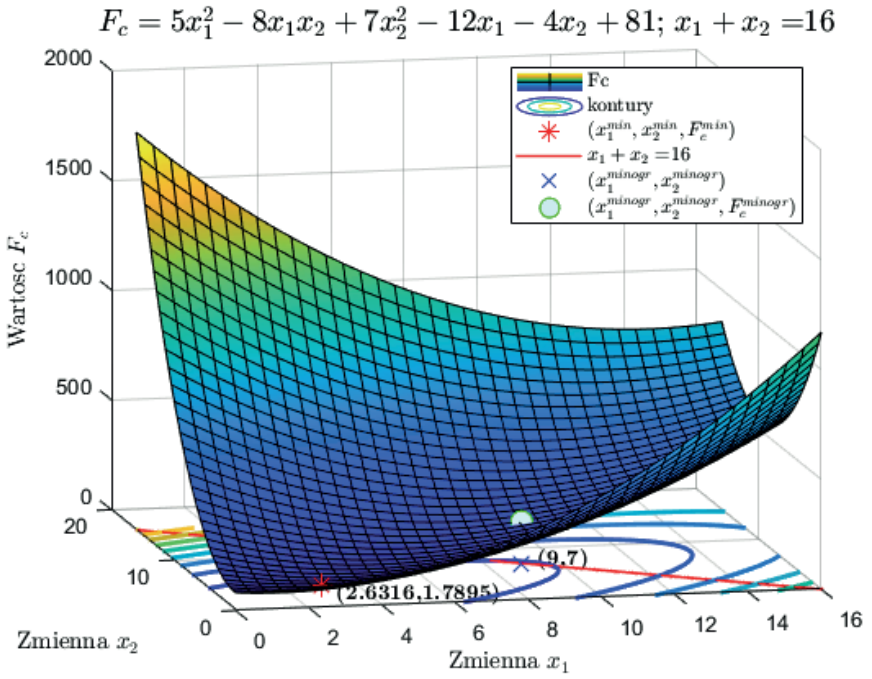
```
clear; close all; clc;
przesuniecie=81;
Fc = @(x1,x2) 5*x1.^2-8*x1.*x2+7*x2.^2-12*x1-4*x2+przesuniecie; % wzór
%(1.39)
% ustalenie parametrów dla przykładu 2
H = [2*5 -8; -8 2*7]; % parametry ze wzoru (1.43)
f = [-12; -4];
Aeq = [1 1];
beq = [16];
fig=figure;
hold on
grid on
zakres=[0 1  0 1]*16;
fsurf(Fc,zakres);
fcontour(Fc,zakres,'LineWidth', 2);
x0=[0 0]; %punkt startowy optymalizacji
options = optimset('TolFun',1e-6,'TolX',1e-6, 'Display','iter');
[xmin,Fcmin,exitflag, output] =
fminsearch(@(data)Fc(data(1),data(2)),x0,options)
plot3(xmin(1),xmin(2),Fcmin,'r*','MarkerSize',10);
[xminogr,fminogr,exitflag,output,lambda] =
quadprog(H,f,[],[],Aeq,beq,[],[],[],options);
% [xmin,Fcmin] =
%fmincon(@(data)Fc(data(1),data(2)),x0,[],[],Aeq,beq,[],[],[],optio% ns);
x=0:0.1:beq;
line=beq-x;
plot(x,line,'-rs','LineWidth',1,'MarkerSize',1);
plot(xminogr(1),xminogr(2), 'bx','MarkerSize',10);
plot3(xminogr(1),xminogr(2), fminogr+przesuniecie,
'go','MarkerSize',10,'MarkerFaceColor','#D9FFFF');
set(fig,'defaultTextInterpreter','latex');
title(['$F_c=5x_{1}^2-8x_{1}x_{2}+7x_{2}^2-12x_{1}-4x_{2}+81$;
',strcat('$x_1+x_2=$', num2str(beq))], 'FontSize',14);
xlabel('Zmienna $x_1$');
ylabel('Zmienna $x_2$');
zlabel('Wartosc $F_c$');
legend({'Fc','kontury','$(x_{1}^{min},  x_{2}^{min},
F_{c}^{min})$',strcat('$x_1+x_2=$',num2str(beq)),'$(x_{1}^{minogr},
x_{2}^{minogr})$','$(x_{1}^{minogr},  x_{2}^{minogr},
F_{c}^{minogr})$',},'interpreter','latex');
d=0.5;
text(xmin(1)+d,xmin(2)+d,strcat('\bf (' , num2str(xmin(1)),',',
num2str(xmin(2)),')'));
text(xminogr(1)+d,xminogr(2)+d,strcat('\bf (' , num2str(xminogr(1)),',',
num2str(xminogr(2)),')'));
```

Do rozwiązania zadania optymalizacji z przykładu 2 (rys. 1.4) można zastosować funkcję ***fmincon*** (1.8) lub ***quadprog*** (programowanie kwadratowe) z pakietu Optimisation Toolbox środowiska Matlab (tab 1.5). Relacje między opisem wyrażonym wzorem (1.6), a inną standardową postacią programowania kwadratowego przedstawia zależność:

$$\min\{c_1x_1 + c_2x_2 + \cdots + h_1x_1^2 + h_{12}x_1x_2 + \cdots + h_2x_2^2 + h_{23}x_2x_3\}, \tag{1.49}$$

oraz:

$$H_q = \begin{bmatrix} 2h_1 & \cdots & h_{1n} \\ \vdots & \ddots & \vdots \\ h_{1n} & \cdots & 2h_n \end{bmatrix}, c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}. \tag{1.50}$$



Rysunek 1.4. Rozwiązanie zadania optymalizacji nieliniowej (przykład 2) z zastosowaniem kodu przedstawionego w tabeli 1.5

W przypadku optymalizacji nieliniowej, w których wszystkie ograniczenia są nierównościowe, można rozwiązać metody nieoznaczonych mnożników Lagrange'a. W tym celu należy zamienić nierówności na równania przez wprowadzenie zmiennych nieistotnych u^2 [26].

Gdyby w przykładzie 2 warunek ograniczający (1.40) miał postać:

$$x_1 + x_2 \leq 16, \tag{1.51}$$

wówczas równanie ograniczające (1.51) może przyjąć postać:

$$x_1 + x_2 - 16 + u^2 = 0. \tag{1.52}$$

W takim przypadku równania (1.45) oraz (1.46) pozostają niezmienione, a układ równań rozszerzony jest o dwa kolejne równania (1.53) oraz (1.54):

$$\frac{\partial L(x, \lambda_1)}{\partial \lambda_1} = x_1 + x_2 - 16 + u^2 = 0, \tag{1.53}$$

$$\frac{\partial L(x, \lambda_1)}{\partial u} = 2u\lambda_1 = 0. \tag{1.54}$$

W takiej sytuacji może znaleźć zastosowanie twierdzenie *Kuhna-Tuckera*.

Jedną z możliwych wersji ostatecznych postaci warunków *Kuhna-Tuckera* [26], [55], [56], [57] (występującymi także pod nazwą warunków *Karusha-Kuhna-Tuckera* KKT) dla praktycznego zastosowania minimalizacji funkcji kryterialnej i ograniczeń $g(x) = g_i(x_1, \ldots, x_n) \leq 0$, może być zastosowanie układu równań (1.55–1.60):

$$\frac{\partial L(x, \lambda_1)}{\partial x_j} = \frac{\partial f(x)}{\partial x_j} + \sum_{i=1}^{r} \lambda_1 \frac{\partial g_i(x)}{\partial x_j} - v_j = 0, \tag{1.55}$$

$$\text{dla } j = 1, \ldots, n,$$

$$\sum_{\mathrm{j}=1}^{r} v_j x_j = 0, \tag{1.56}$$

$$g_i(x) + w_i = 0, \text{dla } i = 1, \ldots, r, \tag{1.57}$$

$$\sum_{\mathrm{i}=1}^{r} w_i \lambda_i = 0, \tag{1.58}$$

$$x_j \geq 0, \text{dla } j = 1, \ldots, n, \tag{1.59}$$

$$\lambda_i \geq 0, \text{dla i} = 1, \ldots, r. \tag{1.60}$$

Dla analizowanego nierównością 1.51 warunku ograniczającego (dla przykładu 2) przedstawiony zestaw równań (1.55)–(1.60) prowadzi do rozwiązania optymalnego dla: $w_1 = 0$, $v_1 = 0$, $v_2 = 0$. W ogólnym przypadku należy zbadać wszystkie rozwiązania spełniające równania: (1.56) oraz (1.58) [26].

Zastosowanie metody Lagrange'a w optymalizacji przepustowości systemów MIMO

Podstawową ideą techniki wieloantenowej MIMO (*Multiple Input Multiple Output*) jest wykorzystanie większej liczby anten po jednej lub po obu stronach łącza bezprzewodowego [58], [59].

Miarą ilości informacji, która może być przesłana i odebrana, nazywa się przepustowością kanału. Zgodnie z twierdzeniem Shannona-Hartleya, w łączu o znanej szerokości pasma (B Hz), przy założeniu, że moc sygnału jest ograniczona, przepustowość kanału C_{SISO} b/s dla układu jednoantenowego SISO (*Single Input Single Output*) jest zdefiniowana jako:

$$C_{SISO} = B \cdot log_2\left(1 + \frac{P}{N_0 B}|h|^2\right) = B \cdot log_2\left(1 + \frac{P}{\sigma^2}|h|^2\right) = B \cdot log_2(1 + \rho|h|^2), \tag{1.61}$$

gdzie: B – szerokość pasma, P – poziom mocy sygnału, p – stosunek mocy sygnału P do mocy szumu ($\sigma^2 = N_0 B$) SNR (*Signal to Noise Ratio*), wyrażony w skali liniowej, N_0 – gęstość widmowej mocy szumów AWGN (*Additive White Gaussian Noise*), h – funkcja przejścia kanału radiowego.

Dla systemów MIMO, w których po obu stronach łącza występuje większa liczba anten, przepustowość można wyrazić zależnością:

$$C_{MIMO} = B \cdot log_2\left[det\left(I_{N_R} + \frac{\rho}{N_T} H H^H\right)\right], \tag{1.62}$$

gdzie: det( ) – wyznacznik macierzy,

I_{N_R}– macierz jednostkowa o wymiarach N_R x N_R (główną przekątną stanowią jedynki, zaś pozostałe jej elementy to zera),

$(.)^H$ – operacja sprzężenia i transpozycji macierzy H (tzw. sprzężenie hermitowskie),

H – macierz transmitacji, która wiąże wejście z wyjściem w przypadku układów o wielu wejściach i wyjściach:

$$y_{N_R} = \sum_{j=1}^{N_T} h_{N_R j}\, x_j + n_{N_R}, \tag{1.63}$$

gdzie: N_T – liczba anten po stronie nadawczej, N_R – liczba anten po stronie odbiorczej.

Dla systemu MIMO, w przypadku deterministycznym – kiedy nadajnik i odbiornik posiadają pełną informację o kanale CSI *(Channel State Information)*, przepustowość systemu można określić z wykorzystaniem dekompozycji SVD *(Singular Value Decomposition)* macierzy kanału. Warto wspomnieć, że w standardzie IEEE 802.11az informacje CSI mogą zostać wykorzystane do dokładnego pozycjonowania urządzeń – także z zastosowaniem uczenia maszynowego [60].

Rozkład macierzy transmitancji na wartości osobliwe można opisać jako:

$$H = U\Lambda V^H, \tag{1.64}$$

gdzie: H i V są macierzami unitarnymi, czyli spełniającymi zależność $U^H{+}UU^H{=}1$, A jest macierzą mającą wartości niezerowe (wartości szczególne macierzy H) wyłącznie na głównej przekątnej oraz $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$, a $n = \min(N_T, N_R)$.

Przepustowość kanału MIMO wynosi:

$$C_{MIMO} = B \cdot \sum_{i=1}^{n} log_2\left(1 + \frac{P_i}{\sigma^2}\lambda_i^2\right). \tag{1.65}$$

W rozważanym systemie występują ograniczenia na całkowitą emitowaną moc:

$$P = \sum_{i=1}^{n} P_i. \tag{1.66}$$

Analogicznie do równania (1.30) można zapisać funkcję Lagrange'a:

$$F(P_1, P_2, \ldots, P_n, L) = B \cdot \sum_{i=1}^{n} log_2\left(1 + \frac{P_i}{\sigma^2}\lambda_i^2\right) + L\left(P - \sum_{i=1}^{n} P_i\right), \tag{1.67}$$

oraz:

$$\frac{\partial F}{\partial P_1} = \frac{\partial F}{\partial P_2} = \cdots = \frac{\partial F}{\partial P_n} = \frac{\partial F}{\partial \mathrm{L}} = 0. \tag{1.68}$$

Ponieważ:

$$\frac{\partial F}{\partial P_i} = \frac{\partial\left(B \cdot log_2\left(1 + \frac{P_i}{\sigma^2}\lambda_i^2\right)\right)}{\partial P_i} = \frac{\partial\left(\frac{B \cdot \ln\left(1 + \frac{P_i}{\sigma^2}\lambda_i^2\right)}{\ln 2}\right)}{\partial P_i} - \mathrm{L} = 0, \tag{1.69}$$

to korzystając ze wzoru na pochodną logarytmiczną funkcji: $(lnf)' = \frac{f'}{f}$, otrzymano:

$$\frac{\frac{\lambda_i^2}{\sigma^2}}{\left(1+\frac{P_i}{\sigma^2}\lambda_i^2\right) ln2} - \mathrm{L} = 0. \tag{1.70}$$

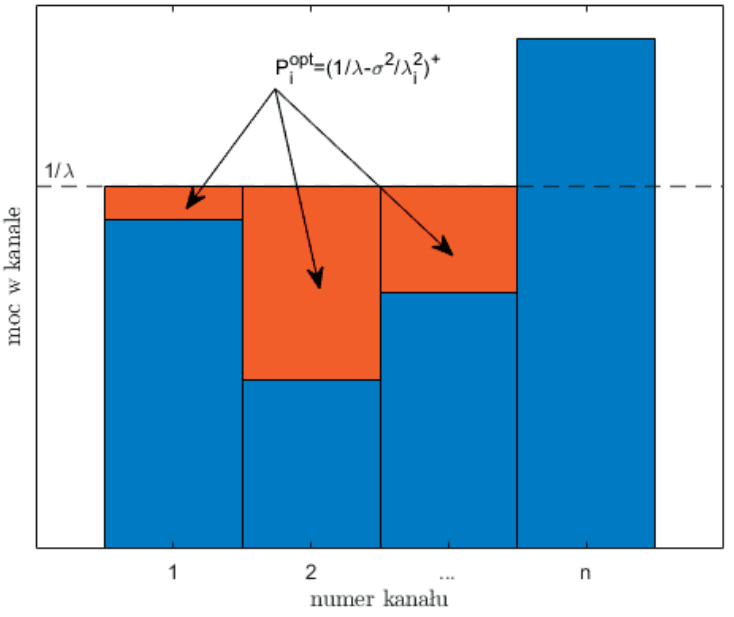
Po przekształceniach równania (1.66) uzyskano:

$$P_i = P_i^{opt} = \left(\frac{1}{\lambda} - \frac{\sigma^2}{\lambda_i^2}\right)^+, \tag{1.71}$$

gdzie: $\lambda + Lln2$, $(x)^+ = \max(0, x)$, w celu uniknięcia przekroczenia linii $\frac{1}{\lambda}$ (rys. 1.5) i przydziału zbyt dużej mocy w kanale radiowym.

Realizacja takiego podejścia znana jest w literaturze jako algorytm zalewowy (*waterfilling*), funkcjonuje także pod nazwą: algorytmu lania wody (*water pouring*), gdzie poziom linii$\frac{1}{\lambda}$(rys. 1.5) podnoszony jest w sposób iteracyjny tak długo, aż pole powierzchni wieloboku odpowiadające wartościom mocy przydzielonych do poszczególnych kanałów nie przekroczy *P*.

W przypadku, gdy wzmocnienie kanału λ_i^2 jest małe, przyjmuje się arbitralnie wartość P_i =0. Następnie oblicza się wartość stałej λ_i z pominięciem tego kanału.



Rysunek 1.5. Przydział mocy dla poszczególnych anten z zastosowaniem algorytmu zalewowego

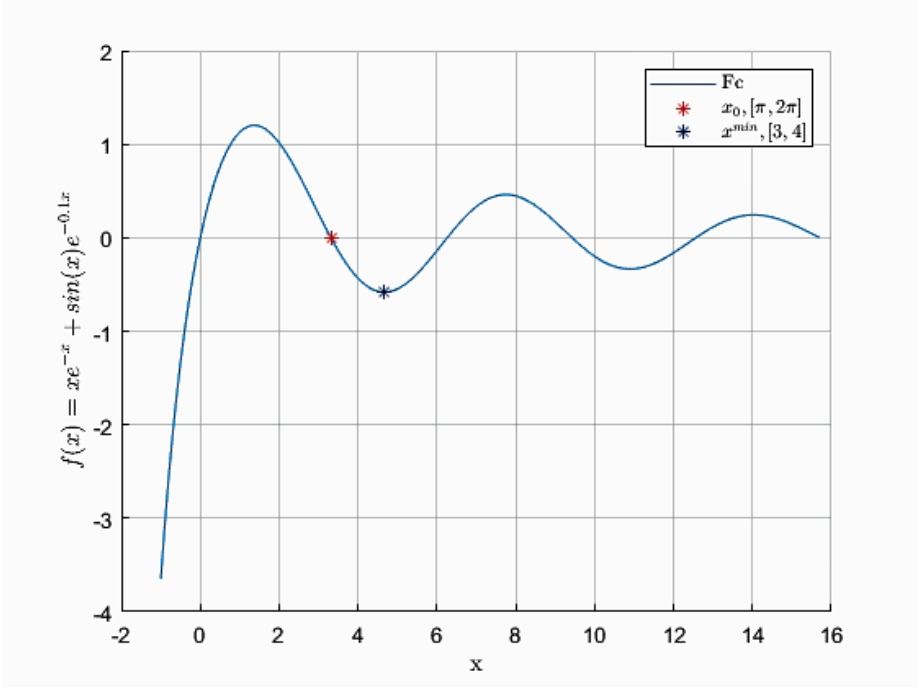
1.4.3. Wybrane algorytmy optymalizacji nieliniowej

Osobną grupę algorytmów tworzą metody optymalizacji nieliniowej, dla których funkcja celu albo ograniczenia są nieliniowe.

Wśród takich metod możemy wyróżnić metody bezgradientowe (bezpośredniego szukania), jak i metody gradientowe [6].

Do optymalizacji nieliniowej jednej zmiennej w przedziale można zastosować klasyczne metody bezgradientowe: złotego podziału, Fibonacciego czy oparte na interpolacji Lagrange'a [6]. W pakiecie Matlab do takiego typu optymalizacji można zastosować funkcję ***fminbnd*** (1.5) [34].

Przykład zastosowania optymalizacji nieliniowej jednej zmiennej w zadanym przedziale przedstawiono w tabeli 1.7 oraz rysunku 1.6. W analizowanym przykładzie, w założonym przedziale [π 2π], otrzymano minimum w punkcie $x^{\min} = 4{,}6674$ (rys. 1.6), a miejsce zerowe znajduje się w punkcie $x_0 = 3{,}3106$.



Rysunek 1.6. Wizualizacja funkcji oraz uzyskane rozwiązania

Tabela 1.7. Obliczenie miejsca zerowego oraz minimum funkcji w zadanym przedziale

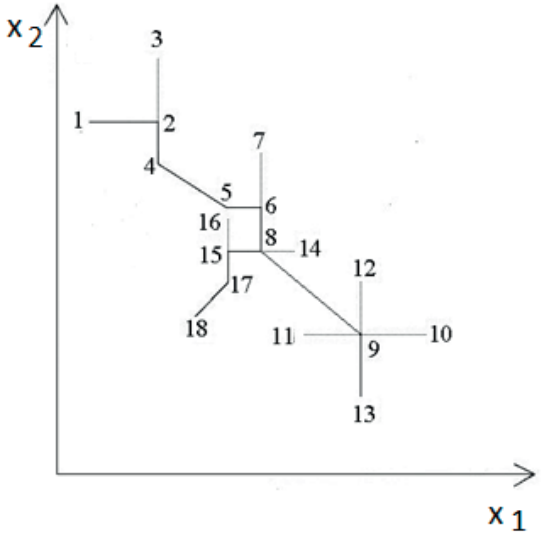
```
clear; close all; clc
x=-1:0.1:5*pi
% Definicja funkcji kryterialnej
Fc=@(x) x.*exp(-x)+sin(x).*exp(-0.1*x)
fig=figure(1)
set(fig,'defaultTextInterpreter','latex');
grid on; hold on
plot(x,Fc(x))
xlabel('x');
ylabel('$f(x)=x e^{-x}+sin(x)e^{-0.1x}$')
% Użycie funkcji fzero (1.11) – miejsce zerowe funkcji Fc
% w przedziale [3 4]
[x0, fc0, exit_flag, inform]=fzero(Fc,[3 4],optimset('Display','iter'))
plot(x0,fc0,'r*')
% Zastosowanie funkcji fminbnd (1.5) – minimum Fc w przedziale [pi, %
2pi]
[xmin, fcmin]=fminbnd(Fc,pi,2*pi,optimset('Display','iter'))
plot(xmin,fcmin,'b*')
legend('Fc','$x_{0}  ,  [\pi,  2\pi]$','$x^{min},  [3,4]$','interpreter',
'latex');
```

W wielowymiarowych metodach poszukiwań prostych można zastosować m.in. algorytmy [3]: Hooke'a i Jeevesa, Rosenbrocka oraz Neldera-Meada (który w pakiecie Matlab zaimplementowany jako funkcja fminsearch (1.7) [37]).

Algorytm Hooke'a-Jeevesa

Algorytm Hooke'a-Jeevsa (tab. 1.8) jest deterministyczną bezgradientową, iteracyjną metodą optymalizacji bez ograniczeń. Każda iterakcja składa się z dwóch etapów [61]:

1. etapu próbnego – służącego do zbadania lokalnych zmian funkcji celu $F: R^n \rightarrow R$ w otoczeniu pewnego punktu przez wykonywanie niewielkich kroków we wszystkich kierunkach ortogonalnej bazy;
2. etapu roboczego – polegającego na przejściu do kolejnego punktu (rozwiązania).



Rysunek 1.7. Przykładowe kroki realizacji algorytmu [62]

Przykładowa trajektoria – punktów podczas realizacji algorytmu: **Hooke'a-Jeevsa** kroki udane (zaznaczono linią ciągłą) i nie udane (narysowano linią przerywaną): 1–4 etap próbny, 4–5 etap roboczy, 5–8 etap próbny 8–9 etap roboczy, 9–13 etap próbny i cofnięcie do punktu 8 (rys. 1.8), 14–17 etap próbny 17–18 etap roboczy [62].

Tabela 1.8. Implementacja algorytmu Hooke'a-Jeevsa [61], [63] w środowisku Matlab

```
function [xb yb j]=fminHJ(nazwafunkcji,xb,K,Nmaxiter,e)
   N=length(xb) % liczba zmiennych decyzyjnych – wymiar zadania
 % start algorytmu
 yb=feval(nazwafunkcji,xb) % badany punkt –
                           % wartość funkcji w badanym punkcie
 dK=1;
 j=1;
 % pętla główna – liczba iteracji i założony błąd optymalizacji
 while((j<Nmaxiter)&&(dK>e))
    % Etap próbny – badnie funkcji w kierunkach ortogonalnych
    % (prostopadłych)
    xp=xb;
    yp=yb;
    for i=1:N                      % analiza wszystkich zmiennych
       xp(i)=xp(i)+K(i);           % kierunek "dodatni"
       yt=feval(nazwafunkcji,xp); % tymczasowe rozwiązanie
       if yt<yp                    % sukces w fazie próbnej
          yp=yt;
       else                        % brak pomyślnego kroku
          xp(i)=xp(i)-2*K(i);      % analiza w przeciwnym kierunku
          yt=feval(nazwafunkcji,xp); % tymczasowe rozwiązanie
          if yt<yp                 % sukces w fazie próbnej
             yp=yt;
          else
             xp(i)=xp(i)+K(i);     % krok nie został wykonany
          end
       end
    end
% Etap roboczy
% odpowiedź na sukces w fazie próbnej – zmiana x we właściwym kierunku
    if yp<yb
       xt=xb+1.4*(xp-xb);          % jakiś krok został wykonany
       yt=feval(nazwafunkcji,xt);
       if yt<yp
          K=1.2*K;                 % zwiększenie długości kroku
          xb=xt;
          yb=yp;
       else
          K=0.2*K;                 % zmniejszenie długości kroku
          xb=xp;
            yb=yp;
        end
     else
        K=0.2*K;                   % zmniejszenie długości kroku
     end
     dK=sqrt(K*K');
     j=j+1;
 end
end
```

Zastosowanie algorytmu Hooke'-Jeevesa (tab. 1.8) [6] w optymalizacji funkcji wyrażonej wzorem:

$$F_c = x_1^2 + x_2^2 - \frac{400}{100x_1^2 + x_2^2 + 1} - \frac{100}{100\,(x_1 - 2)^2 + (x_2 - 2)^2 + 1}, \tag{1.72}$$

przedstawiono w tabeli 1.9.

Można wspomnieć, że różne punkty startowe x0 (tab. 1.9) np. x0=[1 1] i x0=[3 3], prowadzą do różnych (dwóch) rozwiązań.

Tabela 1.9. Zastosowanie algorytmu Hooke'a-Jeevesa

```
clear; close all; clc
x0   =[1 1];        % punkt startowy optymalizacji
K    =[0.1 0.1];    % początkowy wektor kroków
Niter=200;          % maksymalna założona liczba iteracji
e    =10^(-6);      % błąd optymalizacji
% przykładowa funkcja
Fc=@(x1,x2)  x1.^2+x2.^2-  400./(100*(x1).^2+x2.^2+1)  -  100./(100*(x1-
2).^2+(x2-2).^2+1); % wzór (1.68)
[xopt fopt iter]= fminHJ (@(data)Fc(data(1),data(2)),x0,K,Niter,e);
fig=figure;
set(fig,'defaultTextInterpreter','latex');
hold on; grid on;
fsurf(Fc)
xlabel('$x_1$'); ylabel('$x_2$');
zlabel('Wartość   $F_c=x_1^2+x_2^2-   400/(100x_1^2+x_2^2+1)-100/(100(x1-
2)^2+(x2-2)^2+1)$');
plot3(xopt(1),xopt(2), fopt, 'Marker', '+','MarkerSize',20,'Color', 'r');
title('$F_c=x_1^2+x_2^2-     400/(100x_1^2+x_2^2+1)-100/(100(x1-2)^2+(x2-
2)^2+1)$', 'FontSize',14);
legend({'Fc','$(x_{1}^{min}, x_{2}^{min}, F_{c}^{min})$'},'interpreter',
'latex');
```

Algorytm Neldera-Meada

Metoda Neldera-Meada [64] (NM) (sympleksowa metoda spadku) jest to numeryczna deterministyczna metoda bezgradnientowa (bez korzystania z pochodnych funkcji) do wyznaczania ekstremum (typowo minimum) nieliniowych funkcji wielu zmiennych bez ograniczeń.

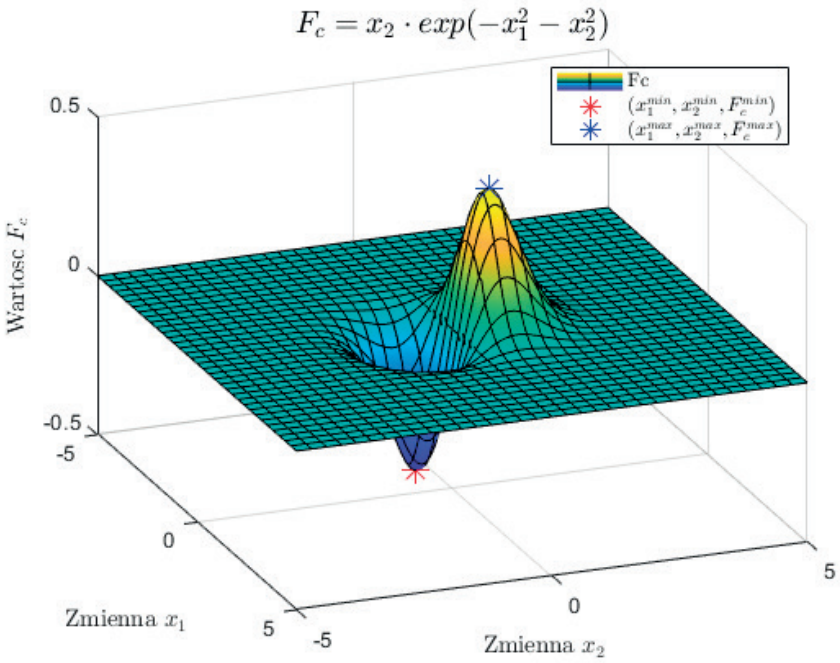
Funkcja ***fminsearch*** (tab. 1.1) dostępna w pakiecie Matlab wykorzystuje algorytm Neldera-Meada (NM) przedstawiony w artykule Lagarias et al. [65]. Dokładny opis procedury NM można uzyskać w pomocy (podręczniku) pakietu Matlab [66] oraz innych podręcznikach [6], [67].

W algorytmie Neldera-Meada (NM) realizowane są przekształcenia: **odbicia**, **ekspansji**, **zawężenia** oraz **redukcji**.

Zastosowanie minimalizacji oraz maksymalizacji funkcji danej wzorem:

$$F_c = x_2\, exp(-x_1^2 - x_2^2), \tag{1.73}$$

w przedziale x_1, $x_2 \in [-2, 2]$, z użyciem funkcji ***fminsearch*** (algorytm Neldera-Meada) przedstawiono w tabeli 1.10.

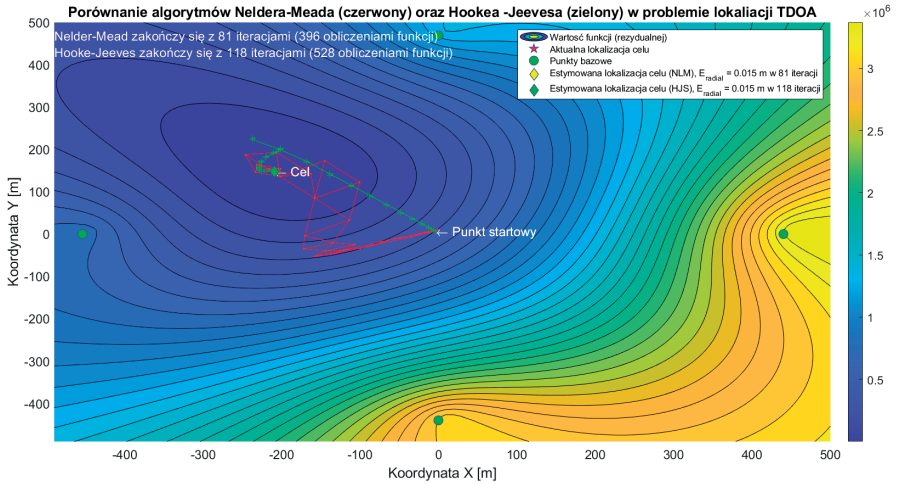


Rysunek 1.8. Wizualizacja funkcji (1.69) z rozwiązaniami optymalizacji

Tabela 1.10. Zastosowanie funkcji fminsearch (algorytm Neldera-Meada)

```
clear; close all; clc;
% optymalizowana funkcja
Fc = @(x1,x2) x2.*exp(-x1.^2-x2.^2);
fig=figure;
set(fig,'defaultTextInterpreter','latex');
hold on
grid on
zakres=[-1 1 -1 1]*5;
fsurf(Fc,zakres);
options = optimset('TolFun',1e-6,'TolX',1e-6, 'Display','iter');
x0=[0 0]; % punkt startowy optymalizacji
[xmin,Fcmin,exitflag, output] = fminsearch(@(data)Fc(data(1), data(2)),
x0,options);
[xmax,Fcmax] = fminsearch(@(data)-Fc(data(1),data(2)),x0,options);
plot3(xmin(1),xmin(2),Fcmin,'r*','MarkerSize',12);
plot3(xmax(1),xmax(2),-Fcmax,'b*','MarkerSize',12);
title(['$F_c=x_{2} \cdot exp(-x_1^2-x_2^2)$'], 'FontSize',14);
xlabel('Zmienna $x_1$');
ylabel('Zmienna $x_2$');
zlabel('Wartosc $F_c$');
legend({'Fc','$(x_{1}^{min}, x_{2}^{min}, F_{c}^{min})$','$(x_{1}^{max},
x_{2}^{max}, F_{c}^{max})$'},'interpreter','latex');
```

Przykład działania dwóch metod optymalizacji Hooke'a-Jeevesa oraz Neldera–Meada do rozwiazywania problemu nawigacji (lokalizacji) techniką *Time-Difference-Of-Arrival* (TDOA) (rys. 1.9) zaprezentowano w projekcie UCNLNav (*Underwater Communication and Navigation Labolatory*) [68]. Pomiar różnicy czasowej nadejścia sygnału (TDOA), czyli multilateracja, jest techniką lokalizacji wykorzystywaną w cywilnych oraz wojskowych zastosowaniach nadzoru dla dokładnego określenia położenia: samolotów, pojazdów lub nadajników stacjonarnych.



Rysunek 1.9. Przykładowe minimalizacja funkcji rezydualnej dla TDoA z użyciem dwóch algorytmów: Nerleda-Meada oraz Hookea-Jeevesa

Metody gradientowe

Metody gradientowe oprócz znajomości aktualnych wartości funkcji celu wykorzystują gradient lub wielkości z nim związane. Dla takich algorytmów funkcja celu musi być określona i różniczkowalna w całej przestrzeni $\mathbb{R}^n$.

Gradient funkcji celu f w punkcie $x = [x_1, x_2 \ldots x_n]^T$ jest określony równaniem:

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1} \frac{\partial f(x)}{\partial x_2} \ \ldots \ \frac{\partial f(x)}{\partial x_{\mathrm{n}}} \right]^T. \tag{1.74}$$

Zgodnie z przyjętą strategią optymalizacji, kolejne punkty prowadzące do minimum wyznacza się jako:

$$\mathrm{x}^{(i+1)} = \mathrm{x}^{(i)} + h^{(i)} \, d^{(i)}, \tag{1.75}$$

gdzie: $d^{(i)}$ ustalony kierunek optymalizacji w kroku i, $h^{(i)}$ długość kroku w i-tej iteracji.

Oprócz standardowych warunków stopu – wzory (1.1–1.2), w metodach gradientowych (szczególnie w metodzie ze stałym krokiem), można zastosować test stacjonarności rozwiązania wyrażony wzorem:

$$\|\nabla f(x)\|_2 < \varepsilon, \tag{1.76}$$

gdzie: $\varepsilon > 0$ założona dokładność rozwiązania.

Metody gradientowe w różny sposób ustalają wartości parametrów $d^{(i)}$ oraz $h^{(i)}$ (metody stałokrokowe oraz metody zmiennokrokowe).

Do najważniejszych metod gradientowych można zaliczyć:

- metodę **największego spadku**:

$$d^{(i)} = -\nabla f\left(\mathrm{x}^{(i)}\right), \tag{1.77}$$

$$g'\left(h^{(i)}\right) = \sum_{j=1}^{n} \frac{\partial f\left(\mathrm{x}^{(i)} + h^{(i)}\, d^{(i)}\right)}{\partial x_j} d_j^{(i)} = 0, \tag{1.78}$$

- metodę **gradientów sprzężonych**:

$$h^{(i)} = -\frac{\left(d^{(i)}\right)^T \nabla f\left(\mathrm{x}^{(i)}\right)}{(d^{(i)})^T A d^{(i)}}, \tag{1.79}$$

gdzie: A (1.63) macierz współczynników funkcji wyrażonej wzorem:

$$f(x) = \frac{1}{2} x^T A x + b^T x, \tag{1.80}$$

$$d^{(0)} = -\nabla f\left(\mathrm{x}^{(0)}\right), \tag{1.81}$$

$$d^{(i+1)} = -\nabla f\left(\mathrm{x}^{(i+1)}\right) + \beta^{(i+1)} d^{(i)}, \tag{1.82}$$

- metodę Fletchera-Reevesa:

$$\beta^{(i+1)} = \frac{\nabla f\left(x^{(i+1)}\right)^T \nabla f\left(x^{(i+1)}\right)}{\nabla f(x^{(i)})^T \nabla f(x^{(i)})} = \frac{\left(\left\|\nabla f\left(x^{(i+1)}\right)\right\|_2\right)^2}{(\|\nabla f(x^{(i)})\|_2)^2}, \tag{1.83}$$

- metodę Polaka-Ribière'a:

$$\beta^{(i+1)} = \frac{\nabla f\left(\mathrm{x}^{(i+1)}\right)^T \left(\nabla f\left(\mathrm{x}^{(i+1)}\right) - \nabla f\left(\mathrm{x}^{(i)}\right)\right)}{\nabla f(\mathrm{x}^{(i)})^T \nabla f(\mathrm{x}^{(i)})}, \tag{1.84}$$

- metodę Hestenesa-Stiefela:

$$\beta^{(i+1)} = \frac{\nabla f\left(\mathrm{x}^{(i+1)}\right)^T \left(\nabla f\left(\mathrm{x}^{(i+1)}\right) - \nabla f\left(\mathrm{x}^{(i)}\right)\right)}{(d^{(i)})^T \left(\nabla f(\mathrm{x}^{(i+1)}) - \nabla f(\mathrm{x}^{(i)})\right)}. \tag{1.85}$$

- metodę Newtona:

$$d^{(i)} = -H^{-1}\left(\mathrm{x}^{(i)}\right) \nabla f\left(\mathrm{x}^{(i)}\right), \tag{1.86}$$

gdzie: macierz $H(x)$ jest macierzą Hessego funkcji f w punkcie x,

$H^{-1}(x)$ jest macierzą odwrotną $H(x)$.

Wzór (1.86) wynika z rozwinięcia Taylora, przybliżeniu optymalizowanej funkcji $f(x)$ formą kwadratową (F_K), różniczkowaniu stronami tego rozwinięcia i przyrównaniu pochodnej do zera:

$$F_k(x, x_k) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2!}f''(x_k)(x - x_k)^2, \tag{1.87}$$

$$F_k'(x, x_k) = 0 \Leftrightarrow f'(x_k) + f''(x_k)(x - x_k) = 0, \tag{1.88}$$

$$x = x_k - \frac{f'(x_k)}{f''(x_k)} \Rightarrow \mathrm{x}^{(i+1)} = \mathrm{x}^{(i)} - H^{-1}\left(\mathrm{x}^{(i)}\right)\nabla f\left(\mathrm{x}^{(i)}\right). \tag{1.89}$$

Metoda Newtona działa poprawnie w sytuacji kiedy można obliczyć wartość $H^{-1}(x)$ w każdym z punktów $x^{(i)}$, czyli hesjan jest macierzą nieosobliwą. Dodatkowo w metodzie Newtona należy uwzględnić dwa dodatkowe warunki stopu: hesjan $H(x^{(i)})$ jest macierzą nieodwracalną oraz zerowanie się gradientu: $\nabla f\left(\mathrm{x}^{(i)}\right) = 0$.

- **Metody quasi-newtonowskie:**

$$d^{(i)} = -V^{(i+1)}\nabla f\left(\mathrm{x}^{(i)}\right), \tag{1.90}$$

$$V^{(0)} = I, \tag{1.91}$$

gdzie: I jest macierzą jednostkową,

– metoda Davidona-Fletchera-Powella (DFP):

$$V^{(i+1)} = V^{(i)} + A^{(i)} + B^{(i)}, \tag{1.92}$$

$$A^{(i)} = \frac{a^{(i)}(a^{(i)})^T}{(a^{(i)})^T s^{(i)}}, \tag{1.93}$$

$$B^{(i)} = \frac{V^{(i)}s^{(i)}(s^{(i)})^T V^{(i)}}{(s^{(i)})^T V^{(i)} s^{(i)}}, \tag{1.94}$$

$$s^{(i)} = \nabla f\left(\mathrm{x}^{(i)}\right) - \nabla f\left(\mathrm{x}^{(i-1)}\right), \tag{1.95}$$

$$a^{(i)} = \mathrm{x}^{(i)} - \mathrm{x}^{(i-1)}. \tag{1.96}$$

– metoda Broydena-Fletchera-Goldfarba-Shano (BFGS):

$$V^{(i+1)} = V^{(i)} + \left(1 + C^{(i)}\right)A^{(i)} + D^{(i)}, \tag{1.97}$$

$$A^{(i)} = \frac{a^{(i)}(a^{(i)})^T}{(a^{(i)})^T s^{(i)}}, \tag{1.98}$$

$$C^{(i)} = \frac{s^{(i)}V^{(i)}s^{(i)}}{(a^{(i)})^T s^{(i)}}, \tag{1.99}$$

$$D^{(i)} = \frac{a^{(i)}(s^{(i)})^T V^{(i)} + V^{(i)}s^{(i)}(a^{(i)})^T}{(a^{(i)})^T s^{(i)}}, \tag{1.100}$$

$$s^{(i)} = \nabla f\left(\mathrm{x}^{(i)}\right) - \nabla f\left(\mathrm{x}^{(i-1)}\right), \tag{1.101}$$

$$a^{(i)} = \mathrm{x}^{(i)} - \mathrm{x}^{(i-1)}. \tag{1.102}$$

Algorytm Lavenberga-Marquardata

Jest to algorytm iteracyjny, łączący w sobie cechy metody największego spadku i metody Gaussa-Newtona.

W przypadku ogólnym algorytmem Levenberga-Marquardta można poszukiwać rozwiązania zadania optymalizacji nieliniowej dla funkcji kryterialnej postaci:

$$f(x) = \frac{1}{2}\sum_{i=1}^{N} r_i^2\,(x), \tag{1.103}$$

gdzie $x \in R^n, N \geq n$.

Pochodne funkcji $f(x)$ można zapisać z użyciem macierzy Jacobiego funkcji r, zdefiniowanego jako:

$$\{J(x)\}_{ij} = \frac{\partial r_i}{\partial x_j}(x). \tag{1.104}$$

Gradient funkcji $f(x)$ można zapisać jako:

$$\nabla f(x) = \sum_{i=1}^{N} r_i(x)\nabla r_i(x) = J(x)^T r(x), \tag{1.105}$$

a macierz Hessego:

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{N} r_i(x)\,\nabla^2 r_i(x). \tag{1.106}$$

W przypadku gdy funkcje r_i można aproksymować funkcjami liniowymi w otoczeniu rozważanego punktu ($\nabla^2 r_i\,(x)$ jest bliskie zeru lub $r_i\,(x)$ jest małe), wówczas macierz Hessego przyjmuje postać:

$$\nabla^2 f(x) = J(x)^T J(x), \tag{1.107}$$

Stosując najprostszą minimalizację metoda największego spadku:

$$\mathrm{x}^{(i+1)} = \mathrm{x}^{(i)} - h^{(i)}\,\nabla f\left(\mathrm{x}^{(i)}\right), \tag{1.108}$$

Po rozwinięciu w szereg Taylora i przyjęciu przybliżenia kwadratowego funkcji $f(x)$ w otoczeniu analizowanego punktu, otrzymujemy wzór metody Gaussa-Newtona:

$$\mathrm{x}^{(i+1)} = \mathrm{x}^{(i)} - \left(\nabla^2 f\left(\mathrm{x}^{(i)}\right)\right)^{-1}\nabla f\left(\mathrm{x}^{(i)}\right). \tag{1.109}$$

Kenneth Levenberg zauważył, że obie metody: największego spadku i Gaussa-Newtona uzupełniają się i zaproponował modyfikację kroku jako:

$$\mathrm{x}^{(i+1)} = \mathrm{x}^{(i)} - \left(H\left(\mathrm{x}^{(i)}\right) + hI\right)^{-1}\nabla f\left(\mathrm{x}^{(i)}\right), \tag{1.110}$$

Levenberg zaproponował, żeby algorytm znajdowania minimum funkcji $f(x)$ (np. dopasowania danych pomiarowych, gdzie analizowana funkcja może być błędem średniokwadratowym MSE *mean square error*) wykorzystał

metodę gradientu z dala od minimum oraz metodę rozwinięcia w szereg Taylora w jego pobliżu.

Donald Marquardt zauważył, że w sytuacji dużych wartości h (macierz Hessego praktycznie nie jest wykorzystywana), można wykorzystać informację zawartą w drugiej pochodnej minimalizowanej funkcji, przez skalowanie każdego komponentu wektora gradientu w zależności od krzywizny w danym kierunku. Takie podejście jest użyteczne w minimalizacji źle uwarunkowanych zadań typu *error valley*.

Poprawka Marquardta modyfikuje krok metody według równania:

$$\mathrm{x}^{(i+1)} = \mathrm{x}^{(i)} - \left(H\left(\mathrm{x}^{(i)}\right) + h\, diag[H]\right)^{-1} \nabla f\left(\mathrm{x}^{(i)}\right). \tag{1.111}$$

Metody gradientowe – wybrane zastosowania w środowisku Matlab

W opcjach zastosowania funkcji ***fminunc*** (tab. 1.1) można użyć dwóch typów algorytmów: metody obszaru zaufania (*trust region methods*) oraz metody quasi-newtonowskie. Dla tych ostatnich zastosowano metody: BFGS, LBFGS *low-memory* (małe użycie pamięci) BFGS dla aproksymacji hesjanu oraz inne warianty/metody. Porównanie obliczeń z użyciem różnych wariantów algorytmu (tab. 1.11) (dostępnych opcji funkcji ***fminunc***) dla n=1000 wymiarowej funkcji Rosenbrocka wyrażonej wzorem:

$$F_c(x) = \sum_{i=1}^{n-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2], \ \ \forall x \ \in \mathbb{R}^n, \tag{1.112}$$

dla której punkt startowy (x_0) każdej zmiennej ustawiony został wartością (–2), przedstawiono w tabeli 1.11.

*Tabela 1.11. Zestawienie wyników obliczeń dla różnych wariantów **fminunc** [66]*

Algorytm	Czas obliczeń [s]	F_c(wartość funkcji))	Liczba iterakcji
BFGS_NoGrad	110,4400	5,0083 10^{-8}	7137
LBFGS_NoGrad	53,1430	2,4760 10^{-7}	4902
BFGS_Grad	35,4910	2,9865 10^{-8}	7105
LBFGS_Grad	1,2056	9,7505 10^{-8}	4907
Anallityczny (trust region)	7,0991	1,6710 10^{-10}	2301
fin–diff–grads	5,2170	1,1422 10^{-15}	1382
LSQ_NoJacob	94,7080	3,7969 10^{-25}	1664
LSQ_Jacob	6,5225	3,0056 10^{-25}	1664

Przykład 3 – zastosowanie metod gradientowych (funkcja ***fminunc***, obliczenia symboliczne, metoda Newtona) w poszukiwaniu ekstremum funkcji wyrażonej wzorem:

$$F_c(x) = x_1^3 + x_2^3 - 3x_1x_2\,. \tag{1.113}$$

Kod programu do rozwiązania zadania z przykładu 3 z zastosowaniem obliczenia symbolicznego z pakietu Matlab [70] przedstawiono w tabeli 1.12. Zastosowanie funkcji ***fminunc*** (quasi-newtonowska BFGS) do rozwiązania zadania optymalizacji z przykładu 3, przedstawiono w tabeli 1.13.

Tabela 1.12. Rozwiązanie symboliczne dla przykładu 3

```
clear; close all; clc;
syms x_1 x_2;
f= x_1^3+x_2^3-3*x_1*x_2; %analizowana funkcja (przykład 3)
pretty(f);
figure
fsurf(f,[-2,2]);
title(['f(x,y,z) = ' texlabel(f)]);
xlabel('x_1'), ylabel('x_2'), zlabel('f(x_1,x_2)');
gradf=gradient(f);
pretty(gradf);
extrem=solve(gradf==0);
x=double([extrem.x_1(1) extrem.x_2(1)]);
x=[x; double([extrem.x_1(2) extrem.x_2(2)])];
J=jacobian(gradf, [x_1, x_2]);
H=hessian(f,[x_1,x_2]);
H_1=double(subs(H, [x_1 x_2], [x(1,1), x(1,2)]));
H_2=double(subs(H, [x_1 x_2], [x(2,1), x(2,2)]));
detH_1=det(H_1),detH_2=det(H_2);
wynik=[];
for  i=1:size(x,1)
    wynik=[wynik;eval(subs(f, [x_1,x_2],[x(i,:)]))];
end
x
wynik
hold on
plot3(x(2,1),x(2,2), wynik(2),'b*','MarkerSize',12);
```

Analizowana funkcja posiada ekstremu lokalne, gdy gradient $\Delta f = 0$ oraz wyznacznik hesjanu jest dodatni. W przypadku hesjanu dodatnie określonego $|H| > 0$ (wszystkie minory główne hesianu są dodatnie – kryterium Sylvestera / wszystkie wartości własne hesjanu – wynikająca z rozwinięcia Taylora) – osiągane jest minimum lokalne, natomiast w przypadku hesjanu ujemnie określonego $|H| < 0$ funkcja posiada maksimum lokalne w analizowanym punkcie [70]. Jeśli część wartości własnych hesjanu jest dodatnia a część ujemna, to analizowany punkt jest punktem siodłowym.

*Tabela 1.13. Zastosowanie algorytmu Newtona oraz funkcji **fminunc** do rozwiązania zadania z przykładu 3*

```
clear; close all; clc;
%optymalizowana funkcja (przykład 3)
Fc=@(x1, x2) x1.^3 + x2.^3 - 3.*x1.*x2;
grad_f=@(x1, x2) [3*x1^2-3*x2; 3*x2^2-3*x1];
hess_f=@(x1, x2) [6*x1, -3; -3, 6*x2];
fig=figure;
set(fig,'defaultTextInterpreter','latex');
rozmiar=2.5
axes('Xlim', [-rozmiar rozmiar], 'Ylim', [-rozmiar rozmiar]);
hold on; grid on;
fcontour(Fc,'LineWidth', 1,'LineColor','k')
odstep = 0.2;
[X1, X2] = meshgrid(-rozmiar:odstep:rozmiar, -rozmiar:odstep:rozmiar);
x0= [2 2]  %punkt startowy optymalizacji
x=x0;
xlabel('x1', 'FontSize', 12);
ylabel('x2', 'FontSize', 12);
zlabel('Fc(x1, x2)', 'FontSize', 12);
view(30, 90);
[DX,DY] = gradient(Fc(X1,X2),odstep);
quiver(X1,X2,DX,DY,'r','LineWidth',2)
fs=fsurf(Fc)
alpha(fs,.4)
max_iter=10; %przyjęta maksymalna liczba iteracji
iter=0;
plot(x(1), x(2), 'ro', 'MarkerFaceColor', 'r');
a=[]; b=[];a=[a;x(1)]; b=[b;x(2)];
%Realizacja algorytmu Newtona
while true
 g=grad_f(x(1), x(2));
 H=hess_f(x(1), x(2));
 delta_x=-H\g
 x=x+delta_x'
 if norm(g) < 1e-6 || iter >= max_iter
    break;
 end
 iter=iter+1
 a=[a;x(1)]; b=[b;x(2)];
end
%Wywołanie funkcji fminunc
options=optimset('Display','final');
[xmin,Fcmin,exitflag, output,grad, hes]= fminunc(@(data)Fc(data(1),
data(2)), x0,options)
plot(a, b, 'rx', 'MarkerFaceColor', 'r','MarkerSize',10);
plot([x0(1), xmin(1)], [x0(2), xmin(2)], 'bo');
line(a,b);
title(['$F_c=x_{1}^3 + x_{2}^3 - 3x_{1} x_{2}$'], 'FontSize',14);
xlabel('Zmienna $x_1$'); ylabel('Zmienna $x_2$');
zlabel('Wartość $F_c$');
legend({'kontury     Fc','gradient','Fc','$(x0_{1},     x0_{2}$','kolejne
punkty','$(x_{1}^{min}, x_{2}^{min})$'},'interpreter','latex');
d=0.1;
text(xmin(1)+d,xmin(2)+d,strcat('\bf    (     ,     num2str(xmin(1)),',',
num2str(xmin(2)),')'));
```

Przykład optymalizacji **metodą Newtona** funkcji danej wzorem (1.113) dla przyjętego punktu startowego $x_0 = \begin{bmatrix}2\\2\end{bmatrix}$.

Gradient i hesjan dla funkcji z przykładu 3 (1.97) zostały obliczone jako:

$$\nabla f(x_1, x_2) = \begin{bmatrix}3(x_1^2 - x_2)\\3(x_2^2 - x_1)\end{bmatrix}, \tag{1.114}$$

oraz:

$$H(\boldsymbol{x}) = \begin{bmatrix}\dfrac{\partial^2 f(x)}{\partial x_1^2} & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1}\\ \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f(x)}{\partial x_2^2}\end{bmatrix} = \begin{bmatrix}6x_1 & -3\\-3 & 6x_2\end{bmatrix}. \tag{1.115}$$

W pierwszych krokach iteracji dla przyjętego (przykładowego) – stałego kroku algorytmu $h = 1$ i punktu startowego $x_0 = \begin{bmatrix}2\\2\end{bmatrix}$, w przestrzeni $\mathcal{R}^2$ otrzymano kolejne przybliżenia rozwiązania **przykładu 3**:

$$x_0 = x^{(1)} = (x_1, x_2)^{(1)} = \begin{bmatrix}2\\2\end{bmatrix}, \tag{1.116}$$

$$\nabla f(x_1, x_2)^{(1)} = \begin{bmatrix}3(x_1^2 - x_2)\\3(x_2^2 - x_1)\end{bmatrix} = \begin{bmatrix}6\\6\end{bmatrix}, \tag{1.117}$$

$$H\left(\boldsymbol{x}^{(1)}\right) = \begin{bmatrix}12 & -3\\-3 & 12\end{bmatrix}, \tag{1.118}$$

$$\boldsymbol{d}^{(1)} = -H^{-1}\left(\boldsymbol{x}^{(1)}\right)\nabla f\left(\boldsymbol{x}^{(1)}\right) = -\begin{bmatrix}12 & -3\\-3 & 12\end{bmatrix}^{-1} \cdot \begin{bmatrix}6\\6\end{bmatrix} = -\begin{bmatrix}\frac{2}{3}\\\frac{2}{3}\end{bmatrix}, \tag{1.119}$$

$$x^{(2)} = x^{(1)} + h\boldsymbol{d}^{(1)} = \begin{bmatrix}2\\2\end{bmatrix} - 1 \cdot \begin{bmatrix}\frac{2}{3}\\\frac{2}{3}\end{bmatrix} = \begin{bmatrix}1\frac{1}{3}\\1\frac{1}{3}\end{bmatrix}, \tag{1.120}$$

$$\nabla f(x_1, x_2)^{(2)} = \begin{bmatrix}3(x_1^2 - x_2)\\3(x_2^2 - x_1)\end{bmatrix} = \begin{bmatrix}1\frac{1}{3}\\1\frac{1}{3}\end{bmatrix}, \tag{1.121}$$

$$H\left(\boldsymbol{x}^{(2)}\right) = \begin{bmatrix}8 & -3\\-3 & 8\end{bmatrix}, \tag{1.122}$$

$$\boldsymbol{d}^{(2)} = -H^{-1}\left(\boldsymbol{x}^{(2)}\right)\nabla f\left(\boldsymbol{x}^{(2)}\right) = -\begin{bmatrix}8 & -3\\-3 & 8\end{bmatrix}^{-1} \cdot \begin{bmatrix}1\frac{1}{3}\\1\frac{1}{3}\end{bmatrix} = \tag{1.123}$$

$$= \begin{bmatrix}-0{,}26(6)\\-0{,}26(6)\end{bmatrix},$$

$$x^{(3)} = x^{(2)} + h\boldsymbol{d}^{(2)} \approx \begin{bmatrix} 1\frac{1}{3} \\ 1\frac{1}{3} \end{bmatrix} - 1 \cdot \begin{bmatrix} 0{,}26(6) \\ 0{,}26(6) \end{bmatrix} \approx \begin{bmatrix} 1{,}06(6) \\ 1{,}06(6) \end{bmatrix}. \tag{1.124}$$

Kolejne kroki (iteracje) algorytmu prowadzą do obliczenia współrzędnych:

$$x^{(4)} \approx \begin{bmatrix} 1{,}003921568627451 \\ 1{,}003921568627451 \end{bmatrix}, \quad x^{(5)} \approx \begin{bmatrix} 1{,}000015259021897 \\ 1{,}000015259021897 \end{bmatrix}, \quad x^{(5)} \approx \begin{bmatrix} 1{,}000000000232831 \\ 1{,}000000000232831 \end{bmatrix},$$

gdzie rozwiązaniem dokładnym analizowanego **przykładu 3** jest punkt $x_{min} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

1.4.4. Algorytmy oraz algorytmy metaheurystyczne (MAs/MH/MHSs/MHS) Meta-heuristic search algorithms

W literaturze tematu zaprezentowano oraz opisano dużą liczbę heurystycznych metod optymalizacyjnych, dla których inspiracją była obserwacja zjawisk: przyrodniczych, fizycznych a nawet socjologicznych. Niektóre z nich są mało popularne (rzadko stosowane) lub wtórne do już istniejących, będące ich modyfikacjami, usprawnieniami albo połączeniem/kombinacją innych algorytmów.

Zgodnie z teorią NFL (*No-Free Lunch*), średnia dokładność dowolnego algorytmu optymalizacyjnego jest taka sama jak błądzenia losowego. Dlatego nie istnieje uniwersalny algorytm dający najlepsze rozwiązania dla różnych problemów optymalizacyjnych [11], [71].

Dwa dowolne algorytmy (a_1,a_2) są równoważne, jeśli dla wszystkich problemów (f), w kroku iteracji (m):

$$\sum_f P\left(d_m^y | f, m, a_1\right) = \sum_f P\left(d_m^y | f, m, a_2\right), \tag{1.125}$$

gdzie: d_m^y – oznacza uporządkowany zbiór o rozmiarze m, wartości kosztów y powiązanych z wartościami wejściowymi $x \in X$, $f: X \rightarrow Y$ jest optymalizowaną funkcją (celu), $P\left(d_m^y | f, m, a\right)$ prawdopodobieństwo warunkowe uzyskania danej sekwencji wartości kosztu z algorytmu (a), uruchomionych (m) razy na funkcji (f) [71].

Pewien algorytm może być jednak lepszy od innych dla określonej, wyspecyfikowanej klasie problemów. Z uwagi na trudności w procesie

poszukiwania rozwiązania optymalnego (zbieżność lub wysokie nakłady czasów obliczeń), pożądana jest znajomość więcej niż jednej metody optymalizacji [6].

Na podstawie przeglądu literatury można wyróżnić algorytmy/metody optymalizacji [72], [73], [74], [75], [76], [77] takie jak:

1. Sztuczne sieci neuronowe ANN (*Artificial Neural Network*) [75], [76];
2. Systemy rozmyte FS (*Fuzzy Systems*) [77], [78];

Algorytmy ewolucyjne EAs/EBs (*evolutionary algorithms/ evolution-based algorithms/ evolution-inspired algorithms*), inspirowane biologicznie (*bio-inspired not SI-based*)

3. Algorytm jSO, zmodyfikowany algorytm jSO (MjSO) – zaawansowana wersja algorytmy DE (*Differential Evolution*);
4. Programowanie ewolucyjne EP (*Evolutionary Programming*) [86], CEP (*classical* EP), FEP (*Fast EP*), CEP (*Conventional EP*), LEP (*Lévy EP*);
5. Algorytmy ewolucyjne EA (*Evolutionary Algorithms*) [4], [89], [90], [91] oraz modyfikacje: IEA Islands EA [92], SCE–UA *(Shuffled Complex Evolution)* [93], CEA (*Co-evolving algorithm*) [94], QEA (*Quantum Inspired EA*), DEA (*Differential Evolutionary Algorithm*);
6. Obliczenia ewolucyjne EC (*Evolutionary Computation*), NCCO (*Neighborhood-based Consensus*) [95];
7. Algorytm ewolucyjny inspirowany ekologią ECO (*Eco Inspired Evolutionary Algorithm*);
8. Programowanie genetyczne GP (*Genetic Programming*) [96], [97], [98], binarne programowanie genetyczne BGP (*Binary Genetic Programming*), rozwojowe programowanie genetyczne DGP (*Developmental Genetic Programming*);
9. Losowe sieci ewolucyjne GE–RWN (*Genetic Evolutionary Random Weight Networks*) [99];
10. Optymalizacja inspirowana naturą BIO (*Bio-Inspired Optimisation*) [100];
11. Uczenie przyrostowe oparte na prawdopodobieństwie PBIL (*Probability-based Incremental Learning*) [102];
12. Ewolucyjny algorytm membranowy EMA (*Evolutionary Membrane Algorithm*) [104];
13. Algorytmy szacowania dystrybucji EDA (*Estimation of Distribution Algorithms*) nazywane także PMBGAs (*Probabilistic Model-Building Genetic Algorithms*) [105];
14. Wyszukiwanie nowości NS (*Novelty Search*);

15. Rojowy sieciowy algorytm ewolucyjny NEA (*Swarm Networked Evolutionary Algorithm)*;
16. Algorytm samolubnego genu SGA (*Selfish Gene Algorithm*);
17. Ewolucja gramatyczna GE (*Grammatical Evolution*);
18. Gradientowy algorytm ewolucyjny GEA (*Gradient Evolution Algorithm*);
19. Sieciowy algorytm ewolucyjny NEA (*Networked Evolutionary Algorithm*) [107];
20. Ewolucyjna transformacja quasi-afiniczna QUATRE (*QUasi-Affine TRansformation Evolutionary*);
21. Algorytm agregacji naturalnej NAA (*Natural Aggregation Algorithm*);
22. Optymalizator oparty na odejmowaniu średniej SABO (*Subtraction-Average-Based Optimizer*);

Algorytmy fizyczno-matematyczne (*Physics-Mathematics-based algorithms* PMA) Algorytmy lokalnego szukania (*direct/local search*) – nieheurystyczne

23. Przeszukiwanie z tabu TS (*Taboo Search*);
24. Losowy marsz RW (*Random Walk*);
25. Przeszukiwanie wspinaczkowe HC/βHC (*β-Hill Climbing*) [108];
26. Iteracyjne wyszukiwanie lokalne ILS (*Iterated Local Search*) [109];
27. Przeszukiwanie ze zmiennym sąsiedztwem VNS (*Variable Neighborhood Search*) [110], [111];
28. Zachłanna losowa procedura wyszukiwania adaptacyjnego GRASP (*Greedy Randomized Adaptive Search Procedure*) [112];
29. Kierowane wyszukiwanie lokalne GLS (*Guided Local Search*) [113];
30. Szybkie wyszukiwanie lokalne FLS (*Fast Local Search*) [113];
31. Wyszukiwanie wzorca PS (*Pattern Search*) [114];
32. Wyszukiwanie losowe RS (*Random Search*) [115] oraz modyfikacje [116]: OSSRS (*Optimum Step Size Random Search*), ORSSRS (*Optimized Relative Step Size Random Search*), ASSRS (*Adaptive Step Size Random Search*);
33. Wielokierunkowe wyszukiwanie heurystyczne (*Multidirectional Heuristic Search*) [117];
34. Rozproszonego przeszukiwania SS (*Scatter Search*) [118];
35. Przełączanie ścieżek PR (*Path Relinking*);
36. Chaotyczne wyszukiwanie lokalne CLS (*Chaotic Local Search*);

Algorytmy matematyczne MBs (*mathematics-based*)

37. Optymalizator średniej geometrycznej GMO/GeMO (*Geometric Mean Optimizer*) [119];

38. Arytmetyczny algorytm optymalizacji AOA/AROA/ArOA (*Arithmetic Optimisation Algorithm*) [120], CSOAOA (hybrydowy z *Criss-Cross Strategy*);
39. Dynamiczny arytmetyczny algorytm optymalizacji DAOA (*Dynamic Arithmetic Optimisation Algorithm*);
40. Chaotyczny arytmetyczny algorytm optymalizacji CAOA (*Chaotic Arithmetic Optimisation Algorithm*);
41. Uogólniona aproksymacja wypukła GCA (*Generalized Convex Approximation*) [121];
42. Średnia ważona pozycji wektorów INFO (*weIghted meaN oF vectOrs*) [122];
43. Modyfikacje algorytmu BFGS: *chaos*–BFGS, Monte Carlo-BFGS [123];
44. Wyszukiwanie liniowe z ponownym startem LSRS (*Line Search ReStart*) [124];
45. Optymalizator gradientowy GBO (*Gradient-based Optimizer*), algorytm zmodyfikowany MGBO (*Modified Gradient-based Optimizer*);
46. Optymalizator Rungego-Kutty RUN/RUNge (*Runge Kutta Optimizer*) [125];
47. Chaotyczne wyszukiwanie lokalne z użyciem złotego podziału CGRGL (*Chaotic Golden Ratio Guided Local search*);
48. Algorytm optymalizacji wyszukiwania w szkółce rybackiej FSS (*Fish School Search Optimisation Algorithm*);
49. Algorytm wyszukiwania kołowego CSA (*Circle Search Algorithm*);
50. Optymalizator rozkładu wykładniczego EDO (*Exponential Distribution Optimizer*);
51. Optymalizacja interpolacji kwadratowej QIO (*Quadratic Interpolation Optimisation*);
52. Rozkład lotów Lévy'ego LFD (*Lévy Flight Distribution*) [126];
53. Algorytm sinus-cosinus SCA (*Sine-Cosine Algorithm*) [127] oraz ulepszony ISCA (*Improved Sine-Cosine Algorithm*) [128]; hybrydowy SCA-ABC; kwantowy QSCA (*Quantum-based* SCA *Sine-Cosine Algorithm*), MTV-SCA (*Multi Trial Vector*);
54. Algorytm heurystyczny DG-Alg (*Dahiya-Garg Heuristic Algorithm*) [129];
55. Bezmodelowy rozmyty algorytm sterowania adaptacyjnego Takagi-Sugeno CFDL–PDTSFA (*Model-Free Adaptive Control Takagi-Sugeno Fuzzy Algorithm*);

56. Uogólniony optymalizator rozkładu normalnego GNDO (*Generalized Normal Distribution Optimizer*);
57. Algorytm Rao RAs (*Rao Algorithms*);
58. Ewolucyjny algorytm Rao ERA (*Evolutionary Rao Algorithm*);
59. Optymalizator zależny od dopasowania FDO (*Fitness Dependent Optimizer*) [130], [131], poprawiony IFDO (*Improvement Fitness Dependent Optimizer*) [132], zmodyfikowany MFDO [133];
60. Inteligentna metaheurystyka czterech wektorów FVIM (*Four Vector Intelligent Metaheuristic*);
61. Trójfazowa technika unikania optymalizacji lokalnej (wyszukiwanie-ucieczka-synchronizacja) SES (*Search-Escape-Synchronize*);
62. Algorytmem wyszukiwania stycznego TSA (*Tangent Search Algorithm*);
63. Algorytm teorii logicznej IbI ILA (*Incomprehensible but Intelligible-in--time IbI Logics Algorithm*);

Algorytmy bazujące na zjawiskach fizycznych PBs/PBA (*physical–based/physic–inspired*)

64. Algorytm wielkiego wybuchu – wielkiego kolapsu BBBC/BB-BC/BBC (*Big-Bang Big-Crunch, Big-Bang Crunch*) [134] oraz ulepszenie HBB-BC (*Hybrid Big-Bang Big-Crunch*) [135];
65. Algorytm grawitacyjny GSA (*Gravitational Search Algorithm*) [136], HGSA (*hierarchical* GSA), DGSA (*distributed framework* GSA);
66. Wyszukiwanie oparte na grawitacji GBS (*Gravitational-based Search*);
67. Gradientowe wyszukiwanie grawitacyjne GGS (*Gradient-based Gravitational Search*);
68. Algorytm pola grawitacyjnego GFA (*Gravitation Field Algorithm*);
69. Optymalizacja oddziaływań grawitacyjnych GIO (*Gravitational Interactions Optimisation*);
70. Optymaliztor gromady galaktyk GSO (*Galactic Swarm Optimisation*);
71. Algorytm czarnej dziury BH/BHA/BHO (*Black Hole Algorithm/Optimisation*) [137], chaotyczny binarny algorytm czarnej dziury CBBHA (*Chaotic Binary Black Hole Algorithm*);
72. Algorytm galaktyczny GBA (*Galaxy-based Algorithm*) [138], GbSA/GBSA (*Galaxy-based Search Algorithm*) [139], GbSA-PCA (*Principal Components Analysis*), GBSO (*Galaxy-based Search Optimisation*), GbSA-MLT (*MultiLevel Tresholding*);
73. Ważona superpozycja przyciągania WSA (*Weighted Superposition Attraction*);

74. Optymalizacja zakrzywionej przestrzeni CSO (*Curved Space Optimisation*);
75. Optymalizator wieloświatowy MVO (*Multi-Verse Optimizer*) [140];
76. Chaotyczny algorytm optymalizacji wielowariantowej CMVO (*Chaotic Multi-Variate Optimisation Algorithm*);
77. Optymalizator supernowej SO (*Supernova Optimizer*);
78. Algorytm balistyczny AIG (*Algorithm of Innovative Gunner*) [69];
79. Wyszukiwanie naładowanego układu (prawo Coulomba) CSS (*Charged System/Systems Search*) [143];
80. Optymalizacja cząstek naładowanych elektrycznie ECPO/MECPO (*Eelectric Charged Particles Optimisation*) [144];
81. Optymalizacja elektromagnetyczna EMO (*Electromagnetism/ Electro-magnetism Optimisation*) [145],
82. Algorytm optymalizacji promienia RO (*Ray Optimisation Algorithms*) [146];
83. Optymalizacja zderzających się ciał CBO (*Colliding Bodies Optimisation*) [148];
84. Algorytm wyszukiwania błyskawic LSA (*Lightning Search Algorithm*) [149];
85. Optymalizacja parowania wody WEO (*Water Evaporation Optimisation*) [150];
86. Optymalizacja procedury dołączania błyskawic LAPO (*Lightning Attachment Procedure Optimisation*);
87. Algorytm wyszukiwania elektrycznego ES (*Electro-Search Algorithm*);
88. Znajdź-napraw-zakończ-wykorzystaj-analizuj F3EA (*Find-Fix-Finish -Exploit-Analyze*);
89. Optymalizacja siły centralnej CFO (*Central Force Optimisation*) [152];
90. Optymalizacja pola elektromagnetycznego EFO (*Electromagnetic Field Optimisation*) [153];
91. Optymalizacja reakcji nuklearnych NRO (*Nuclear Reaction Optimisation*) [154];
92. Algorytm dynamicznej optymalizacji różnicowej z wyżarzaniem DDAO (*Dynamic Differential Annealed Optimization Algorithm*);
93. Algorytm sztucznego pola elektrycznego AEFA (*Artificial Electric Field Algorithm*), zmodyfikowany lotami Lévy'ego mAEFA [155];
94. Mechanizm podobny do elektromagnetyzmu EM (*Electromagnetism-like Mechanism*) [156];

95. Algorytm optymalizacji stanu materii SMS (*States of Matter Search/ Optimisation Algorithm*) [157];
96. Algorytm optymalizacji spiralnej SO/SOA/Spiral/SPOA/SPOASelf (*Spiral Optimisation Algorithm*);
97. Algorytm optymalizacji dynamiki spiralnej SDIO/SO (*Spiral Dynamics Inspired Optimisation*) [158], [159];
98. Optymalizacja inspirowana układami optycznymi OIO (*Optics-Inspired Optimisation*) [160], ROIO (*Rotation OIO*), COIO (*Convex Combination OIO*);
99. Optymalizacja poszukiwania atomu ASO (*Atom Search Optimisation*) [161];
100. Algorytm kolizji cząstek PCA (*Particle Collision Algorithm*) [162]
101. Algorytm wyszukiwania wirów VSA/VS (*Vortex Search Algorithm*) [116];
102. Optymalizacja kinematyki windy EKO (*Elevator Kinematics Optimisation*) [163];
103. Algorytm grawitacji kosmicznej SGA/SGO (*Space Gravitational Algorithm/Optimisation*) [164];
104. Algorytm wielkiego kolapsu BCA (*Big Crunch Algorithm*);
105. Magnetyczny algorytm optymalizacji MOA (*Magnetic Optimisation Algorithm*);
106. Optymalizacja histerezy HO (*Hysteretic Optimisation)*;
107. Optymalizacja promienia światła LRO (*Light Ray Optimisation*);
108. Wyszukiwanie transferu ciepła HTS (*Heat Transfer Search*);
109. Algorytm optymalizacji transferu ciepła HTOA (*Heat Transfer Optimisation Algorithm*);
110. Algorytm optymalizacji inspirowany bilardem BOA (*Billiards-Inspired Optimisation Algorithm*);
111. Optymalizator równowagi EO/EOA (*Equilibrium Optimizer/ Optimisation Algorithm*);
112. Algorytm figur Lichtenberga (wzory w formie rozgałęzionego drzewa) LA (*Lichtenberg Algorithm*);
113. Optymalizator doliny energetycznej (stabilność oraz różne tryby rozpadu cząstek) EVO (*Energy Valley Optimizer*);
114. Algorytm optymalizacji Archimedesa AOA (*Archimedes Optimisation Algorithm*), hybrydowy mutualizmowy algorytm MAOA (*Mutualism Archimedes Optimisation Algorithm*);

115. Algorytm metaheurystyczny Newtona NMA (*Newton Metaheuristic Algorithm*);
116. Optymalizator budowy piramid w Gizie GPC (*Giza Pyramids Construction–based Optimizer*);
117. Wyszukiwanie orbitali atomowych AOS (*Atomic Orbital Search*) [171];
118. Optymalizacja widma światła LSO (*Light Spectrum Optimisation*);
119. Algorytm teorii strun STA (*String Theory Algorithm*);
120. Optymalizacja według prawa Ficka (dyfuzja) FLA (*Fick's Law Optimisation*);
121. Algorytm mikroskopu optycznego OMA (*Optical Microscope Algorithm*);
122. Algorytm optymalizacji Keplera KOA/KOT (*Kepler Optimisation Algorithm/Technique*);
123. Optymalizator eksperymentu Younga z podwójną szczeliną YDSE (*Young's Double-Slit Experiment Optimizer*);
124. Optymalizacja cząsteczek homonuklearnych HMO (*Homonuclear Molecules Optimisation*);
125. Algorytm kierunku przepływu FDA (*Flow Direction Algorithm*);
126. Algorytm optymalizacji opadów ROA (*Rain Optimisation Algorithm*) [174];
127. Algorytm kropel wody WDA (*Water Drops Algorithm*);
128. Algorytm wyszukiwania sprężynowego SSA (*Spring Search Algorithm*);
129. Algorytm struktury krystalicznej CryStAl (*Crystal Structure Algorithm*);
130. Optymalizacja algorytmem określenia promienia Ziemi Al–Bruiniego BER (*Al-Biruni Earth Radius*);
131. Przeszukiwanie szczególnej teorii względności SRS (*Special Relativity Search*);
132. Algorytm przeszukiwania ogólnejj teorii względności GRSA (*General Relativity Search Algorithm*);
133. Algorytm wyszukiwania według pędu MSA (*Momentum Search Algorithm*);
134. Bazujący na prawie Coulomba algorytm świetlika CFA (*Coulomb Firefly Algorithm*);
135. Algorytm reżimu przepływu FRA (*Flow Regime Algorithm*);
136. Samonapędzające się cząsteczki SPP (*Self-driven/Self-propelled Particles/Self-Driven Particles*) [175];
137. Stochastyczne przeszukiwanie sieci (dyfuzyjne/różnicowe) SS (*Stochastic Searching Networks/Stochastic Difusion Search*) [176];

138. Wyszukiwanie tranzytu TrS/TS (*Transit Search*) [177];
139. Algorytm optymalizacji sztucznej fizyki APO (*Artificial Physics Optimization Algorithm*);
140. Optymalizacja inspirowana sonarem SIO (*Sonar Inspired Optimization*);
141. Optymalizacja generowania plazmy PGO (*Plasma Generation Optimization*);
142. Optymalizacja pocisków PRO (*Projectiles Optimization*);
143. Algorytm Układu Słoneczego SSA (*Solar System Algorithm*);
144. Algorytm optymalizacji bazujący na komunikacji COA (*Communication-based Optimization Algorithm*);
145. Algorytm optymalizacji granularności GBO/GrBO (*Granular-ball Optimisation*);
146. Algorytm systemów cząstek wibrujących VPO (*Vibrating Particle Systems Algorithm*);
147. Wyszukiwanie w całym sąsiedztwie ACS/ANS (*Across Neighborhood Search*);

Różne

148. Model chmur w atmosferze ACM/ACMO (*Atmosphere Clouds Model*);
149. Sztuczny algorytm wyszukiwania kooperacyjnego ACS (*Artificial Cooperative Search Algorithm*);
150. Systemy barowe BS (*Bar Systems*);
151. Optymalizacja wyszukiwania wstecznego BSA/BSO (*Backtracking Search Optimisation Algorithm*);
152. Algorytm ewolucji różnicowej oparty na modelu chmury CMBDE (*Cloud Model-Based Differential Evolution Algorithm*);
153. Algorytm optymalizacji chaosu COA (*Chaos Optimisation Algorithm*);
154. Algorytm wyszukiwania różnicowego DS/DSA (*Differential Search Algorithm*);
155. Algorytm rynku walutowego EMA (*Exchange Market Algorithm*);
156. Optymalizacja ekstremalna EO (*Extremal Optimisation*);
157. Algorytm fajerwerków FWA/FA (*Fireworks Algorithm*) [198], FAO (*Fireworks Algorithm Optimisation*), binarnyBFWA (*Binary Fireworks Algorithm*);
158. Metoda eksplozji granatu GEM (*Grenade Explosion Method*) [142];
159. Algorytm złotego sinusa GSA/Gold-SA (*Golden Sine Algorithm*);
160. Optymalizacja pracy serca HO (*Heart Optimization*);

161. Algorytm przeszukiwania wnętrza ISA (*Interior Search Algorithm*) [217], H-ISA (*Interior Search Algorithm* z *hill climbing*);
162. Programowanie Kaizen (zmiany pracy na lepsze) KP (*Kaizen Programming*);
163. Algorytm membranowy MA (*Membrane Algorithm/Algorithms*);
164. Algorytm wybuchu miny MB/MBA (*Mine Blast Algorithm*) [141];
165. Algorytm komunikacji neuronów NCA (*Neuronal Communication Algorithm*);
166. Algorytm wyszukiwania pereł PHA (*Pearl Hunting Algorithm*);
167. Wyszukiwanie przejeżdżających pojazdów PVS (*Passing Vehicle Search*);
168. Algorytm sztucznej kropli deszczu ARA/RDA (*Artificial Raindrop Algorithm*) [173];
169. Algorytmy naukowe SA (*Scientifics Algoritmhs*);
170. Optymalizacja inżynierii społecznej SEO (*Social Engineering Optimizer/Optimization*) [200];
171. Stochastyczne wyszukiwanie fraktalne SFS (*Stochastic Fractal Search*), ISFS (*Improved Stochastic Fractal Search Algorithm*) [106];
172. Algorytm wyszukiwania grupy SGA (*Search Group Algorithm*);
173. Prosta optymalizacja SO (*Simple Optimization*);
174. Algorytm optymalizacji teorii „małego świata" SWOA/SWO (*Small–World Optimisation Algorithm*), BSWA (*Binary-coding Small World Algorithm*);
175. Algorytm Wielkiego Potopu TGD (The Great Deluge Algorithm);
176. Optymalizacja oparta na wietrze WDO (*Wind Driven Optimization*);
177. Optymalizacja pary Yin-Yang YYPO/YYOP (*Yin-Yang-Pair Optimisation*) [218]; **Metaheurystyka inspirowana naturą i biologią w ramach kategorii ruchu wektora różnicowego, w której na wektor różnicowy wpływa cała populacja**
178. Algorytm optymalizacji sztucznej roślinności APO (*Artificial Plants Optimization Algorithm*);
179. Chaotyczny algorytm ważki CDA (*Chaotic Dragonfly Algorithm*);
180. Algorytm klastrowania grawitacyjnego GCA (*Gravitational Clustering Algorithm*);

Metaheurystyka inspirowana naturą i biologią w kategorii ruchu wektora różnicowego, w której na wektor różnicowy wpływają reprezentatywne rozwiązania

181. Zachowanie zwierząt na polowaniu ABH (*Animal Behavior Hunting*);

182. Algorytm sztucznego oddziału przeszukującego ASSA (*Artificial Searching Swarm Algorithm*) [214];
183. Algorytm sztucznego plemienia ATA (*Artificial Tribe Algorithm*);
184. Optymalizacja chemotaksji bakterii BCO (*Bacterial Chemotaxis Optimization*);
185. Inteligencja nietoperzy BI (*Bat Intelligence*);
186. Algorytm inspirowany nietoperzami BIA (*Bat Inspired Algorithm*);
187. Algorytm migracji biologicznej BMA (*Biology Migration Algorithm*);
188. Algorytm ślepego golca piaskowego BNMR (*Blind, Naked Mole-Rats Algorithm*);
189. Optymalizacja rojem pszczół BSO (*Bee Swarm Optimization*);
190. Algorytm optymalizacji rojem bioluminizującym BSO (*Bioluminiscent Swarm Optimization Algorithm*);
191. Binarny algorytm optymalizacji wielorybów BWOA (*Binary Whale Optimization Algorithm*);
192. Zbiorowe zachowanie zwierząt CAB (*Collective Animal Behavior*);
193. Zachowanie podczas podróży na wielbłądzie COA (*Camel Travelling Behavior*);
194. Stado według przywódcy FL (*Flock by Leader*);
195. Algorytm oddziaływań grawitacyjnych GIA (*Gravitational Interactions Algorithm*);
196. Algorytm optymalizacji przeszukiwania pingwinów PSOA (*Penguins Search Optimization Algorithm*);
197. Optymalizacja ruchu promieniowego RMO (*Radial Movement Optimisation*);
198. Algorytm ewolucji roju powierzchniowo-złożonego SSSE (*Surface--Simplex Swarm Evolution Algorithm*);
199. Algorytm watahy wilków WCA (*Wolf Colony Algorithm*);
200. Wyszukiwanie wilczego stada/watahy WPS (*Wolf Pack Search*);
201. Optymalizacja przetrwania zombie ZSO (*Zombie Survival Optimisation*);

Metaheurystyka inspirowana naturą i biologią w ramach kategorii ruchu wektora różnicowego, w której na wektor różnicowy wpływają subpopulacje

202. Hierarchiczny model roju HSM (*Hierarchical Swarm Model*);
203. Optymalizacja robaka WO (*Worm Optimization*);

Metaheurystyka inspirowana naturą i biologią w kategorii ruchu wektora różnicowego, w której wektor różnicowy jest pod wpływem sąsiedztwa

204. Biomimikra żerujących społecznych bakterii dla rozproszonej optymalizacji BFOA (*Biomimicry Of Social Foraging Bacteria for Distributed Optimization*);
205. Lokalne wyszukiwanie z emulacją grawitacji GELS (*Gravitational Emulation Local Search*);

Metaheurystyka inspirowana naturą i biologią w kategorii tworzenie rozwiązań – łączenie

206. Algorytm sztucznego ula ACAABA (*Artificial Beehive Algorithm*);
207. Algorytm optymalizacji energii kryształu CEO (*Crystal Energy Optimization Algorithm*);
208. Algorytm selekcji klonów CSA/CLONALG (*Clonal Selection Algorithm*);
209. Algorytm elementów harmonii HEA (*Harmony Elements Algorithm*);
210. Histereza dla optymalizacji HO (*Hysteresis for Optimization*);
211. Naturalny algorytm agregacji hipersześcianem HYNAA (*Hypercube Natural Aggregation Algorithm*);
212. Algorytm lwa LA (*Lion Algorithm*);
213. Metoda kompozycji muzycznej MMC (*Method of Musical Composition*);
214. Wyszukiwanie melodii MS (*Melody Search*);
215. Algorytm fotosyntezy PA (*Photosynthetic Algorithm*);
216. Algorytm muzyki popularnej PopMusic (*Pop Music Algorithm*);
217. Algorytm superpozycji kwantowej QSA (*Quantum Superposition Algorithm*);
218. Algorytm optymalizacji opadów deszczu RFOA (*Rain-Fall Optimization Algorithm*);
219. Zachowanie stada nosorożców RHB (*Rhino Herd Behavior*);
220. Algorytm reinkarnacji RA (*Reincarnation Algorithm*);
221. Algorytm optymalizacji koncepcji reinkarnacji RCOA (*Reincarnation Concept Optimization Algorithm*);
222. Symulowana kolonia pszczół SBC (*Simulated Bee Colony*);
223. Algorytm optymalizacji stada owiec ShFO/SFOA (*Sheep Flock Optimisation Algorithm*);
224. Algorytm dziedziczenia stada owiec SFHA/SFHM (*Sheep Flock Heredity Algorithm /Model*);
225. Prosta optymalizacja SOPT (*Simple Optimization*);
226. Optymalizacja systemem wirusowym VSO (*Viral System Optimisation*);

227. Algorytm kolonii os WCA (*Wasp Colonies Algorithm*);
228. Optymalizacja roju os WSO (*Wasp Swarm Optimization*);

Algorytmy bazujące na zjawiskach chemicznych (*chemical-based*)

229. Sztuczne procesy chemiczne ACP (*Artificial Chemical Process*) [178];
230. Algorytm optymalizacji sztucznych reakcji chemicznych ACROA (*Artificial Chemical Reaction Optimisation Algorithm*) [181];
231. Algorytm optymalizacji sztucznej reakcji chemicznej ARA (*Artificial Reaction Algorithm*);
232. Algorytm optymalizacji reakcji chemicznej CRO (*Chemical Reaction Optimisation Algorithm*) [180];
233. Algorytm chemioterapii CSA (*Chemotherapy Science Algorithm*) [182];
234. Algorytm optymalizacji cząstek gazu GBMO (*Gases Brownian Motion Optimisation Algorithm)* [147];
235. Optymalizacja rozpuszczalności gazu Henry'ego HGSO (*Henry Gas Solubility Optimisation*) [184], oraz modyfikacje: *Opposition-based Learning* (OBL) HGSO, CHGSO (*Chaotic Sequence-based Strategies*), BL-HSGO Lévy-HGSO i Brown-HGSO, QHGSO (*Quantum* HGSO), MHGSO (*Modified* HGSO), HGSORF (HGSO-*based Random Forest*), HHGSO (z poprawioną procedurą sąsiedztwa), LR-HGSO (*Lagrange Relaxation-based* HGSO), MHGSO (*Mutation-based* HGSO), EHGSO (*Exponential*-HGSO), SFS-HGSO (*Stochastic Fractal Search-based* HGSO);
236. Algorytm ruchu jonów IMA/IMO (*Ions Motion Algorithm, Ions Motion Optimisation Algorithm*);
237. Zintegrowany algorytm promieniowania IRA/IRO (*Integrated Radiation Algorithm/Optimisation*);
238. Kinetyczna optymalizacja cząsteczek gazu KGMO (*Kinetic Gas Molecule Optimisation*);
239. Algorytm tworzenia materiałów MGA (*Material Generation Algorithm*);
240. Algorytmy fotosyntetyczne PA (*Photosyntetic Algorithms*);
241. Algorytmy fotosyntetyczne i enzymatyczne PaEA (*Photosyntetic and Enzyme Algorithms*) [183];
242. Symulowane wyżarzanie SA (*Simulated Annealing*) [13], [79], ESA (*Enhanced* SA);
243. Synergiczna optymalizacja fibroblastów SFO (*Synergistic Fibroblast Optimisation*);

244. Optymalizacja wymiany ciepła TEO (*Thermal Exchange Optimisation*) [151];
245. Algorytm optymalizacji chemicznej (C*hemistry-based Metaheuristic Optimisation Method*) [179];

Algorytmy bazujące na zachowaniach ludzkich (społecznościowe)HBs/ HBA (*human-based, human behawior-based*, *social-based)*

246. Optymalizacja społeczeństwa anarchicznego ASO (*Anarchic Society Optimisation*);
247. Algorytm optymalizacji burzy mózgów BSO (*Brain Storm Optimisation Algorithm*) [187];
248. Algorytm optymalizacji burzy mózgów GBSO (*Global-Best Brain Storm Optimisation Algorithm*);
249. Algorytm zachowania w transporcie autobusowym BTA (*Bus Transportation Behavior Algorithm*);
250. Algorytm optymalizacji decyzji zbiorowych CDOA (*Collective Decision Optimization Algorithm*);
251. Algorytm optymalizacji zachowań poznawczych COA (*Cognitive Behavior Optimisation*) [204];
252. Algorytm konkurencyjnej optymalizacji COOA (*Competitive Optimization Algorithm*);
253. Algorytmy kulturowe CA (C*ultural Algorithms*) [212], [213];
254. Algorytm pojedynku DA (*Duelist Algorithm*) [210], DOA (*Duelist Optimisation Algorithm*);
255. Optymalizacja doradztwa grupowego GCO (*Group Counseling Optimisation*); algorytm dynamicznej optymalizacji grupowej DGCO (*Dynamic Group-based Optimization Algorithm*);
256. Algorytm optymalizacji liderów grup GLOA (*Group Leaders Optimization Algorithm*);
257. Zachłanna optymalizacja polityczna GPO (*Greedy Politics Optimisation /Algorithm*);
258. Ludzki algorytm ewolucyjny HEA/HEM (*Human Evolutionary Model*) [103];
259. Tworzenie się grup ludzkich HGF (*Human Group Formation*);
260. Algorytm zainspirowany człowiekiem HIA (*Human-Inspired Algorithm*) [186];
261. Algorytm ideologii IA (*Ideology Algorithm*);
262. Imperialistyczny algorytm konkurencyjny ICA (*Imperialist/imperialistic Competitive Algorithm*) [207], [208], FICA (*Fuzzy Imperialist*

Competitive Algorithm) [209], EICA (*Enhanced Imperialist Competitive Algorithm*), BBICA (*Bare-Bones Imperialist Competitive Algorithm*), MICA-IWO hybrydowy zmodyfikowany ICA z IWO (*Invasive Weed Optimisation*);

263. Algorytm optymalizacji liderów i naśladowców LFA (*Leaders and Followers Algorithm*);
264. Akceptacja starego kawalera OBA (*Old Bachelor Acceptance*);
265. Algorytm wyszukiwania zorientowanego OSA (*Oriented Search Algorithm*);
266. Parlamentarny algorytm optymalizacji POA (*Parliamentary Optimization Algorithm*);
267. Algorytm wyszukiwania kolejkowego QS/QSA (*Queuing Search Algorithm*) [205];
268. Algorytm optymalizacji oparty o zachowania społeczne SaC (*Society and Civilization*) [211];
269. Algorytm optymalizacji oparty o zachowania społeczne SBO (*Social Behavior Optimization Algorithm*);
270. Algorytm optymalizacji społeczno-poznawczej SCOA (*Social Cognitive Optimization Algorithm*);
271. Społeczna optymalizacja emocjonalna SEA/SEOA (*Social Emotional Optimisation Algorithm*);
272. Stochastyczne wyszukiwanie zogniskowane SFS (*Stochastic Focusing Search*);
273. Optymalizacja typu nauczanie-uczenie TLBO/TLB/TLBA/TLBOA (*Teaching-Learning-Based Optimisation/Algorithm*) [188], [189], [190], ulepszony ITLBO (*Improved* TLBO), konwergentna CTLBO (*Converged Teaching-Learning-Based Optimisation*) [191], DGSTLBO (*Dynamic Group Strategy TLBO*), DFL-TLBO (*Distance–Fitness Learning TLBO*), HTLBOHS (*Hybrid* TLBO – *Harmony Search*); hybrydowy adaptacyjny algorytm TLBO z algorytmem DE ATLDE, algorytm uogólnionej opozycyjności GOTLBO (*Generalized Oppositional* TLBO), samodostosowujący się SATLBO (*Self-Adaptive* TLBO), hybrydowy TLABC (*Hybrid Teaching-Learning-based Artificial Bee Colony*), HTC (*Hybrid* TLBO – *Charged System Search* CSS);
274. Optymalizacja oparta na nauczaniu i uczeniu się w trzech fazach TPTLBO (*Triple-Phase Teaching-Learning-based Optimisation*);
275. Wyszukiwanie nieświadome US (*Unconscious Search*);
276. Mądrość sztucznego tłumu WAC (*Wisdom of Artificial Crowds*);

277. Optymalizacja biednych i bogatych PRO (*Poor and Rich Optimisation*);
278. Algorytm optymalizacji pracy zespołowej TOA (*Teamwork Optimisation Algorithm*);
279. Algorytm optymalizacji doktora i pacjenta DPO (*Doctor and Patient Optimisation*);
280. Przeszukiwanie ludzkiego umysłu HMS (*Human Mental Search*);
281. Ali Baba i czterdziestu rozbójników AFT/PPFA (*Ali Baba/ Aladdin's and the Forty Thieves*);
282. Ewolucja rad miejskich CCE (*City Councils Evolution*);
283. Algorytm optymalizacji oparty na wyborach EA/EBOA (*Election Algorithm, Election-based Optimisation Algorithm*);
284. Algorytm kampanii wyborczej ECA (*Election Campaign Algorithm*);
285. Optymalizacja uczenia się społecznościowego SLO (*Social Learning Optimisation*);
286. Przeszukiwanie z tabu TS (*Tabu Search*) [185];
287. Algorytm optymalizacji oparty na równowadze kondycji i odległości oraz uczeniu się sztucznej kolonii pszczół FDB-TLABC (*Fitness-Distance Balance and Learning-based Artificial Bee Colony*);
288. Uczenie ze wzmocnieniem roju cząstek rozmytych FPSRL (*Fuzzy Particle Swarm Reinforcement Learning*);
289. Algorytm społecznościowy SBA (*Social-based Algorithm*);
290. JAYA/JA (*Jaya Algorithm*) [192] oraz różne wersje algorytmu: zmodyfikowana MJA, binarna, samodostosowująca się, parta na elitaryzmie, oparty na elitaryzmie samodostosowujący się wielopopulacyjny algorytm, chaotyczny, sieci neuronowej, hybrydyzacja z algorytmami ewolucyjnymi, hybrydyzacja z algorytmami inteligencji roju, hybrydyzacja z algorytmami fizycznymi, hybrydyzacja z innymi komponentami;
291. Algorytm siatkarskiej ekstraklasy VPL (*Volleyball Premier League Algorithm*);
292. Algorytm oparty na zdobywaniu i udostępnianiu wiedzy GSK (*Gaining Sharing Knowledge-based Algorithm*);
293. Algorytmy poszukiwania harmonii HS/HSA (*Harmony Search*) [193], [194], [195], [196], FHSA (*Fuzzy Harmony Search Algorithm*) [197], udoskonalony IHS (*Improved Harmony Search*), PSF-HS (*Parameter Setting Free Harmony Search*), HHSA (*Hybrid Harmony Search Algorithm*), HHSRS (*Hybrid Harmony Search-Random Search*), HSOSHS (*Hybrid Symbiotic Organisms Searc-Harmony Search*);

294. Adaptacyjne wyszukiwanie harmonii oparte na rodzinie zastępczej ASBHS (*Adaptive Surrogate-based Harmony Search*);
295. Algorytm żyzności gruntów rolnych FF (*Farmland Fertility*) [199];
296. Optymalizacja społecznościowego narciarza SSD (*Social ski–driver Optimisation*) [201], [202];
297. Ewolucja społeczna i optymalizacja uczenia się SELO/ELO (*Socio Evolution and Learning Optimisation, Evolution and Learning Optimisation*) [203];
298. Algorytm optymalizacji poszukiwacza SOA (*Seeker Optimisation Algorithm*) [206];
299. Algorytm ewolucji kulturowej CEA (*Cultural Evolution Algorithm*);
300. Algorytm optymalizacji bazujący na uczeniu się umiejętności kulinarnych CBOA (*Chef-based Optimisation Algorithm*);
301. Optymalizacja oparta na wyborze życiowym LCBO (*Life Choice-based Optimisation*);
302. Optymalizacja społeczno-poznawcza SCO (*Social Cognitive Optimisation*) [215];
303. Algorytm dochodzenia kryminalistycznego FBI (*Forensic-based Investigation Algorithm*);
304. Optymalizator wzrostu GO (*Growth Optimizer*);
305. Algorytm wyszukiwania ścieżek PFA (*Pathfinder Algorithm*) [216];
306. Algorytm działania nerek KA (*Kidney-inspired Algorithm*) [219];
307. Algorytmy memetyczne MA (*Memetic Algorithm*) [220], [221], [222];
308. Algorytm optymalizacji przeciwko koronowirusaowi ACOA (Anti *Coronavirus Optimisation Algorithm*);
309. Optymalizator odporności stada na koronawirusa CHIO (*Coronavirus Herd Immunity Optimizer*) [226];
310. Optymalizacja odporności stada HIO (*Herd Immunity Optimisation*);
311. Algorytm osocza IP/IPA (*Immune Plasma Algorithm*) [227];
312. Algorytm optymalizacji negatywnych skutków sztucznego oświetlenia (*The Ecological Impacts of Nighttime Light Pollution Algorithm*) [69], [228];
313. Optymalizacja wyszukiwania i ratowania SAR (*Search And Rescue Optimisation*);
314. Algorytm harmonii barw CHA (*Color Harmony Algorithm*);
315. Stochastyczny optymalizator malowania SPO (*Stochastic Paint Optimizer*);
316. Optymalizator polityczny PO (*Political Optimizer*);

317. Algorytm optymalizacji uczenia się społeczno-ewolucyjnego SELOA (*Socio Evolution Learning Optimisation Algorithm*);
318. Algorytm ucieczki ESC (*Escape Algorithm*);
319. Optymalizator bazujący na stercie HBO (*Heap-based Optimizer*) zainspirowany korporacyjną hierarchią rang CRH (*Corporate Rank Hierarchy*);
320. Optymalizator ludzi nomadycznych NPO (*Nomadic People Optimizer*);
321. Algorytm nomadycznych NA (*Nomadic Algorithm*);
322. Algorytm optymalizacji umiejętności SOA (*Skill Optimisation Algorithm*);
323. Optymalizator katastrofy w Czarnobylu CDO (*Chernobyl Disaster Optimizer*);
324. Algorytm optymalizacji grup społecznych SGO (*Social Group Optimisation*);
325. Algorytm optymalizacji grupowej GO (*Group Optimisation*);
326. Poszukiwacz oparty na głodzie HGS (*Hunger Games Search)*;
327. Chaotyczna optymalizacja wyszukiwania oparta na głodzie CHGSO (*Chaotic Hunger Games Search Optimisation*);
328. Algorytm optymalizacji inspirowany kaznodziejstwem POA (*Preaching-Inspired Optimisation Algorithm*);
329. Algorytm optymalizacji bitwy królewskiej BRO (*Battle Royale Optimisation Algorithm*);
330. Algorytm wyszukiwania tożsamości nastolatka AISA (*Adolescent Identity Search Algorithm*);
331. Algorytm zachowania oparty na wynikach ucznia LPB (*Learner Performance-based Behavior Algorithm*);
332. Optymalizacja oparta na szkoleniu kierowcy DTBO (*Driving Training-based Optimisation*);
333. Optymalizacja rozwoju rysowania u dzieci CDDO (*Child Drawing Development Optimisation*);
334. Inspirowana naturą metaheurystyczna optymalizacja oparta na wiedzy NMHK (*Nature-Inspired Meta-Heuristic Knowledge-based*);
335. Algorytm optymalizacji nauczania grupowego GTOA (*Group Teaching Optimisation Algorithm*);
336. Algorytm nauki gotowania LCA (*Learning Cooking Algorithm*);
337. Algorytm optymalizacji kierowcy ROA/RideNN (*Rider Optimisation Algorithm*), poprawiony IROA (*Improved Rider Optimisation Algorithm*);

338. Optymalizacja oparta na układzie krążenia CSBO (*Circulatory System-based Optimisation*);
339. Optymalizacja łowca-ofiara HPO (*Hunter-Prey Optimisation*);
340. Algorytm pamięci człowieka HMO (*Human Memory Algorithm*);
341. Optymalizacja gromadą plemników SSO (*Sperm Swarm Optimisation*);
342. Algorytm ruchliwości plemników SMA (*Sperm Motility Algorithm*);
343. Algorytm plemnika wieloryba SWA (*Sperm Whale Algorithm*);
344. Algorytm postępowania w przypadku raka wątroby LCA (*Liver Cancer Algorithm*);
345. Algorytm fanów FO (*Fans Optimisation*);
346. Optymalizacja oparta na zachowaniu człowieka HBBO (*Human Behavior-based Optimisation*);
347. Elastyczna optymalizacja oparta na wyborach EVEBO (*EVolutive Election-based Optimisation*);
348. Optymalizacja oparta na popycie i podaży SDO (*Supply Demand–based Optimisation*);
349. Optymalizacja oparta na psychologii ucznia SPBO (*Student Psychology-based Optimisation*);
350. Optymalizacja inwazyjnym wzrostem nowotworu ITGO (*Invasive Tumor Growth Optimisation*);
351. Nauczanie technik optymalizacji OT (*Teach Optimisation Techniques*);
352. Zaawansowany algorytm przeszukiwania ładunku ACSS (*Advanced Charged System Search*);
353. Optymalizacja wyszukiwania grupowego GSO (*Group Search Optimizer*) [229];
354. Optymalizacja lidera grupy GLO (*Group Leader Optimization*);
355. Algorytm pracownika-pracodawcy WEA (*Worker Employer Algorithm*);
356. Wyszukiwanie według wskazówek konsultanta CGS (*Consultant Guide(d) Search*);
357. Algorytm przeszukiwania myśliwskiego HuS/HS (*Hunting Search / Algorithm*) [230];
358. Algorytm ochrony w maseczce przed koronawirusem CMPA (*Coronavirus Mask Protection Algorithm*);
359. Optymalizacja strategii wojennej WSO (*War Strategy Optimisation*), WSO-HO hybryda WSO i optymalizacji hipopotamów HO (*Hippopotamus Optimisation*);
360. Algorytm sadowniczy OA (*Orchard Algorithm*);

361. Algorytm optymalizacji edukacji przedszkolnej PEOA (*Preschool Education Optimisation Algorithm*);
362. Optymalizator dla wielu liderów MLO (*Multi-Leader Optimizer*);
363. Podążający algorytm optymalizacji FOA (*Following Optimisation Algorithm*);
364. Macierzysty algorytm optymalizacji MOA (*Mother Optimisation Algorithm*);
365. Optymalizacja agenta podążającego za źródłem cząsteczek zapachowych SAO (*Smell Agent Optimisation*);
366. Optymalizacja mimiki (naśladowcy) społecznej SMO (*Social Mimic Optimization*), ESM) (*Elite Social Mimic Optimization*);
367. Algorytm przewidywania ExA (*Expectation Algorithm*);
368. Algorytm inspirowany mediami społecznościowymi SMIA (*Social Media Inspired Algorithm*);
369. Algorytm społeczeństwa i cywilizacji SACA (*Society and Civilization Algorithm*);
370. Algorytm optymalizacji dynastycznej DOA (*Dynastic Optimization Algorithm*);
371. Optymalizator mieszany oparty na liderach MLBO (*Mixed Leader Based Optimizer*);
372. Algorytm widzenia ludzkiego oka HEVA (*Eye Vision Algorithm*);
373. Algorytm ludzkiego szczęścia HFA (*Human Felicity Algorithm*);
374. Optymalizacja oparta na treningu szycia STBO (*Sewing Training-Based Optimization*);
375. Algorytm optymalizacji antykoronawirusowej ACVO (*Anti-Coronavirus Optimization Algorithm*);
376. Algorytm optymalizacji oparty na liderach-adwokatach-wierzących LAB (*Leader– Advocate–Believer-based Optimization Algorithm*);
377. Optymalizator gorączki złota GRO (*Gold Rush Optimizer*);
378. Optymalizacja dynamicznego przywództwa łowieckiego DHL (*Dynamic Hunting Leadership Optimization*);
379. Optymalizacja znajomych influencerów InBO (*Influencer Buddy Optimization*);
380. Wieloskalowy algorytm uczenia się medalistów MMLA (*Multiscale Medalist Learning Algorithm*);

Bazujące na grach/sporcie (*Game/sports-based*)

381. Algorytm optymalizacji bazujący na snookerze SBOA (*Snooker-based Optimisation Algorithm*);

382. Algorytm strzelania z łuku AA (*Archery Algorithm*);
383. Optymalizacja inspirowana jazdą na nartach ASO (*Alpine Skiing Optimisation*);
384. Optymalizacja zespołowa w alpinizmie MTBO (*Mountaineering Team-based Optimisation*);
385. Optymalizacja przeciągania liny TWO (*Tug-of-War Optimisation*);
386. Algorytm inspirowany grą w piłkę nożną FGA/FGIA/FGBO (*Football Game Algorithm, Football*
387. *Game-Inspired Algorithm, Football Game-based Optimisation*) [231];
388. Algorytm Ligi Mistrzów LCA (*League Championship Algorithm*) [232],
389. Optymalizacja ligii piłkarskiej SLO (*Soccer League Optimization*);
390. Optymalizacja gry piłkarskiej SGO (*Soccer Game Optimization*);
391. Algorytm zawodów w piłce nożnej SLC/SLCA (*Soccer League Competition Algorithm*) [233];
392. Algorytm optymalizacji mistrzostw świata FIFAAO (*FIFA World Cup Competition Optimisation Algorithm*);
393. Optymalizacja mistrzostw świata FIF WCO (*World Cup Optimization*);
394. Optymalizacja inspirowana pierwszą ligą siatkówki PVL (*Premier Volleyball League*);
395. Algorytm optymalizacji układanki (puzzli) POA (*Puzzle Optimisation Algorithm*);
396. Algorytm wyszukiwania orientacyjnego OSA (*Orientation Search Algorithm*);
397. Optymalizacja gry w ukrywanie obiektów HOGO (*Hide Objects Game Optimisation*);
398. Optymalizacja oparta na grze Ring Toss (rzucanie obręczami) RTGBO (*Ring Toss Game-based Optimisation*);
399. Optymalizacja gry w kości DGO (*Dice Game Optimisation*);
400. Algorytm optymalizacji bazujący na nauczaniu fitness na odległość DFL (*Distance-Fitness Learning*);
401. Algorytm gry zespołowej TGA (*Team Game Algorithm*);
402. Inteligencja rojowa bazująca na grze planszowej Ludo LGSI (*Ludo Game-based Swarm Intelligence*);
403. Algorytm optymalizacji polowania na jelenie DHOA (*Deer Hunting Optimisation Algorithm)*;
404. Algorytm najbardziej wartościowego gracza MVPA (*Most Valuable Player Algorithm*);
405. Optymalizacja handlu na giełdzie SETO (*Stock Exchange Trading Optimisation*);

406. Optymalizator gry *Squid* SGO/SGOa (*Squid Game Optimizer*);
407. Optymalizator gry w rzutki DGO (*Darts Game Optimizer*);
408. Optymalizator gry w chaos CGO (*Chaos Game Optimisation*);
409. Optymalizacja wyścigów smoczych łodzi DBO (*Dragon Boat Optimisation*);
410. Algorytm konkursu złotej piłki GB (*Golden Ball*);
411. Algorytm optymalizacyjny bazujący na grze z Indii Kho-Kho KKO (*Kho-Kho Optimization Algorithm*);
412. Optymalizacja bazujaca na grze w trzy kubki SGO (*Shell Game Optimization*);
413. Algorytm bazujący na stylu tiki taka gry w piłkę nożną TTA (*Tiki-Taka Algorithm*);
414. Algirytm meczu bokserskiego (walki bokserskiej) BMA (*Boxing Match Algorithm*);

Algorytmy inteligencji rojowej SI (*Swarm Intelligence/Swarm-Inspired/Swarm-based*) *Biology-based, Nature-based, Animal-based)*

Zjawiska

415. Algorytm cyklu wody WCA (*Water Cycle Algorithm*) [165], chaotyczny cykl wody CWCA (*Chaotic Water Cycle Algorithm*) [166], z chaotycznymi sekwencjami i lotami Lévy'ego CLWCA (*Chaotic* Lévy *Water Cycle Algorithm*) [167] oraz innymi modyfikacjami [166], NGBWCA (*Ggaussian Bare-bones* WCA);
416. Algorytm wody deszczowej RWA (*Rain Water Algorithm*) [168];
417. Algorytm optymalizacji opadu deszczu RFO (*Rain-Fall Optimization Algorithm*);
418. Inteligentny algorytm kropli wody IWD (*Inteligent Water Drops Algorithm*) [139];
419. Optymalizacja fal wodnych WWO (*Water Wave Optimisation*);
420. Algorytm fal wodnych WWA (*Water Wave Optimisation Algorithm*);
421. Algorytm cyklu hydrologicznego HCA (*Hydrological Cycle Algorithm*) [169];
422. Dynamika formowania rzeki RFD (*River Formation Dynamics*) [170];
423. Algorytm dynamiki formowania się rzeki RFDA (*River Formation Dynamics Algorithm*);
424. Optymalizator przepływu wody WFO (*Water Flow Optimizer/Algorithm Optimisation*), poprawiony IWOF (*Impoved WFO*) [172], CCWFO (*Crisscross-Strategy-Boosted Water Flow Optimizer*);

425. Optymalizator przepływu wody WFA (*Water Flow-Like Algorithm*);
426. Optymalizator przepływu wody WFA (*Water Flow Algorithm*);
427. Optymalizacja turbulentnego przepływu wody TFWO (*Turbulent Flow of Water-based Optimisation*);
428. Optymalizator ablacji (topnienia) śniegu SAO (*Snow Ablation Optimizer*);
429. Algorytm pasterza (owiec) SSO/SSOA/SSAO (*Shuffled Shepherd Optimization Algorithm*), wersja ulepszona ESSOA (*Enhanced Shuffled Shepherd Optimization Algorithm*);

Ssaki

430. Algorytm jelenia szlachetnego RDA/RDO (*Red Deer Algorithm*), zmodyfikowany (*Modified Reed Deer Algorithm*);
431. Optymalizacja stada antylop gnu WHO (*Wildebeest Herd Optimisation*);
432. Optymalizacja stada łosi EHO (*Elk Herd Optimisation*) [234];
433. Algorytm nietoperzy BA/BAT (B*at Algorithm*) [235], [236] z modyfikacjami [237] algorytm: ulepszony z systemem samoadaptacji, lokalnym przeszukiwaniem pamięci, adaptacyjny, wielopopulacyjny, binarny BBA (*Binary Bat Algorithm*), hybrydowy samodostosowujący się, chaotyczny CBA [238], ulepszony [239], zoptymalizowany binarny OBBA (*Optimized Binary Bat Algorithm*);
434. Algorytm wyszukiwania wilka WSA (*Wolf Search Algorithm*) [240];
435. Algorytm migracji łososi TGSR/GSR (*The / Great Salmon Run*);
436. Algorytm optymalizacji szarego wilka (*Canis lupus*) GWO (*Grey Wolf Optimizer*) [241], [242], MDGWO (*Modified Discrete Grey Wolf Optimizer Algorithm*) [243], QI-BGWO (*Quantum Inspired Binary Grey Wolf Optimizer*), MGWO (*memory-based GWO*), chaotyczny CGWO [244], hybrydowy HGWO, z lotami Lévy'ego LGWO, IGWO, GWO-PSO, binarny BGWO, QIBGWO, algorytm kwantowy binarny szarego wilka QBGWO (*Quantum Inspired Binary Grey Wolf Optimizer*), ulepszony IGWO (*Improved Gray Wolf Algorithm*), SCA-GWO (*Sine-Cosine Algorithm* z *Grey Wolf Optimizer*), GWO-PSO/ PSOGWO (*Hybrid Grey Wolf Optimisation-Particle Swarm Optimisation*), GWOCS, wybiórczy oparty na opozycji (*Selective Opposition-based* GWO), hybryda GWO z algorytmem optymalizacji pasikonika (konika polnego) GWO-GOA (*Grey Wolf Optimizer* z GlowwormSwarm *Algorithm*), chaotyczny GWO (*Chaotic* GWO). MDE-GWO (*Modified Differential Evolution* GWO), ulepszony GWO oparty na logarytmicznej wadze bezwład-

ności LGWO *(Logarithmic Inertia Weight* GWO), ulepszony EGWO (*Enhanced Gray Wolf Optimizer*);

437. Elitarne uczenie się oparte na opozycji i chaotyczna k-najlepsza strategia wyszukiwania grawitacyjnego bazująca na optymalizatorze szarego wilka EOCSGWO (*Elite Opposition-based Learning and Chaotic k-best Gravitational Search Strategy-based Grey Wolf Optimizer*);
438. Algorytm wielorybi/optymalizacji wieloryba WOA (*Whale Optimisation Algorithm*) [245], [246], lot Lévy'ego LWOA/LFWOA [247], [248], grupowej optymalizacji GWOA (*Group-based Whale Optimisation Algorithm*) [249], ulepszony EWOA (*enhanced WOA*), chaotyczny CWOA [250], IWOA; hybrydowy SA-WOA, hybrydowy WOA-DE; bazujący na opozycji (*Opposition-based* WOA), MWOA (*Modified* WOA);
439. Algorytm gromady wielorybów WSA (*Whale Swarm Algorithm*) [22];
440. Algorytm/optymalizator hieny cętkowanej SHO (*Spotted Hyena Optimizer*) [251];
441. Optymalizator lisa piaskowego RFO (*Rüppell's Fox Optimizer*);
442. Algorytm wilczego stada WPA (*Wolf Pack Algorithm)* [252],
443. Algorytm przeszukiwania wiewiórki SSA/SqSA (*Squirrel Search Algorithm*) [253];
444. Algorytm echolokacji delfinów DE/ DA (*Dolphins Echolocation Algorithm*) [254];
445. Algorytm optymalizacji stada delfinów DSOA (*Dolphin Swarm Optimisation Algorithm*);
446. Algorytm zachowania żerowania delfinów butlonosych MRA (*Mud Ring Algorithm*);
447. Optymalizacja sfory kotów pustynnych (piaskowych) SCSO (*Sand Cat Swarm Optimisation*) [255], wariant hybrydowy z wieloma strategiami HSCSO, MSCO (*Modified Sand Cat Swarm Optimisation*), ISCSO (*Improved* SCSO) [256];
448. Algorytm kociej sfory CSO/CSA (*Cat Swarm Optimisation/Algorithm*) [257], [258], [259], CSOA (*Cat Swarm Optimisation Algorithm*), CSO-BM/ CSO-BMA (*Block Matching /Algorithm*), CSO-BMA, kwantowy QCSO, ICSO (*Improved CSO*), AICSO (*Average inertia Weighted CSO*), (*Best CSO*), ADSCSO (*Adaptive Cat Swarm Algorithm-based on Dynamic Search*) oraz inne modyfikacje (autorzy [260] zestawili 29 algorytmów);
449. Algorytm optymalizacji ratela miodożernego HBOA/HBA (*Honey Badger Optimisation Algorithm*) [261];

450. Algorytm optymalizacji wieloryba-białuchy arktycznej BWO/BWOA (*Beluga Whale Optimisation Algorithm/Optimizer*) [262];
451. Ewolucyjny algorytm króla małp MKE (*Monkey King Evolutionary*) [263];
452. Algorytm drapieżnictwa orki OPA (*Orca Predation Algorithm*) [264];
453. Optymalizator szympansa karłowatego (bonobo) BO (*Bonobo Optimizer*), IBO (*Improved Bonobo Optimizer*);
454. Algorytm orki zabójcy KWA (*Killer Whale Algorithm*) [168], [265];
455. Wyszukiwanie małpie MS (*Monkey Search/Optimisation*);
456. Algorytm koczkodana czarnosiwego BMA (*Blue Monkey Algorithm*);
457. Algorytm afrykańskiego dzikiego psa AWDA (*African Wild Dog Algorithm*);
458. Algorytm stada wielbłądów CHA (*Camel Herds Algorithms*);
459. Algorytm nosorożca RSA (*Rhinoceros Search Algorithm*);
460. Algorytm lwi LOA (*Lion Optimisation Algorithm*);
461. Algorytm optymalizacji gazeli GOA (*Gazelle Optimisation Algorithm*);
462. Optymalizator gazeli górskiej MGO (*Mountain Gazelle Optimizer*);
463. Algorytm małpi MAs (*Monkey Algorithms*);
464. Algorytm optymalizacji pieska preriowego (nieświeszczuka czarnoogonowego) PDO (*Prairie Dog Optimisation Algorithm*), ulepszony algorytm IPDO;
465. Optymalizator dzikich koni WHO *(Wild Horse Optimizer)*;
466. Algorytm optymalizacji stada koni HOA (*Horse herd Optimisation Algorithm*);
467. Algorytm optymalizacji szympansów ChOA/COA (*Chimp Optimisation Algorithm*);
468. Optymalizator oddziałów sztucznych goryli AGTO/GTO (*Artificial Gorilla Troops Optimizer*);
469. Algorytm optymalizacji australijskiego dzikiego psa DOA (*the australian Wild Dog Optimisation Algorithm*);
470. Optymalizacja oparta na zachowaniu kotów i mysz CMBO (*Cat and Mouse-based Optimisation*);
471. Optymalizacja sztucznych królików ARO/ARA (*Artificial Rabbits Optimisation/Algorithm*), AFDB-ARO (*Adaptive Fitness-Distance Balance-based Artificial Rabbits Optimisation*);
472. Chaotyczna optymalizacja sztucznych królików CARO (*Chaotic Artificial Rabbits Optimisation*);
473. Adaptacyjna optymalizacja lisa AFO (*Adaptive Fox Optimisation*);

474. Optymalizator oparty na zachowaniu geparda CBO/CBA (*Cheetah–based Optimizer/ Algorithm*) [266];
475. Optymalizator geparda CO (*Cheetah Optimizer*) [267];
476. Optymalizator dingo (rodzina psowatych) DOX (*Dingo Optimizer*);
477. Algorytm małpy czerwonolicej RCM (*Red Colobuses Monkey*);
478. Optymalizator szczurzego stada RSO (*Rat Swarm Optimizer*);
479. Algorytm czepiaka SMO (*Spider Monkey Optimisation*) [268], hybrydowy HSMO (*Hybrid Spider Monkey Optimisation*) [269];
480. Algorytm optymalizacji kojota COA (*Coyote Optimisation Algorithm*) [270];
481. Dynamiczny algorytm wirtualnych nietoperzy DVBA (*Dynamic Virtual Bats Algorithm*) [237];
482. Optymalizacja stada słoni EHO (*Elephant Herding Optimisation*) [271];
483. Optymalizacja rudego lisa RFO (*Red Fox Optimisation*);
484. Algoryt optymalizacyjny „latających lisów" – rudawek FFO (*Flying Fox Optimization Algorithm*);
485. Optymalizacja lwa morskego (uszanki/uchatki kalifornijskiej) SLnO (*Sea lion Optimisation*) [272];
486. Algorytm optymalizacji wokalizacji humbaka (długopłetowca oceanicznego) VWOA (*Vocalization of humpback Whale Optimisation Algorithm*) [273];
487. Algorytm klanu surykatek MCA (*Meerkat Clan Algorithm*);
488. Algorytm optymalizacji surykatek MOA (*Meerkat Optimisation Algorithm*);
489. Algorytm golca piaskowego NMR/NMRA (*Naked Mole-Rat Algorithm*);
490. Algorytm optymalizacji mangusty karłowatej DMOA/DMO (*Dwarf Mongoose Optimisation Algorithm*) [274];
491. Algorytm optymalizacji gromadą szczurów RSA (*Rat Swarm Optimisation Algorithm*);
492. Optymalizacja amfitryty lamparciej (rodzina: fokowate) LSO (*Leopard Seal Optimisation*);
493. Optymalizator pumy PO (*Puma Optimizer*);
494. Optymalizacja pantery mglistej CLO (*Clouded Leopard Optimization*);
495. Optymalizacja szakala złocistego GJO (*Golden Jackal Optimisation*);
496. Optymalizacja lisa pustynnego (fenek pustynny) FFO/FFA (*Fennec Fox Optimisation/Algorithm*);
497. Algorytm wyszukiwania kapucynek (płaskowate) CapSA (*Capuchin Search Algorithm*);

498. Optymalizacja hipopotamów HO (*Hippopotamus Optimisation*);
499. Algorytm optymalizacji amerykańskiej zebry (wymarłego gatunku konia Hagermana/ *Equus simplicidens*) AZOA (*American Zebra Optimisation Algorithm*);
500. Algorytm optymalizacji zebry ZOA (*Zebra Optimisation Algorithm*);
501. Algorytm szympansa karłowatego BO/BOA (*Bonobo Optimizer*), ABOA (*Adaptive Bonobo Optimisation Algorithm*);
502. Algorytm optymalizacji szopa pracza ROA (*Raccoon Optimisation Algorithm*);
503. Algorytm optymalizacji morsa WaOA (*Walrus Optimisation Algorithm*);
504. Optymalizator morsa WO (*Walrus Optimizer*);
505. Optymalizacja diabła tasmańskiego TDO (*Tasmanian Devil Optimisation*);
506. Algorytm optymalizacji ostronosa COA (*Coati Optimisation Algorithm*);
507. Algorytm wieloryba (modyfikacja) WOABHC (*Whale Optimisation Algorithm* z *β-hill climbing*);
508. Optymalizacja twierdzenia osła DTO (*Donkey Theorem Optimisation*);
509. Algorytm optymalizacji hipopotama HOA (*Hippopotamus Optimisation Algorithm*);
510. Nowy algorytm nietoperza NBA (*Novel Bat Algorithm*);
511. Algorytm kierunkowy nietoperza DBA (*Directional Bat Algorithm*);
512. Optymalizacja bawołu afrykańskiego ABO (*African Buffalo Optimization*);
513. Algorytm optymalizacji byka BOA (*Bull Optimisation Algorithm*);
514. Algprytm optymalizacji pantery śnieżnej (irbisa) SLOA (*Snow Leopard Optimisation Algorithm*);
515. Algorytm zachowania żubra BBA (*Bison Behavior Algorithm*);
516. Kulturowy algorytm optymalizacji kojota CCOA (*Cultural Coyote Optimization Algorithm*);
517. Algorytm optymalizacji kojota COA (*Coyote Optimization Algorithm*);
518. Optymalizacja partnerów delfina DPO (*Dolphin Partner Optimization*);
519. Algorytm jaguara JA (*Jaguar Algorithm*);
520. Algorytm inspirowany surykatkami MIA (*Meerkats Inspired Algorithm*);
521. Algorytm wyszukiwania słonia ESA (*Elephant Search Algorithm*);

Ptaki

522. Algorytm Hajiaghaei-Keshteli-Aminnayeri (zachowanie żywieniowe kaczek Anas) KA (*Keshtel Algorithm*);
523. Optymalizacja rojem cząstek PSO (*Particle Swarm Optimisation*) [275], [276] (inaczej algorytm ptasi), ptymalizacja chaotycznym rojem cząstek CPSO (*Chaotic Particle Swarm Optimisation*) [277] oraz jej inne modyfikacje AIW-PSO, B-PSO, CDIW-PSO, DW-PSO, EIW-PSO, NLDA–PSO, NLI-PSO, OIW-PSO, RIW-PSO [278], CPSO-AT [279], IPSO (*Improved Particle Swarm Optimisation*), PSOEM (*Extended Memory*), zmodyfikowany MPSO (*Modified Particle Swarm Optimizatio*), BLPSO, CGPSO; CLPSO (*Comprehensive Learning* PSO), optymalizacja rojem cząstek oparta na współczynniku odległości i wydajności FDR–PSO (*Fitness-Distance-Ratio-based* PSO), zespołowa optymalizacja EPSO (*Ensemble Particle Swarm Optimiser*), ulepszony mPSO (*improved PSO*), ulepszony dyskretny DPSO (*improved Discrete PSO*), ulepszony szkielet IBPSO (*Improved bare skeleton PSO*), QPSO (*Quantum-based* PSO), HS-PSO (*Hybrid Symmetry*-PSO), hybrydowy PSOGA (*Hybrid* PSO-GA), ulepszona wiodąca optymalizacja rojem cząstek EPSO (*Enhanced Leader Particle Swarm Optimisation*), dopasowany APSO (*alignment* PSO), APSO (*Accelerated* PSO), wyżarzany APSO (*Annealing-based PSO*), algorytm kwantowy aptymalizacji rojem cząstek QPSO (*Quantum Inspired Particle Swarm Optimisation*), optymalizacja rojem cząstek z mutacją elitarną EMPSO (*Elitist-Mutation Particle Swarm Optimisation*), PSwarm (*Particle Swarm Pattern Search Algorithm*), starzejący się przywódca i pretendenci ALC-PSO (*Aging Leader and Challengers* PSO), algorytm hybrydowy PSO-DE, PSO-ICA (PSO z *Imperialist Competitive Algorithm*), hybryda PSO z algorytmem grawitacyjnym PSO-GSA (*Gravitational Search Algorithm*), HFPSO (*Hybrid Firefly* i PSO);
524. Algorytm kojarzenia dzięciołów WMA (*Woodpecker Mating Algorithm*);
525. Algorytm kondora andyjskiego ACA (*Andean Condor Algorithm*);
526. Polityka optymalizacji roju cząstek PSO-P (*Particle Swarm Optimisation Policy*);
527. Algorytm roju nieważkości WSA (*Weightless Swarm Algorithm*);
528. Algorytm sztucznego kolibra AHA (*Artificial Hummingbird Algorithm*) [280];

529. Algorytm inspirowany agresywnymi ptakami z fimu Hitchcocka HBIA (*Hitchcock Bird-Inspired Algorithm*);
530. Algorytm drapieżnictwa kolonii CPA (*Colony Predation Algorithm*) [281], ulepszenie CPA wersja hbrydowa z NM (sympleks Neldera–Meada);
531. Przeszukiwanie kukułczane (algorytm kukułki) CS/CSA (C*uckoo Search / Algorithm*) [183], [282], [283], COA (C*uckoo Optimisation Algorithm*) [284], zmodyfikowany algorytm MCS (*Modified Cuckoo Search*) [285], rozszerzony ECS (*Enhanced Cuckoo Search*) [286], ulepszony ICS (*Improved Cuckoo Search*) [253], HCS (*Hybrid Cuckoo Search*), hybrydowy FCS (*Fuzzy Cuckoo Search*) [209], DMCS (*Double Mutation Cuckoo Search Algorithm*);
532. Ulepszony algorytm kukułki ImCSA (*Improved CS Algorithm*);
533. Eksploracyjne wyszukiwanie kukułki ECS (*Exploratory Cuckoo Search*);
534. Strategia orła ES (*Eagle Strategy*) [287];
535. Optymalizator orła AO/AQO (*Aquila Optimizer*), ulepszony optymalizator orła IAO (*Improved Aquila Optimizer*);
536. Optymalizator orła przedniego GEO (*Golden Eagle Optimizer*);
537. Algorytm optymalizacji stada czapli ESOA (*Egret Swarm Optimisation Algorithm*) [290], [291];
538. Algorytm stada szpaków SMO (*Starling Murmuration Optimizer*) [292], BSMO (*binary SMO*);
539. Algorytm optymalizacji roju jaskółek SSO (*Swallow Swarm Optimisation Algorithm*);
540. Algorytm ptasiego roju BSA (*Bird Swarm Algorithm*) [293], chaotyczny algorytm ptasiego roju CBSA (*Chaotic Bird Swarm Algorithm*);
541. Algorytm wyszukowania wrony CSA (*Crow Search Algorithm*) [294]; ECSA (*Evolutionary Crow Search Algorithm*) poprawiony ECSA (*Emended Crow Search Algorithm*), CCSA (*Conscious Neighborhood-based Crow Search Algorithm*);
542. Przeciwstawny algorytm wyszukowania wrony OCSA (*Oppositional Crow Search Algorithm*);
543. Algorytm optymalizacji łyski COA (*Coot Optimisation Algorithm*) [295] oraz ulepszony algorytm ECOA (*Enhanced Coot Optimisation Algorithm*) [296], COOT (*Coot Bird Natural Life Model*) [297];
544. Algorytm myszołowca towarzyskiego HHO *(Harris Hawks Optimisation*) [298], CMDHHO (*DE-driven Multi-Population HHO*), IHHO

(*Improved HHO*), MIHHO-AM (*Multi leader HHO Adaptive Mutation*), HHOSA (*Harris Hawks Optimisation Simulated Annealing*);
545. Chaotyczny algorytm myszołowca towarzyskiego CHHO (*Chaotic Harris Hawks Optimisation*);
546. Instynktowna strategia reakcji oparta na optymalizacji myszołowca towarzyskiego IRSHHO (*Instinctive Reaction Strategy-based on Harris Hawks Optimisation*);
547. Chaotyczna nieliniowa optymalizacja optymalizacji myszołowca towarzyskiego NCHHO (*Nonlinear-based Chaotic Harris Hawks Optimisation*);
548. Algorytm optymalizacji afrykańskich sępów AVO/AVOA (*African Vultures Algorithm)* [299];
549. Algorytm optymalizacji egipskiego sępa EV/EVOA (*Egyptian Vulture Optimisation Algorithm*);
550. Algorytm wyszukiwania wróbla SSA (*Sparrow search algorithm*) [300], CSSA (*chaotic SSA*) [301];
551. Wyszukiwanie bielika amerykańskiego BES (*Bald Eagle Search/Optimisation*) [302], [303], zmodyfikowany mBES [304];
552. Algorytm optymalizacji mewy SOA (*Seagull Optimisation Algorithm*) [305];
553. Optymalizacja ptaków wędrownych MBO (*Migrating Birds Optimisation*);
554. Algorytm żerowania ptaków BFS (*Birds Foraging Search Algorithm*);
555. Algorytm zielonej czapli GHO (*Green Heron Optimisation*);
556. Algorytm układania kurcząt LCA (*Laying Chicken Algorithm)*;
557. Algorytm gołębi PIO (*Pigeon-Inspired Optimisation*);
558. Algorytm nocujących kruków RRO (*Raven Roosting Optimisation*);
559. Kwantowy optymalizator nawigacji ptaków QANA (*Quantum-based Avian Navigation Optimizer*);
560. Algorytm optymalizacji pelikana POA (*Pelicon/Pelican Optimisation Algorithm*);
561. Algorytm dzikiej gęsi WGA (*Wild Goose Algorithm*);
562. Algorytm zachowania gęsi GOOSE [306];
563. Algorytm optymalizacji głuptaka białego (*Morus bassanus*) GOA/GaOA (*Gannet Optimisation Algorithm*);
564. Optymalizacja jastrzębia zwyczajnego/gołębiarza NGO (*Northern oshawk Optimisation*);
565. Optymalizacja gęsi gęgawy GGO (*Greylag Goose Optimisation*);

566. Algorytm gromady kaczek DSA (*Duck Swarm Algorithm*) [307];
567. Algorytm kurzego roju CSO/CSOA (*Chicken Swarm Optimisation Algorithm*); DCSO (*Dynamic Distribution Chicken Swarm Optimisation*);
568. Optymalizator pingwinów cesarskich EPO (*Emperor Penguins Optimizer*) [308], oraz inne modyfikacje: IEPO, MSEPO, QEPO, EPO z eQuest, EPOW, EPOSH, EPOUA, BEPO, CEPO-ELM, EPO-SEO, CEPO-SVM, ESA, MOSHEPO, MoSSE, HDNN-EPO, Hybrydowy EPO, HDNN-AGSO, HDEPO, EPO-BES, HEPSO, DON-MOEPO, MOEPO, MOO, MOEPHO, ACMOEPO;
569. Optymalizator pingwinów cesarskich AFO (*Aptenodytes Forsteri Optimisation*);
570. Optymalizator kolonii pingwinów cesarskich EPC (*Emperor Penguins Colony Optimizer*);
571. Algorytm jastrzębia czerwonogoniastego RTH (*Red-Tailed Hawk Algorithm*);
572. Algorytm optymalizacji orzechówki NOA (*Nutcracker Optimisation Algorithm*);
573. Optymaizator jastrzębia ognistego FHO (*Fire Hawk Optimizer*);
574. Sokoli algorytm optymalizacji FOA (*Falcon Optimisation Algorithm*);
575. Algorytm optymalizacji błotowca SOA (*Sandpiper Optimisation Algorithm*);
576. Optymalizator altannika lśniącego SBO (*Satin Bowerbird Optimizer*) [309], [310];
577. Optymalizator zimorodka (rybaczka) srokatego PKO (*Pied Kingfisher Optimizer*);
578. Hybrydowe wyszukiwanie kukułki/optymalizator oparty na biogeografii BHCS (*Hybridized Cuckoo Search/Biogeography-based Optimizer*);
579. Hybryda metody naturalnego przetrwania z algorytmem sztucznego kolibra NSM-AHA (*Natural Survivor Method -Artificial Hummingbird Algorithm*);
580. Optymalizacja maskonurów APO (*Arctic Puffin Optimisation*);
581. Algorytm optymalizacji rybitwy popielatej STOA (*Sooty Tern Optimisation Algorithm*);
582. Algorytmu poszukiwania uciekających ptaków EBS (*Escaping Bird Search*);
583. Pawi algorytm optymalizacyjny POA/PaOA (*Pavo muticus/cristatus Optimisation Algorithm*);

584. Optymalizacja pluszcza (*Cinclus cinclus*) DTO (*Dipper Throated Optimization*);
585. Optymalizator ostrygojada zwyczajnego EOO (*Eurasian Oystercatcher Optimiser*);
586. Algorytm optymalizacyjny rybołowa OOA (*Osprey Optimisation Algorithm);*
587. Heurystyczna optymalizacja dudków HHO (*Hoopoe Heuristic Optimization*);
588. Algorytm optymalizacji sekretarza (*Sagittarius serpentarius)* SBOA (*Secretary Bird Optimization Algorithm*);
589. Algorytm optymalizacji kukułki COA (*Cuckoo Optimization Algorithm*);
590. Szybki dryf z algorytmem przeszukiwania kukułki SDCS (*Snap-Drift Cuckoo Search*);
591. Optymalizacja stada gęsi GTO (*Goose Team Optimization*);
592. Optymalizacja piskląt kuropatewki piaskowej SSPCO (*See-See Partidge Chicks Optimization*);
593. Optymalizacja kanii rudej RKO (*Red Kite Optimisation*) [414];
594. Ulepszony algorytm gawrowania kruków IRRO/IRRA (*Improved Raven Roosting Algorithm*);

Gady

595. Hybrydowy algorytm optymalizacji węża HSOA (*Hybrid Snake Optimizer Algorithm*);
596. Optymalizator węża SO (*Snake Optimizer*) [311];
597. Algorytm stada kamelonów CSA/ChSA (*Chameleon Swarm Algorithm*) [312];
598. Algorytm przeszukiwania gadów (polowania krokodyli) RSA (*Reptile Search Algorithm*), poprawiony algorytm IRSA (*Improved Reptile Search Algorithm);*
599. Algorytm jaszczurki Uta SBLA (*The Side-Blotched Lizard Algorithm*);
600. Algorytm optymalizacji jaszczurki rogatej HLOA (*Horned Lizard Optimisation Algorithm*);
601. Optymalizacja wyszukiwania za pomocą sztucznej jaszczurki ALSO (*Artificial Lizard Search Optimisation*);
602. Algorytm warana z Komodo ze spacjalnym sposobem poruszania się (*Mlipir*) KMA (*Komodo Mlipir Algorithm*) [313];

Płazy

603. Algorytm wywoływania japońskich żab drzewnych FROGSIM (*Japanese Tree Frogs Calling Algorithm*);
604. Algorytm żabiego skoku SFL/SFLA (S*huffled Frog Leaping Algorithm*) [314], [315], hybrydowy IWO-SFLA (*Hybrid Invasive Weed Optimisation-Shuffled Frog-Leaping Algorithm*);
605. Optymalizacja ambystomy meksykańskiej (aksolotla meksykańskiego) *Ambystoma mexicanum* MAO (*Mexican Aksalotl Optimization*);

Ryby

606. Algorytm optymalizacyjny ryb rozdymkowatych POA (*Pufferfish Optimisation Algorithm*);
607. Algorytm sztucznej ławicy ryb AFSA/ASFA (*Artificial Fish Swarm Algorithm*) [316], [317] z wieloma modyfikacjami [318]: wersja hybrydowa (*Hybrid*) AFSA, AFSA-PSO, ABC-AFSA, CIAFSA, NFSA, PSO-EM-FSA, MSAFSA, FDMABC, SA-IAFSA, AF-GBFO, AFSA-BFO, LFFSA, HAFPSO, RCGA-AFSA, AFSAACA-BPN, DN-AFS, RP-ABC, BCC-AFSA, RNA-AFSA, GAFSA-SVR, GAFSA, FSLR, MSAFSA-SVR, ZAFSA, CPAFSA, AFSA-CA, mAFSA;
608. Algorytm ławicy ryb FSA/FSO (*Fish Swarm Algorithm/Optimization);*
609. Algorytm inspirowany zachowaniem ryb w czasie odpływu ETFI/ETFA (*Ebb-Tide-Fish-Inspired Algorithm*) [319];
610. Algorytm wylęgu w pyszczku ryby MBF (*Mouth Brooding Fish Algorithm*);
611. Optymalizacja zapachu rekina SSO (*Shark Smell Optimisation*) [320];
612. Algorytm rekina SA (*Shark / Search Algorithm*) [321];
613. Optymalizacja ławicy tuńczyków TSO (*Tuna Swarm Optimisation*);
614. Algorytm optymalizacyjny remory (ryby) ROA (*Remora Optimisation Algorithm*);
615. Algorytm wyszukiwania roju ryb FSSA (*Fish Swarm Search Algorithm*);
616. Optymalizacja żaglicy SFO/SO (*Sailfish Optimizer*) [322], ISFO-DL (*Improved Sailfish Optimisation Algorithm with Deep Learning*), ISFO--CS (*Controller Selection*);
617. Algorytm optymalizacji roju ryb salp SSA (*Salp Swarm Algorithm*) [323], [324], chaotyczny CSSA [325]; hybryda algorytmów SSO oraz GSA-HSSOGSA (*Hybrid Sperm Swarm Optimisation SSO Gravitational search Algorithm GSA*); ISSA (*Improved Salp Swarm Algorithm*) z HDPM (*Highly Disruptive Polynomial Mutation*), ESSA (*Enhanced* SSA);

618. Optymalizacja piranii czerwonej (Natterera) RPO (*Red Piranha Optimisation*);
619. Optymallizacja par barakud PBSO (*Pair Barracuda Swarm Optimisation*);
620. Algorytm żerowania płaszczki MRFO (*Manta Ray Foraging Optimisation*), MGL-MRFO (*Improved Manta Ray Foraging Optimisation)*;
621. Optymalizator rekina Czyngis-chana GKS/GKSO (*Genghis Khan Shark Optimizer*);
622. Algorytm sułtanki (barwenowate)/ryba-koza YSG (*Yellow Saddle Goatfish, łac. Parupeneus cyclostomus*) [326];
623. Optymalizacja żerowania stętwa (węgorza elektrycznego) EEFO (*Electric Eel Foraging Optimisation*);
624. Optymalizator białego rekina WSO (*White Shark Optimizer*);
625. Optymalizator konika morskiego *SHO (Sea Horse Optimizer)*;
626. Optymalizator giętkozęba cętkowanego CCO (*Cuckoo Catfish Optimizer*) [327];
627. Algorytm zachowania barweny (żółtej) ryby kozy*Parupeneus cyclostomus* YSGA (*Yellow Saddle Goatfish behawior Algorithm*);
628. Optymalizacja polowania strzelczykowatych *Toxotidae* AHO (*Archerfish Hunting Optimizer*);
629. Algorytm optymalizacji suma CAO (*Catfish Optimization Algorithm*);
630. Algorytm pyszczaków MBF (*Mouth Breeding Fish Algorithm*);
631. Optymalizacja zapachu rekina SSO (*Shark Smell Optimisation*);

Owady

632. Algorytm nartnika (owada) WSA (*Water Strider Algorithm*);
633. Dynamiczny algorytm nartnika (owada) DWSA (*Dynamic Water Strider Algorithm*);
634. Algorytmy karaluchowe CSO (*Cockroach Swarm Optimisation*) [11], [328], [329], [330], CSA (*Cockroach Swarm Algorithm*), SCCSO (*Stochastic Constriction Cockroach Swarm Optimisation*), optymalizacja inwazją karaluchów RIO (*Roach Infestation Optimisation / Algorithm/ Problem*) [331], MCSO (*Modified CSO*), ulepszony ICSO (*Improved CSO*);
635. Algorytm trzmiela B/BA/BB (*Bumblebees Algorithm*);
636. Algorytmy świetlikowe (świetlika) FA/FF/FFA (*Firefly Algorithm*) [183], [332], HFA (*Hybrid Firefly Algorithm*) [333];
637. Dyskretmy algorytmy świetlikowy DFA (*Discrete Firefly Algorithm*), ewolucyjny dyskretmy algorytmy świetlikowy (świetlika) EDFA (*Evolutionary Discrete Firefly Algorithm*);

638. Gaussowski chaotyczny algorytm świetlików CGFA (*Gauss Map-Based Chaotic Firefly Algorithm*);
639. Algorytm muszki/muszek owocowej FOA (*Fruit Fly Optimisation Algorithm*) [334], JS-FOA (J*oint Search Strategies* FOA);
640. Algorytm ważki DA (*Dragonfly Algorithm*) [335], binarny algorytm BDA (*Binary Dragonfly Algorithm*), ulepszony IDA (*Improved the Dragonfly Algorithm*);
641. Adaptacyjny algorytm ważki oparty na strategii odporności i wytrzymałości ARSSDA (*Adaptive Resistance and Stamina Strategy-based Dragonfly Algorithm*);
642. Algorytm optymalizacji motyla BOA (*Butterfly Optimisation Algorithm*) [336], LQBOA (*Quadratic and Lagrange interpolation-based Butterfly Optimisation Algorithm*);
643. Optymalizator motyla BO (*Butterfly Optimizer*), Efektywny optymalizator motyla EBO (*Effective Butterfly Optimizer*);
644. Algorytm optymalizacji ćmy MFO/MOF (*Moth-Flame Optimisation Algorithm*) [337], [338], efektywna strategia znajdowania i zastępowania stagnacji MFP-SFR (*Moth-Flame Optimisation Algorithm Stagnation Finding Replacing Strategy*) [339], EMFO;
645. Optymalizacja rojem świetlików (świętojańskich) GSO (*Glowworm Swarm Optimisation*) [340];
646. Optymalizacja sztucznym motylem ABO (*Artificial Butterfly Optimisation*);
647. Sztuczna kolonia (rój) pszczół ABC (*Artificial Bee Colony*) [341], [193], [342], GGABC (*Global Guided Artificial Bee Colony*), IGGABC (*Improved Gbest Guided Artificial Bee Colony*), ASBC (*Artificial Smart Bee Colony*), IABC (*Improved Artificial Bee Colony*) [253], GABC (*Gbest-guided ABC*), chaotyczny i oparty na przeszukiwaniu sąsiedztwa algorytm CNSABC (*Chaotic and Neighborhood Search-based ABC*); bazujący na lotach Lévy'ego LF-ABC (Lévy *Flight-based ABC*), ABCDP (*Artificial Bee Colony with Dynamic Population Size*), iqABC (*Improved Quick Artificial Bee Colony*), udoskonalony algorytm optymalizacji ABC oparty na przeszukiwaniu chaotycznym [343];
648. Strategia kierowana karami oparta na algorytmie sztucznej kolonii pszczół PGBC (*Penalty Guided Strategy Based on an Artificial Bee Colony Algorithm*);
649. Algorytm kolonii pszczół OptBees (*The OptBees Algorithm*);

650. Optymalizacja roju pszczół BSO/BSOA (*Bees Swarm Optimisation Algorithm*) [344];
651. Optymalizacja sztucznym rojem pszczół ABSO (*Artificial Bee Swarm Optimisation*) [345];
652. Algorytmy pszczele BeA/BA/BeeA (*Bees Algorithms*) [346], [347];
653. Optmalizacja kolonią pszczół BCO (*Bee Colony Optimisation*) [348], chaotyczny algorytm optymalizacji różnicowej kolonii pszczół CDBCO (*Chaotic Differential* BCO), ogólna optymalizacja kolonii pszczół GBCO (*Generic Bee Colony Optimisation*);
654. Algorytm godów pszczół MBO (*Mating Bee Optimisation*);
655. Algorytm godów pszczoły miodnej HBMO (*Honey Bee Mating Optimisation*) [349], zmodyfikowany algorytm godów pszczoły miodnej MHBMO (*Modified Honey Bee Mating Optimisation*) [350], ulepszony EHBMO (*Enhanced Honey Bee Mating Optimisation*), interaktywna optymalizacja kojarzenia pszczół miodnych z wykorzystaniem chaotycznych, równoległych wektorów oceny CPVEIHBMO (*Chaotic Parallel Vector Evaluated Interactive HoneyBee Mating Optimisation*);
656. Szybkie małżeństwo w optymalizacji pszczół miodnych FMHBO (*Fast Marriage in Honey Bees Optimisation*);
657. Algorytm pszczoły miodnej HBA (*Honey Bee Algorithm*) [341];
658. Algorytm harmonijnych pszczół HBA (*Harmonic Bees Algorithm*) [351];
659. Algorytm wirtualnych pszczół VBA (*Virtual Bee/Bees Algorithm*) [352];
660. System pszczeli BS (*Bee System*) [353], [354];
661. Pszczeli ul BH (*Bee Hive/BeeHive*) [355];
662. Algorytm pszczelego ula BHA (*BeeHive Algorithm*);
663. Sztucznej pszczoły miodnej AHB (*Artificial Honey Bee*) [356];
664. Optymalizacja związków u pszczół miodnych MBO (*Marriage in Honey Bees Optimisation*);
665. Migracja królowej pszczół miodnych QHMB (*Queen Honey Bee Migration*);
666. Algorytm genetyczny wspomagany przez algorytm królowej pszczół QBGA (*Queen Bee assisted Genetic Algorithm*);
667. Ewolucja pszczół dla algorytmów genetycznych BEGA (*Bee Evolution for Genetic Algorithms*), ulepszony IBEGA (*Improved Bee Evolution for Genetic Algorithms*);
668. Algorytm genetyczny roju pszczelego BSGA (*Bee Swarm Genetic Algorithm*);

669. Algorytm społecznego żerowania pszczół miodnych HSFA/HBSF (*Honey Bee Social Foraging Algorithm*);
670. Pszczeli algorytm rutingu dla mobilnej sieci *ad hoc* BeeAdHoc;
671. Mrówczy algorytm rutingu dla mobilnej sieci *ad hoc* AntAdHoc;
672. Algorytmy mrówkowe AA (*Ant Algorithms*) [359], optymalizacja kolonią mrówek ACO (*Ant Colony Optimisation*) [360], [361], optymalizacja kolonią mrówek dla dziedzin ciągłych ACO_R (*Ant Colony Optimisation continuous domains*), wirtualny algorytm mrówkowy VAA (*Virtual Ant/Ants Algorithm*) [362], ACOR, chaotyczna, losowa optymalizacja kolonii mrówek RCACO [363], hybrydowy HACO, IACOR, LIACOR, hybrydowy algorytm ACO z iteracyjnym wyszukiwaniem lokalnym ILS (*Iterated Local Search*);
673. Algorytm lwiej mrówki ALO/ALA (*Ant Lion Optimizer/Optimisation*) [364], SHALO (*SHuffled Ant Lion Optimisation*), saALO (*self-adaptive* ALO), EALO (*Exponentially Weighted* ALO), DAALO (*Dynamic Adaptive Ant Lion Optimisation*), MALO (*Modified* ALO), IALO (Improved ALO), IALOT (*Improved* ALO *Tournament selection*);
674. Algorytmy komarze MHSA (*Mosquito Host-Seeking Algorithm*) [365];
675. Optymalizacja lotu komara MFO (*Mosquito Flying Optimisation*), V--MFO (*Variable* Flight *Mosquito Flying Optimisation*);
676. Optymalizacja roju cykad CISO (*Cicada Swarm Optimisation*) [366];
677. Optymalizacja motyla monarchy MBO (*Monarch Butterfly Optimisation*) [367];
678. Wyszukiwanie ćmy MS (*Moth search*) [368];
679. Algorytm roju ciem MSA (*Moth Swarm Algorithm, Moth Search Algorithm*);
680. Algorytm optymalizacji chrząszcza biedronki LBO (*Ladybug Beetle Optimisation Algorithm*);
681. Optymalizator *Charidotella sexpunctata* – gatunku chrząszcza z rodziny stonkowatych GTBO (*Golden Tortoise Beetle Optimizer*);
682. Algorytm optymalizacji jętki MA/MOA (*Mayfly Optimisation Algorithm*);
683. Optymalizacja kolonii termitów TCO (*Termite Colony Optimisation*);
684. Algorytm optymalizacji pasikonika (konika polnego) GOA (*Grasshopper Optimisation Algorithm/Approaches*) [369], [370];
685. Algorytm wyszukiwania bazujący na antenach chrząszcza BAS (*Beetle Antennae Search Algorithm*) [371];

686. Chaotyczna kolonia pszczół CBCA (*Chaotic Bee Colony Algorithms*);
687. Algorytm genetyczny sztucznej kolonii pszczół GABC (*Genetic Artificial Bee Colony*);
688. Optymalizator cyklu życiowego termitów TLCO (*Termite Life Cycle Optimizer*);
689. Mutualizm między mszycami a mrówkami AAM (*Aphid–Ant Mutualism*);
690. Algorytm optymalizacji żuka gnojowego DBO (*Dung Beetle Optimisation Algorithm*);
691. Algorytm rytownika pospolitego PBA (*Pity Beetle Algorithm*), ulepszenie algorytmu z modelem rozpraszania feromonów PBM-PDM (*Pheromone Dispersion Model*);
692. Algorytm wyszukiwania modliszki MSA (*Mantis Search Algorithm*);
693. Efektywny optymalizator motyla EBO-CMAR (*Effective Butterfly Optimizer z Covariance Matrix Adapted Retreat Phase*),
694. Optymalizacja roju szarańczy LSO (*Locust Swarm Optimisation*);
695. Optymalizacja rojowa kowala bezskrzydłego (*Pyrrhocoris apterus*) FSO (*Firebug Swarm Optimisation*);
696. Algorytm królowej termitów TQA (*Termite Queen Algorithm*);
697. Algorytm sztucznego ula pszczelego ABA (*Artificial Bee Hive/BeeHive*);
698. Algorytm inspirowany koloniami pszczół BCIA (*Bee Colony-Inspired Algorithm*);
699. Algorytm życia pszczół BLA (*Bees Life Algorithm*);
700. Optymalizator motyla BO (*Butterfly Optimizer*);
701. Algorytm oparty na zachowaniu świerszcza CBBE (*Cricket Behavior-Based Algorithm*);
702. Społeczne żerowanie pszczół miodnych HSF (*Honeybee Social Foraging*);
703. Optymalizacja biedronki siedmiokropki LBO (*Seven-Spot Labybird Optimization*);
704. Algorytm optymalizacji komarów MOX (*Mox Optimization Algorithm*);
705. Algorytm optymalizacji zwykłego motyla RBOA (*Regular Butterfly Optimization Algorithm*);
706. Algorytm kopca termitów TA (*Termite Hill Algorithm*);
707. Hybryda algorytmu świetlika z technikami wyszukiwania wzorców HFAPS (*Hybridized Firefly* z *Pattern Search Techniques*);
708. Przyrostowa sztuczna kolonia pszczół z wyszukiwaniem lokalnym IABC-LS (*Incremental Artificial Bee Colony with Local Search*);

Pajęczaki

709. Algorytm pająka społecznego SSA/SSO (*Social Spider Algorithm/ Optimisation*) [372];
710. Algorytm optymalizacji czarnej wdowy (pająka) BWO/BWOA (*Black Widow Optimisation Algorithm*);
711. Algorytm optymalizacji skaczącego pająka *Arachnida salticidae* JSOA (*Jumping Spider Optimisation Algorithm*);
712. Algorytm (skaczącego) pająka Portia PSA (*Portia Spider Algorithm*) [373];
713. Optymalizacja tygrzyka paskowanego SWO (*Spider Wasp Optimisation*);

Skorupiaki

714. Optymalizator kojarzenia pąkli BMO (*Barnacles Mating Optimizer*);
715. Algorytm optymalizacji kraba pustelnika HCOA (*Hermit Crab Optimisation Algorithm*);
716. Algorytm wymiany muszli kraba pustelnika HCSE (*Hermit Crab Shell Exchange*);
717. Algorytm ławicy kryla (szczętek) KH/KHA (*Krill Herd*) [374], [375], [376] zmodyfikowany MKHA [376], hybrydowy HKHA [377], [376], algorytm chaotycznego stada kryla (szczętek) (*Chaotic Krill Herd Algorithm*) [378];

Robaki

718. Algorytm optymalizacji dżdżownicy EWA/EOA (*Earthworm Optimisation Algorithm*) [379];

Bakterie

719. Algorytmy optymalizacji kolonii bakterii BCA/BCO (*Bacterial Optimisation/Bacterial Colony Optimisation /Algorithms*) [380] oraz [381]: chemotaksji kolonii bakteryjnej BCC (*Bacterial Colony Chemotaxis*), bakteryjnego żerowania BFA/BFO/BFOA (B*acterial Foraging Optimization*) [382], FABF (*Fuzzy Adaptive Bacterial Foraging*), BGAF/ BFO-GA (*Bacterial-GA Foraging*), pole potencjału bakteryjnego BPF (*Bacterial Potential Field*);
720. Algorytm chemotaksji bakterii BC (*Bacterial Chemotaxis Algorithm*) [383], [384], hybrydowy BC-PSO [385];
721. Szybki algorytm roju bakterii FBSA (*Fast Bacterial Swarming Algorithm*);
722. Algorytm optymalizacji bakterii magnetotaktycznych (MTB) BOA/ MBO (*Magnetotactic Bacteria Optimisation Algorithm*);

Wirusy

723. Przeszukiwanie kolonii wirusów VCS (*Virus Colony Search*) [386];
724. System wirusowy VS (*Viral System*) [381], [387];
725. Algorytm *wirusa* VO/VOA (*Virus Optimisation Algorithm*) [388];
726. Algorytm optymalizacji wirusowości VOA (*Virulence Optimisation Algorithm*) [389];
727. Algorytm optymalizacji koronowirusa COA (*Coronavirus Optimisation Algorithm*);
728. Algorytm optymalizacji koronowirusa COVIDOA (*COronaVIrus Disease Optimisation Algorithm*);
729. Algorytm wyszukiwania optymalizacji wirusa Ebola EOSA (*Ebola Optimisation Search Algorithm*);
730. Algorytm wyszukiwania optymalizacji wirusa Ebola oparty na odporności IEOSA (*Immunity Ebola Optimisation Search Algorithm*);

Flora (*nature-based, plant inspired algorithm, plants based*) [390]

731. Inteligentny algorytm optymalizacji kwiatów SFOA (*Smart Flower Optimisation Algorithm*);
732. Optymalizator mniszka lekarskiego DO (*Dandelion Optimizer*);
733. Algorytm sztucznych alg (glonów) AAA (*Artificial Algae Algorithm*) [391];
734. Algorytm drzewiasty TBA (*Tree-based Algorithm*);
735. Algorytm rośliny z rodziny rosiczkowatych „aldrowandy pęcherzykowatej" WWPA (*Waterwheel Plant Algorithm*);
736. Algorytm flory AFOA (*Arificial Flora Optimisation Algorithm*);
737. Chaotyczny algorytm optymalizacji wzrosu trzmieliny (rośliny) CTOA (*Chaotic-based Tumbleweed Optimisation Algorithm*);
738. Algorytm optymalizacji wzrosu trzmieliny (rośliny) TOA (*Tumbleweed Optimisation Algorithm*);
739. Algorytm optymalizacji śluzowcami SMA/SMO (S*lime Mould Optimisation/Algorithm/Life Cycle*) [394], ulepszony ESMA (*Enhanced Slime Mould Algorithm*);
740. Algorytm optymalizacji lasu FOA (*Forest Optimisation Algorithm*);
741. Algorytm naturalnej regeneracji lasu NFRA (*Natural Forest Regeneration Algorithm*);
742. Algorytm hybrydowej optymalizacji ryżu HRO (*Hybrid Rice Optimisation Algorithm*);
743. Algorytm nasion drzewa TSA (*Tree-Seed Algorithm*);

744. Algorytm optymalizacji słonecznika SFO/SFA (*Sunflower/Sun Flower optimisation Algorithm*);
745. Optymalizacja konkurencji między roślinami PCO (*Plant Competition Optimization*);
746. Algorytm optymalizacji topoli POA (*Poplar Optimisation Algorithm*);
747. Optymalizacji wiktorii królewskiej (*Victoria amazonica (Poepp.) Sowerby*) VOA (*Victoria Amazonica Optimization*);

Algorytmy rozprzestrzeniania się roślin (*plant propagation algorithms*) [390]

748. Algorytm propagacji roślin PPA (*Plant Propagation Algorithm*);
749. Algorytm pola Paddy'ego PFA (*Paddy Field Algorithm)*;
750. Algorytm rozłóg i korzeni RRA (*Runner-Root Algorithm*) [392];
751. Algorytm dorastania sadzonek SGA (*Saplings Growing Up Algorithm*);
752. Algorytm mechanizmu samoobrony rośliny SDMA (*Self-Defense Mechanism Of The Plants Algorithm*);
753. Algorytm wzrostu drzewa TGA (*Tree Growth Algorithm*) [393];
754. Algorytm kolonizacji/ inwazji chwastów IWO/IWOA (*Invasive Weed Optimisation Algorithm/ Weed Colonization Optimisation*) [395] [396], CIWO (*Chaotic Invasive Weed Optimisation Algorithm*);
755. Algorytm optymalizacji kolonii chwastów FWCO (*Foraging Weed Colony Optimisation algorithm*);
756. Algorytm zapylania kwiatów FPA (*Flower Pollination Algorithm*) [397], ulepszony algorytm zapylania kwiatów oparty na strategii mutacji IFPA (*Improved Flower Pollination Algorithm based on mutation strategy*);
757. Algorytm rozrostu truskawek SPPA (*Strawberry Plant Propagation Algorithm*);
758. Algorytm rozsiewania roślin oparty na nasionach SbPPA (*Seed-based Plant Propagation Algorithm*);

Algorytmy szukania światła (*light foraging algorithms*) [390]

759. Algorytm symulacji wzrostu roślin PGSA (*Plant Growth Simulation Algorithm*);
760. Algorytm optymalizacji wzrostu roślin PGO (*Plant Growth Optimisation Algorithm*);

Algorytmy tworzenia korzeni (*root foraging algorithms*) [390]

761. Algorytm wzrostu korzeni RGA (*Root Growth Algorithm*);
762. Algorytm optymalizujący wzrost korzeni RGO (*Root Growth Optimiser Algorithm*);

763. Algorytm optymalizacji sztucznej rośliny (drzewa) APO/APOA (*Artificial Plant Optimisation Algorithm*) [398];

Algorytmy bazujące na biologii BBs *Biology-based*

764. Algorytm skupiska osłonnic (strunowce) TSA/TuSA (*Tunicate Swarm Algorithm*), CTSA (*Chaos-enhanced tunicate swarm algorithm*);
765. Poszukiwanie (sztucznymi) meduzami JS (*Artificial Jellyfish Search*);
766. Algorytm poszukiwania meduzami JFSA (*Jelly Fish Searching Algorithm*);
767. Algorytm żeglarza portugalskiego (aretuzy) PMW (*Portuguese Man o' War*);
768. Algorytm optymalizacji mątwy CFA/OCFA (*Cuttlefish Optimisation Algorithm, Optimized CuttleFish Algorithm*) [399], [400];

Ewolucja oparta na rozmnażaniu *Breeding-based Evolution*

769. Optymalizacja chorób sztucznymi infekcjami AIDO (*Artificial Infections Disease Optimization*);
770. Algorytm optymalizacji rozmnażania bezpłciowego ARO *(Asexual Reproduction Optimisation*) [403], chaotyczna optymalizacjia CARO (*Chaotic Asexual Reproduction Optimisation*) [404];
771. Algorytm biogeograficzny BBO (*Biogeography-based Optimisation*) [101];
772. Optymalizacja ptasich godów BMO/BMA (*Bird Mating Optimisation/ Optimizer*) [288], oraz modyfikacje: RKBMO (*Random-Key Bird Mating Optimizer*), DBMO (*Discrete Bird Mating Optimizer*) [289];
773. Algorytm optymalizacji ziaren fasoli BOA (*Bean Optimization Algorithm*);
774. Algorytm optymalizacji raf koralowych CRO (*a Novel / Coral Reefs Optimisation Algorithm*) [413], ICRO (*Interactive Coral Reefs Optimisation*);
775. Algorytm komórek dendrytycznych DCA (*Dendritic Cells Algorithm*);
776. Ewolucja różnicowa DE (*Differential Evolution/ Evaluation*) [81], [82], MDE (*Modified Differential Evolution Algorithm*), FADE (*Fuzzy Adaptive Differential Evolution Algorithm*), algorytm quasi-opozycyjnej ewolucji różnicowej lotów Lévy'ego QODELFA (*Quasi-Oppositional Differential Evolution Lévy Flights Algorithm*) [83], utrzymujący różnorodność algorytm ewolucji różnicowej z wieloma próbami [84] DMDE (*Diversity-Maintained Multi-Trial Differential Evolution Algorithm*), ewolucja różnicowa oparta na opozycji ODE (*Opposition-based Differential Evolution*) [85], SHADE, L-SHADE/LSHADE (*linear population size reduction*), LSHADE-EpSin, JADE, SaDE (*self-adaptive*

DE), CJADE (*chaotic-local-search-based* JADE), LSHADE-SPACMA, ulepszony algorytm różnicowy EJADF; iL-SHADE, AGDE (*Adaptive Guided Differential Evolution*), EBLSHADE, ELSHADE-SPACMA, SALSHADE-cnEPSin;

777. Optymalizacja oparta na ekogeografii EBO (*Ecogeography-Based Optimization*);
778. Algorytm ewolucyjny inspirowany ekologią EEA (*Eco-Inspired Evolutionary Algorithm*);
779. Strategie ewolucyjne ES (*Evolution Strategies*) [87], CES (*classical* ES), FES (*fast* ES), CMA−ES (*Covariance Matrix Adaptation Evolution Strategy)* [88];
780. Algorytmy genetyczne GA (*Genetic Algorithms*) [79], [4], [80], GE-AGA (*Gray-Encoded Accelerating Genetic Algorithm*), PMGA (*Parallel Migrating Genetic Algorithm*), algorytm genetyczny z krzyżowaniem pierścieniowym RCGA (*Ring Crossover Genetic Algorithm*), hybrydowy kwantowy algorytm genetyczny QIGA (*Hybrid Quantum-Inspired Genetic Algorithm*), PPSOGA (*PSO-GA Algorithm*); ACO-GA (*Ant Colony Optimisation ACOz Genetic Algorihm GA*), QGA (*Quantum-inspired Genetic Algorithm*), hybrydowyz algorytmem pasikonika GA-GOA (*Hybrid Genetic-Grasshopper Algorithm*); algorytm genetyczny optymalizacji z ograniczeniami COGA (*Constraint Optimisation Genetic Algorithm*);
781. Programowanie ekspresji genowej GE/GEP (*Gene Expression Programming*);
782. Algorytm immunologiczny IA (*Immune Algorithm*);
783. Inteligencja obliczeniowa inspirowana układem odpornościowym ICI (*Immune- Inspired Computational Intelligence*);
784. Sztuczne systemy immunologiczne AIS/AIA (*Artificial Immune Systems/Algorithm*) [223], [224], [225], algorytm selekcji klonów odpornościowych ICSA (*Immune Clonal Selection Algorithm*);
785. Algorytm odpornościowy dla dwóch populacji DPIA (*Dual-Population Immune Algorithm*);
786. Ulepszony genetyczny algorytm immunologiczny IGIA (*Improved Genetic Immune Algorithm*);
787. Optymalizacja małżeństw pszczół miodnych MHBO (*Marriage In Honey Bees Optimization*);
788. Ewolucja królowej pszczół QB/QBE (*Queen Bee Evolution*) [357], [358];
789. Algorytm superbakterii SuA (S*uperbug Algorithm*),

790. Algorytm komórek macierzystych SCA (*Stem Cells Algorithm*);
791. Optymalizacja oparta na modelach świńskiej grypy SIMBO (*Swine Influenza Models Based Optimization*);
792. Samoorganizujący się algorytm migracji SOMA (*Self-Organizing Migrating Algorithm*);
793. Optymalizacja siatki o zmiennej wielkości VMO (*Variable Mesh Optimization*);

Nature- and bio-inspired meta-heuristics/Swarm Intelligence (*inne*)

794. Algorytm sztucznych alg AAA (*Artificial Algae Algorithm*);
795. Biomimikra społecznych bakterii żerujących dla rozproszonej optymalizacji BFOA (*Biomimicry Of Social Foraging Bacteria for Distributed Optimization*);
796. Algorytmy stadne FBA (*Flocking Base Algorithms*);
797. Algorytm inspirowany żabimi wezwaniami FCA (*Frog Call Inspired Algorithm*);
798. Grupowe zachowania ucieczkowe GEB (*Group Escape Behavior*);
799. Algorytm migracji ludności PMA (*Population Migration Algorithm*);
800. Algorytm projekcji inspirowany rojem SIP (*Swarm Inspired Projection Algorithm*);

Inne algorytmy

801. Optymalizator sztucznych pierwotniaków APO (*Artificial Protozoa Optimizer*);
802. Optymalizator pijawek wysysających krew BSLO (*Blood-Sucking Leech Optimizer*);
803. Algorytm krzepnięcia ludzkiej krwi BCA (*Blood Coagulation Algorithm*);
804. Optymalizator samolubnego stada SHO (*Selfish Herd Optimizer*); chaotyczny optymalizator samolubnego stada CSHO (*Chaotic Selfish Herd Optimizer*);
805. Optymalizator oparty na średniej dla grupy GMBO (*Group Mean-based Optimizer*);
806. Algorytm drapieżników morskich MPA (*Marine Predators Algorithm*);
807. Ulepszony algorytm drapieżników morskich EMPA (*Enhanced Marine Predators Algorithm*);
808. Optymalizacja wędrówek (migracji) zwierząt AMO (*Animal Migration Optimisation*) [401];

809. Hierarchiczna optymalizacja roju HSO (*Hierarchical Swarm Optimisation*), dwupoziomowy algorytm HSO nazwany ($PS^{2Krill}O$) bazujący na modelu PSO;
810. Inteligencja sztucznego roju ASI (*Artificial Swarm Intelligence*) [402];
811. Konkurencyjny optymalizator rojowy CSO (*Competitive Swarm Optimizer*);
812. Wyszukiwanie ważone-liderów WLS (*Weighted-Leader Search*);
813. Algorytm zmutowanego przywódcy MLA (*Mutated Leader Algorithm*);
814. Optymalizacja dwuetapowa TSO (*Two Stage Optimisation*);
815. Algorytm roju hybrydowego HSA (*Hybrid Swarm Algorithm*);
816. Algorytm przeszłość teraźniejszość przyszłość PPF (*Past Present Future Algorithm*);
817. Optymalizacja wyszukiwania przejściowego TSO (*Transient Search Optimisation*);
818. Algorytm wyszukiwania przyszłości FSA (*Future Search Algorithm*);
819. Algorytm wspomagający optymalizację OBA (*Optimisation Booster Algorithm*) [405];
820. Algorytm przeszukiwania symbiotycznego SOS (*Symbiotic / Symbiosis Organisms Search*) [406], ulepszony ESOS (*Enhanced* SOS); HSOSSA (*Hybrid Symbiotic Organisms Search* z *Simulated Annealing Algorithm*), A-CSOS (*Chaos and Global Competitive Ranking-based Symbiotic Organisms Search Algorithm*);
821. Algorytm przeszukiwania symbiotycznego oparta na równowadze dopasowania i dystansu FDB–SOS (*Fitness-Distance Balance–based Symbiotic Organism Search*) [407];
822. Optymalny algorytm żerowania OFA (*Optimal Foraging Algorithm*) [408]; algorytm rozszerzony EOFA *(Extended Optimal Foraging Algorithm*) [409];
823. Algorytm ekosystemowy – optymalizacja oparta na sztucznym ekosystemie AEO (*Artificial Ecosystem-based Optimisation*) [410], [411], [412], CAEA (*Chaotic Artificial Ecosystem Algorithm*);
824. Algorytm wyszukiwania w sieciach społecznościowych SNS (*Social Network Search Algorithm*), ulepszony algorytm wyszukiwania w sieciach społecznościowych ESNS/ESNSA (*Enhanced Social Network Search Algorithm*);
825. Podejście trzypunktowe TPBA (*Three Point-based Approaches*);
826. Ulepszona złożona ewolucja losowa ISCE (*Improved Shuffled Complex Evolution*);

827. Logistyczny algorytm optymalizacji chaotycznej Rao LCROA (*Logistic Chaotic Rao Optimisation Algorithm*);
828. Ułamkowy chaotyczny zespołowy algorytm PSO FC-EPSO (*Fractional Chaotic Ensemble PSO algorithm*);
829. Optymalizacja sztucznego ekosystemu oparta na równowadze dopasowania i dystansu FDB-AEO (*Fitness-Distance Balance-based Artificial Ecosystem Optimisation*);
830. Równowaga dopasowania i dystansu oparta na adaptacyjnym zdobywaniu i dzieleniu się wiedzą FDB-AGSK (*Fitness-Distance Balance--based Adaptive Gaining-Sharing Knowledge*);
831. Algorytm optymalizacji huraganu HOA/HO (*Hurricane Optimisation Algorithm*);
832. Metoda naturalnego przetrwania NSM (*Natural Survivor Method*);
833. Algorytm oparty na odejmowaniu średniej SAA (*Subtraction-Average--based Algorithm*);
834. Algorytm optymalizacji wydajności kwantowej z kurczącym się rozkładem gaussowskim SG-QPSO (*Shrinking Gaussian Distribution Quantum Performance Optimisation Algorithm*);
835. Algorytm optymalizacji lodu matowego (szronowego) RIME (*Rime Optimisation Algorithm*), łagodzenie nierównowagi między eksploracją a eksploracją IRIME (*Imbalance* RIME), bIRIME (*binary* IRIME);
836. Współpracujący algorytm metaheurystyczny CMA (*Cooperative Metaheuristic Algorithm*);
837. Współpracujący algorytm ewolucji współbieżnej CCEA (*Cooperative Co-evolution Algorithm*);
838. Rojowy algorytm optymalizacji z dobrą/poprawną siatką GLSO (*Good Lattice Swarm Optimisation Algorithm*);
839. Hierarchiczny wieloprzywódczy algorytm sinusowo-kosinusowy HMLSCA (*Hierarchical Multi-Leadership Sine Cosine Algorithm*);
840. Hybrydowy algorytm WMA-WOA z adaptacyjną mutacją Cauchy'ego HWMWOA (*Hybrid* WMA-WOA *Algorithm with Adaptive Cauchy Mutation*);
841. Algorytm optymalizacji rynku nieruchomości REMARK (*Real Estate Market–based Optimisation Algorithm*);
842. Hybrydowy algorytm wielopopulacyjny HMPA (*Hybrid Multi-Population Algorithm*) bazujący na AEO (*Artificial Ecosystem–based Optimisation*) i HHO (*Harris Hawks Optimisation*);

843. Algorytm interakcji między drapieżnikiem (kotami), pasożytem (kukułkami) i żywicielem (wronami) PPA (*Parasitism-Predation Algorithm*);
844. Zachowania łowieckie różnych zwierząt w przyrodzie (*Prey House Mechanism*);
845. Algorytm drapieżnik-ofiara PPA/PPO (*Prey-Predator Algorithm/ Optimisation*);
846. Ewolucyjna teoria gier EGT (*Evolutionary Game Theory*), niestrukturalna ewolucyjna teoria gier U-EGT (*Unstructured Evolutive Game Theory*), niestrukturalne podejście ewolucyjnej teorii gier EGTCP (*Unstructured Evolutive Game Theory approach*),
847. Algorytm religii boskich DRA (*Divine Religions Algorithm*);
848. Algorytm Fishera i Jaikumara z adaptacyjnym wyszukiwaniem w dużym sąsiedztwie FJA-ALNS (*Fisher and Jaikumar Algorithm with Adaptive Large Neighborhood Search*);
849. Heurystyczny algorytm Dahiya-Garg DG-Alg (*Dahiya-Garg Heuristic Algorithm*);
850. Optymalizacja dwietapowa TSO (*Two-Stage Optimization*);
851. Algorytm optymalizacji wyszukiwania kolonii CSOA (*Colony Search Optimization Algorithm*);
852. Optymalizatorem pojedynczego kandydata SCO (*Single Candidate Optimisation*);
853. Algorytm współpracy zespołowej krewetki pistoletowej (*Alpheus rapa / A. rapacida*) i ryby goby (*Psilogobius mainlandi*) ShGa (*Shrimp and Goby association search Algorithm*).

Cześć z wymienionych algorytmów naśladują sposoby i metody wyszukiwania znane z przyrody (np. świata zwierząt), ale mogą mieć niewielki wkład w sam proces optymalizacji. Większość z tych „atrakcyjnie nazwanych" metod może posiadać tylko lokalną wydajności albo została przebadana tendencyjnymi metodami do weryfikacji łatwych problemów z wysokim podobieństwem między interakcjami ich komponentów [128].

Omawiane algorytmy, a zwłaszcza algorytmy rojowe mają przed sobą dobre perspektywy rozwoju, a badania nad nimi wciąż znajdują się w początkowej fazie. Istnieje wiele zagadnień, które należy starannie rozważyć np.: jak skutecznie unikać optimum lokalnego, jak skutecznie połączyć zalety różnych algorytmów optymalizacji, jak ustawić parametry algorytmu, jakie są skuteczne warunki zatrzymania iteracji [418]?

Klasycznym podejściem do testowania algorytmów optymalizacji jest analiza klasycznych funkcji testowych o znanych ekstremach np. Michalewicza, Rosenbrocka, De Jonga, Schwefela, Ackley'a, Rastringa, Easoma, Griewanka, Shuberta [286], [419], Bohachevsky'ego, Matyasa, Zakharova i Goldsteina-Price'a [420], innych funkcji [12], [421] lub z zestawu testowego Tallarda [422], zbioru 30 funkcji testowych CEC'14/CEC'17 (*Congress on Evolutionary Computation*) [423] czy 10 funkcji testowych CEC2020. Inne znane zbiory testowe to np.: CEC 2019, CEC2021.

1.4.5. Algorytm symulowanego wyżarzania SA

Algorytm symulowanego wyżarzania SA należy do rodziny algorytmów heurystycznych, pozwalających na znajdywanie rozwiązań bliskich optimom globalnym. Algorytm ten prezentuje odejście od zasady monotoniczności i w sposób losowy, zgodnie z rozkładem Boltzmana, akceptuje rozwiązania gorsze.

Idea działania algorytmu została zaczerpnięta z metalurgii, gdzie fragment metalu jest ogrzewany do temperatury, w której atomy są wzbudzane termicznie, a następnie pozostawiany jest do powolnego stygnięcia. Powolne chłodzenie umożliwia atomom obniżenie poziomu swojej energii i odnalezienie stanu o minimalnej energii – równowagi termodynamicznej (*thermal equilibrium*).

Algorytm SA jest połączeniem odpowiedniego schematu schładzania [424] z algorytmem Metropolisa (błądzenia losowego). Umożliwia on poszukiwanie optymalnego stanu układu z zadaną funkcją kryterialną (F_c), nazywaną „energią układu". Przekształcenie lokalne w układzie odpowiada ruchowi cząstki, natomiast szukane optimum globalne jest stanem układu o minimalnej energii.

W algorytmie tym dopuszcza się w początkowej fazie obliczeń przejścia ze stanu bieżącego do stanów „gorszych", co daje możliwość przeglądania większego obszaru przestrzeni rozwiązań oraz uniezależnienie się od punktu startowego, a uzyskiwane rozwiązania są na ogół bliskie punktowi optymalnemu [13], [82], [425], [426].

Efektywność procesu poszukiwania, za pomocą algorytmu symulowanego wyżarzania, rozwiązań optymalnych zależy od:

- parametrów startowych, takich jak temperatura początkowa (T_0) i początkowe prawdopodobieństwo akceptacji poszukiwanego rozwiązania (P_0);
- wektora zawierającego wartości poszukiwanych parametrów, tworzących, tzw. zbiór stanów, na który składają się, między innymi, rozmieszczenie punktów AP i stacji ST czy wartości mocy wyjściowych nadajników;
- funkcji kryterialnej (F_c) odpowiadającej „energii układu";
- strategii i szybkości obniżania w kolejnych krokach temperatury układu (r_t);
- funkcji określającej sposób generowania kolejnych rozwiązań;

– długości epoki, czyli liczby wewnętrznych iteracji dla tej samej temperatury (L_{epo});
– końcowych wartości: temperatury (T_{min}) lub prawdopodobieństwa akceptacji (P_N), stanowiących kryteria zakończenia działania algorytmu.

Algorytm symulowanego wyżarzania SA można opisać pseudokodem przedstawionym w tabeli 1.14.

Tabela 1.14. Pseudokod algorytmu SA symulowanego wyżarzania

Algorytm startuje od rozwiązania początkowego odpowiadającego temperaturze $T_0 = T_{max}$

while $T>0$ **do**
  for i=0 **to** L_{epo} do
    wybierz nowe rozwiązanie w' z sąsiedztwa w ($w' \in N(w)$)
    oblicz różnicę funkcji kryterialnej $\Delta F_c = F_c(w') - F_c(w)$
      if $\Delta F_c \leq 0$ **then**
        przyjmij $w \leftarrow w'$
      else if $\Delta F_c > 0$ **then**
        przyjmij $w \leftarrow w'$ z prawdopodobieństwem exp $(\frac{-\Delta F_c}{T})$
        if exp $\left(\frac{-\Delta F_c}{T}\right) > rand(0,1)$ **then**
          zaakceptuj nowe rozwiązanie $w \leftarrow w'$
        else
          Nie akceptuj nowego rozwiązania
        end if
      end if
      Zredukuj temperaturę $T \leftarrow r_t T$, gdzie r_t dobierany jest z zakresu (0,1)
    end for
  end while

Po osiągnięciu temperatury T_{min}, dalej można użyć algorytmu deterministycznego, aż do osiągnięcia optimum funkcji kryterialnej.

Początkowe prawdopodobieństwo akceptacji P_0 (tab. 0.15) gorszego rozwiązania powinno być bliskie 1. Wartość ΔF_c^{max}, czyli największą możliwą zmianę funkcji kryterialnej, można wyznaczyć, analizując z góry założone „sąsiedztwa" punktu startowego dla przyjętej funkcji kryterialnej [427].

Temperaturę początkową T_0 (tab. 0.15) można obliczyć ze wzoru [13]:

$$T_0 = \frac{\overline{\Delta F_c}^{(+)}}{ln\left(\frac{m_2}{m_2 \cdot \chi - m_1 \cdot (1-\chi)}\right)}, \tag{1.126}$$

gdzie:
m_1– liczba kroków pogarszających wynik,
m_2– liczba kroków poprawiających wynik,
$\overline{\Delta F_c}^{(+)}$– średnia wartość różnicy funkcji kryterialnej F_c przy krokach poprawiających wynik obliczeń dla algorytmu SA z parametrem T_0 =0 [13],
x – współczynnik akceptacji przejścia między stanami [13] dla $\overline{\Delta F_c}^{(+)}$.

Tabela 1.15. Parametry sterujące algorytmem SA

Parametry	Oznaczenie/wzór
Największa zmiana F_c	ΔF_c^{max}
Poziom pogorszenia F_c	δ_N
Początkowe/końcowe prawdopodobieństwo akceptacji	P_0/P_N
Początkowa/końcowa temperatura T_0/T_{min}	$T_0 = \frac{-max\Delta E}{lnP_0}$, $T_{min} = \frac{-\delta_N}{lnP_N}$,
Zastosowana strategia schładzania – geometryczna T_i	$T_i = r_t \cdot T_{i-1}, r_t \in (0,1)$,
Długość epoki – liczba powtórzeń dla tej samej temperatury	L_{epo}
Liczba iteracji N_{iter}	$N_{iter} = \frac{log_{10}T_{min} - log_{10}T_0}{log_{10}r_t}$

Wybór temperatury początkowej T_0 układu w istotny sposób wpływa na skuteczność algorytmu SA – zbyt duża jej wartość, może znacznie wydłużyć czas obliczeń, natomiast zbyt mała może nie doprowadzić do znalezienia optimum globalnego. Ważnym jest także wybór strategii oraz szybkości schładzania układu. Jedną ze strategii jest geometryczny schemat schładzania (tab. 1.15) o współczynniku zbieżności (r_t) dobieranym z przedziału [0,85, 1) [424].

Temperatura końcowa (T_{min}) powinna być bliska zeru i jest uznawana za właściwe kryterium stopu algorytmu SA [13]. Zatrzymanie algorytmu SA może być także warunkowane liczbą iteracji (N_{iter}) lub czasem przeprowadzenia obliczeń (T_{opt}). Parametry odpowiedzialne za działanie algorytmu

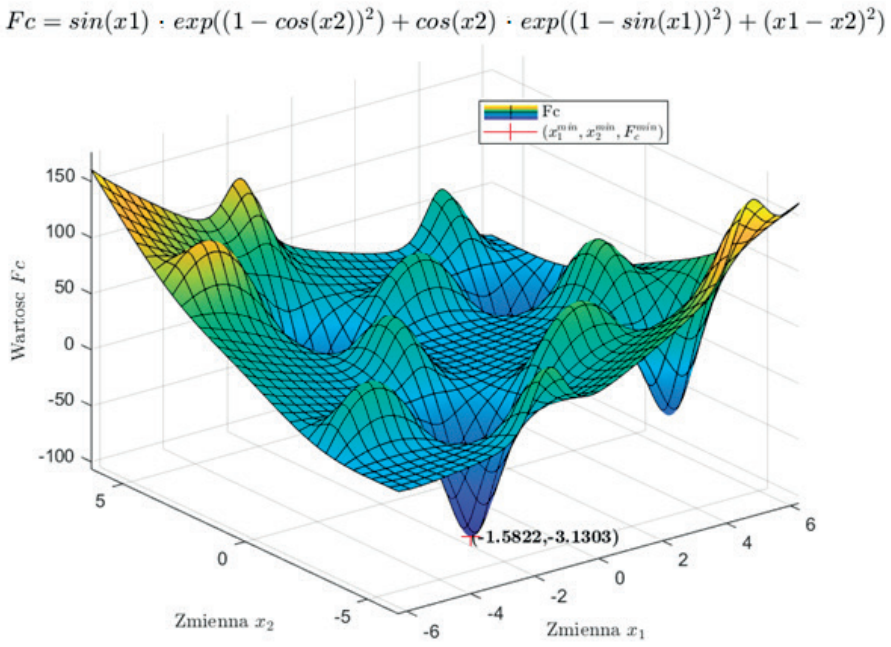
SA wybierane są zazwyczaj drogą prób i błędów związanych z otrzymaną wartością funkcji kryterialnej (dokładnością rozwiązania) oraz czasem, po którym otrzymuje się rozwiązania [428].

Warto wspomnieć, że algorytm symulowanego wyżarzania SA, jako jeden z nielicznych, posiada dowód zbieżności globalnej [13].

Przykład 4. Rozważona została dwuwymiarowa funkcja testowa *(Bird Function*) wyrażona wzorem:

$$F_c = \sin(x_1)\exp\left(\left(1-cos(x_2)\right)^2\right) + cos(x_2)\exp\left(\left(\left(1-sin(x_1)\right)^2\right)\right) + (x_1 - x_2)^2. \quad (1.127)$$

Funkcja ta posiada dwa globalne minima zlokalizowane w: $x^{min} = \begin{bmatrix}4{,}7010\\3{,}1529\end{bmatrix}$ oraz $x^{min} = \begin{bmatrix}-1{,}5821\\-3{,}1302\end{bmatrix}$, dla których wartość funkcji (1.27) osiąga wartość $F_c{}^{min} = -106{,}7645$. W rozważanym przypadku z zastosowaniem SA (tab. 1.16) osiągnięto jedno rozwiązanie. W przeciwieństwie do części algorytmów deterministycznych nie „utknięto" w jednym z ekstremów lokalnych, ale wynikiem rozwiązania jest jedno z rozwiązań globalnych (rys. 1.10).



Rysunek 1.10. Rozwiązanie zadania z przykładu 4 (1.111) z zastosowaniem algorytmu SA

Tabela 1.16. Przykład zastosowania algorytmu SA

```
clear; close all; clc;
format long
rng shuffle
x0= [2 2 ];
lb=[-2*pi -2*pi];
ub=[2*pi 2*pi];
options =
optimoptions('simulannealbnd','PlotFcns',{@saplotbestx,@saplotbestf,@saplot
x,@saplotf,@saplottemperature,@saplotstopping},'MaxIterations',300,'Display
','iter','InitialTemperature',[200 200]);
% Zdefiniowanie testowej funkcji "Bird Function"
Fc = @(x1,x2) sin(x1).*exp((1-cos(x2)).^2)+cos(x2).*exp((1-sin(x1)).^2)
+(x1-x2).^2;
[xmin,fmin,exitFlag,output] =
simulannealbnd(@(data)Fc(data(1),data(2)),x0,lb,ub,options);
fig=figure(2);
set(fig,'defaultTextInterpreter','latex');
fsurf(Fc ,[lb(1) ub(1) lb(2) ub(2)]);
hold on
plot3(xmin(1),xmin(2),fmin,'Marker', '+','MarkerSize',10,'Color', 'r');
przestext=0.1;
text(xmin(1)-przestext,xmin(2)-przestext,fmin,strcat('\bf (' ,
num2str(xmin(1)),',', num2str(xmin(2)),')'));
xlabel('Zmienna $x_1$'); ylabel('Zmienna $x_2$');
zlabel('Wartosc $Fc$');
title('$Fc=sin(x1)*exp((1-cos(x2))^2)+cos(x2)*exp((1-sin(x1))^2)+
(x1-x2)^2$', 'FontSize',14);
legend({'Fc','$(x_{1}^{min},
x_{2}^{min},F_{c}^{min})$'},'interpreter','latex');
```

1.4.6. Wybrane algorytmy rojowe i ich charakterystyka

Obecnie znanych jest co najmniej kilkaset algorytmów optymalizacji OPA wykorzystujących zjawiska obserwowane w świecie zwierząt i roślin (podrozdział 1.4.4), zjawiska i prawa fizyki czy ogólnie zbiorowej inteligencji życia na ziemi [429], [430].

W systemach rojowych występuje kilka dynamicznych mechanizmów odpowiedzialnych za koordynację i wykorzystujących mechanizmy samoorganizacji oraz stygmergii.

Mechanizm samoorganizacji polega na budowie wzorców globalnej struktury na podstawie interakcji niższego poziomu, takich jak [11]:

- sprzężenia zwrotne lokalnych interakcji, a wśród nich sprzężenia pozytywne, wynikające z realizacji prostych reguł behawioralnych, które powodują wzmocnienie aktywności oraz sprzężenia negatywne, prowadzące do stabilizacji zbiorowego wzorca;
- wzmocnienie losowych fluktuacji przez pozytywne informacje;
- wielokrotne interakcje prowadzące do podobnych reakcji deterministycznych i wykształcenia trwałej zbiorowości.

Stygmergia to druga z zasad zbiorowości stadnej dotycząca mechanizmu pośredniej koordynacji zachowania jednostek dzięki modyfikacji środowiska. Każda akcja wykonana przez osobnika z roju zostawia ślad w środowisku, wpływający na sposób ponownego jej wykonania.

Mechanizm stygmergii można wyrazić w trzech zasadach, głoszących, że:

- praca stanowi behawioralną reakcję na stan otoczenia i nie zależy od specyfiki jednostki;
- modyfikacja środowiska wykorzystywana zostaje jako pamięć stanu pracy;
- identyczne zasady zachowania mogą, w zależności od stanu środowiska, tworzyć różne wzorce.

Na ogólny schemat algorytmu stadnego składają się trzy elementy:

- zainicjowanie początkowej populacji rozwiązań oraz ocenę jej jakości za pomocą funkcji kryterialnej F_c lub funkcji dopasowania F_{fit}, którą można wyrazić za pomocą funkcji kryterialnej F_c, jako:

$$F_{fit} = \begin{cases} \dfrac{1}{1+F_c}, \text{dla } F_c \geq 0 \\ 1 + |F_c|, \text{dla } F_c < 0 \end{cases} \tag{1.128}$$

- sprawdzenie czy spełnione jest kryterium stopu lub nie jest spełnione to następuje:

- identyfikacja sąsiedztwa bieżących rozwiązań,
- wybór najlepszych rozwiązań z sąsiedztwa,
- akceptacja rozwiązań kandydujących lub ich odrzucenie,
- utworzenie nowej populacji rozwiązań;

– przedstawienie najlepszych rozwiązań.

Dla potrzeb niniejszej monografii, dla scenariuszy opisanych w trzecim i czwartym rozdziale, w procesie poszukiwania ekstremum funkcji kryterialnej (kryterialnych), posłużono się kilkoma wybranymi algorytmami rojowymi [431], dokładnie opisanymi w kolejnych podrozdziałach.

Dla tego typu algorytmów kluczowe jest zachowanie właściwej równowagi między eksploracją a eksploatacją (tab. 1.17).

Tabela 1.17. Analiza wybranych metaheurystyk [11]

Algorytm	**Mechanizm eksploracji**	**Mechanizm eksploatacji**
Optymalizacji rojem cząstek (ptasi) PSO (*Particle Swarm Optimisation*)	Dostrojenie do optymalnego ekstremum – komponent społeczny	Przeszukanie dużych obszarów – komponent poznawczy
Pszczeli BeA (*Bee Algorithm*)	Rekrutowanie pszczół do przeszukiwania (sąsiedztw)	Losowe przeszukiwanie
Świetlika FA (*Firefly Algorithm*)	Ruch świetlika na podstawie jasności i atrakcyjnosci	Losowy ruch jaśniejszych świetlików
Kukułki CS (*Cuckoo Search*)	Lokalne ruchu	Loty Lévy'ego

1.4.7. Algorytm pszczeli BeA

Pierwszy z rozważanych algorytmów rojowych, **algorytm pszczeli BeA** umożliwia przeszukiwanie przestrzeni rozwiązań zadania optymalizacji o zadanej funkcji kryterialnej F_c w sposób naśladujący zdobywanie nektaru przez rój pszczół.

Prace nad tym algorytmem były prowadzone już na przełomie lat 70-tych i 80-tych XX wieku [349], [350]. Obecnie istnieje duża liczba modyfikacji i rozszerzeń jego podstawowej wersji, w tym, między innymi, zaprezentowany w dalszej części rozdziału, algorytm sztucznej kolonii pszczół ABC.

W algorytmie BeA, początkowa grupa pszczół zwiadowców N_{sb} zostaje wysłana w celu przeszukania w sposób losowy zadanego obszaru i odna-

lezienia miejsc zasobnych w kwiaty. Po powrocie do ula pszczoły, podczas tańca, przekazują informację innym pszczołom o swoim najlepszym odkryciu. Podczas tańca waggle (*waggle dance*) pszczoły przekazują informację o lokalizacji znalezionego źródła pożywienia, czyli kierunku oraz odległości od ula oraz jakości źródła (nektaru). Do najlepszych miejsc (źródeł pożywienia) są wysłane pozostałe pszczoły i to one rozpoczynają zbiór nektaru. Im źródło nektaru jest zasobniejsze, tym więcej pszczół o nim się dowiaduje.

Algorytm pszczeli BeA składa się z czterech następujących po sobie etapów:

1. Etap losowego wyboru rozwiązań startowych w liczbie N_{sb}, równej liczbie pszczół zwiadowców (*scout bees*).
2. Etap obliczenia, dla N_{sb} rozwiązań, wartości funkcji kryterialnej F_c i na tej podstawie ograniczenie obszaru poszukiwań do zbioru (*selected sites*) $N_{ss} \leq N_{sb}$ rozwiązań początkowych.
3. Etap poszukiwania rozwiązania, spełniającego zadane kryterium stopu, którym w szczególności może być maksymalna liczba iteracji N_{iter}^{max}. Procedura poszukiwania optymalnego rozwiązania przebiega według następującego schematu:

– spośród N_{sb} miejsc, odwiedzonych przez pszczoły zwiadowców, należy wybrać (*elite site*) $N_{es} \leq N_{ss} \leq N_{sb}$ najlepszych;
– do tych N_{es} najlepszych miejsc należy wysłać N_{esb} pszczół i wyznaczyć w tych miejscach wartości funkcji celu F_c;
– wskazać miejsce, tzn. najlepszą pszczołę, dla której funkcja celu F_c przyjmuje lokalnie wartość optymalną;
– pozostałe pszczoły przypisać do dalszych losowych poszukiwań najlepszego rozwiązania zadania optymalizacji;
– dla każdej z pszczół, poszukujących nowych – lepszych rozwiązań, należy obliczać za pomocą wzoru (1.112) odpowiadające im wartości funkcji dopasowania.

4. Etap zakończenia poszukiwań rozwiązania zadania optymalizacji w momencie, gdy osiągnięte zostanie kryterium stopu, tzn. jeżeli, na przykład, liczba iteracji osiągnie założoną wartość N_{iter}^{max}. Wówczas, spośród wszystkich dotychczasowych rozwiązań, należy wskazać rozwiązanie najlepsze.

Fragment kodu algorytmu BeA który został zaprezentowany w projekcie Yarpiz [429] (tab. 1.17).

Tabela 1.18. Przykład implementacji algorytmu BeA [429]

```
% Copyright (c) 2015, Mostapha Kalami Heris & Yarpiz
% (www.yarpiz.com)
% All rights reserved. Please read the "LICENSE" file for license
% terms.
% Project Code: YPEA115
% Project Title: Implementation of Standard Bees Algorithm in
% MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
% Developer: Mostapha Kalami Heris (Member of Yarpiz Team)
% Cite as:
% Mostapha Kalami Heris, Bees Algorithm (BeA) in MATLAB (URL:
% https://yarpiz.com/315/ypea115-bees-algorithm), Yarpiz, 2015.
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
clear; close all; clc;
CostFunction = @(x) Birdfcn(x);        % przykładowa funkcja kosztu
% Parametery algorytmu BeA
nScoutBee = 30;                         % Liczba pszczół zwiadowców
nSelectedSite = round(0.5*nScoutBee);   % Liczba selekcjonowanych
                                        % miejsc
nEliteSite = round(0.4*nSelectedSite);  % Liczba selekcjonowanych
                                        % elitarnych miejsc
nSelectedSiteBee = round(0.5*nScoutBee);% Liczba zrekrutowanych
                                        % pszczół do wybranych miejsc
nEliteSiteBee = 2*nSelectedSiteBee;     % Liczba zrekrutowanych
                                        % pszczół do elitarnych miejsc
r = 0.1*(VarMax-VarMin);                % Promień sąsiedztwa
rdamp = 0.95;                          % Współczynnik ograniczenia promienia
                                        % sąsiedztwa
% Inicjalizacja macierzy pszczół
bee = repmat(empty_bee, nScoutBee, 1);
% Utworzenie nowego rozwiązania
for i = 1:nScoutBee
    bee(i).Position = unifrnd(VarMin, VarMax, VarSize);
    bee(i).Cost = CostFunction(bee(i).Position);
end
% Sortowanie
[~, SortOrder] = sort([bee.Cost]);
bee = bee(SortOrder);
% Uaktualnienie najlepszego rozwiązania
BestSol = bee(1);
% Macierz z wartościami dla najlepszych rozwiązań
BestCost = zeros(MaxIt, 1);
% Głowna pętla algorytmu
for it = 1:MaxIt
    % Elitarne miejsca
    for i = 1:nEliteSite
        bestnewbee.Cost = inf;
        for j = 1:nEliteSiteBee
            newbee.Position = PerformBeeDance(bee(i).Position, r);
            newbee.Cost = CostFunction(newbee.Position);
            if newbee.Cost<bestnewbee.Cost
                bestnewbee = newbee;
            end
```

```
        end
        if bestnewbee.Cost<bee(i).Cost
            bee(i) = bestnewbee;
        end
    end
% Wybrane nieelitarne miejsca
    for i = nEliteSite+1:nSelectedSite
        bestnewbee.Cost = inf;
        for j = 1:nSelectedSiteBee
            newbee.Position = PerformBeeDance(bee(i).Position, r);
            newbee.Cost = CostFunction(newbee.Position);
            if newbee.Cost<bestnewbee.Cost
                bestnewbee = newbee;
            end
        end
        if bestnewbee.Cost<bee(i).Cost
            bee(i) = bestnewbee;
        end
    end
    % Niewybrane miejsca
    for i = nSelectedSite+1:nScoutBee
        bee(i).Position = unifrnd(VarMin, VarMax, VarSize);
        bee(i).Cost = CostFunction(bee(i).Position);
    end
    % Sortowanie
    [~, SortOrder] = sort([bee.Cost]);
    bee = bee(SortOrder);
    % Uaktualnienie najlepszego rozwiązania
    BestSol = bee(1);
    % Zachowanie najlepszego rozwiązania
    BestCost(it) = BestSol.Cost;
    % Wyświetlenie informacji o iteracjach
    disp(['Iteration ' num2str(it) ':Best Cost='num2str(BestCost(it))]);
    % Ograniczenie promienia sąsiedztwa
    r = r*rdamp;
    % Wymagana funkcja
    % PerformBeeDance () [429]
```

1.4.8. Algorytm sztucznej kolonii pszczół ABC

Algorytm **sztucznej kolonii pszczół ABC**, inspirowany inteligentnym zachowaniem pszczół miodnych, został po raz pierwszy zaimplementowany przez D. Karaboga [341].

W algorytmie tym, w populacji pszczół miodnych wyróżnia się trzy typy pszczół: zwiadowców, obserwatorów oraz robotnice.

W algorytmie ABC „sztuczne pszczoły" poruszają się w trójwymiarowej przestrzeni poszukiwań, a pszczoły robotnice lub obserwatorzy wybierają źródło na podstawie własnych doświadczeń lub informacji pochodzących od innych pszczół z roju. Z upływem czasu sztuczne pszczoły zaczynają odnajdywać źródła z różną ilością nektaru. Jeżeli odnalezione zostanie bardziej wydajne źródło nektaru, zostaje ono zapamiętane, a informacja o poprzednich (gorszych) źródłach zostaje usunięta.

Algorytm ABC łączy zarówno lokalne, jak i globalne metody poszukiwania, balansując między procesem eksploracji i eksploatacji. Parametrami kontrolnymi w tym algorytmie są: rozmiar kolonii pszczół (N_{pop}) oraz maksymalna liczba iteracji (N_{iter}^{max}) [193].

Pseudokod algorytmu ABC opisują następujące trzy etapy:

1. Etap losowego wyznaczenie N_s początkowych źródeł nektaru $\overrightarrow{x_i} \in (1, \dots, N_s)$, dla pszczół zbierających (robotnic).
2. Etap prowadzenia w kolejnych iteracjach, aż do spełnienia warunku stopu (osiągnięcia (N_{iter}^{max}), następujących działań:

– wysłanie pszczół pracujących (zbierających nektar) do zapamiętanych miejsc pożywienia oraz wyznaczenie ilości nektaru,
– obliczenie wartości prawdopodobieństw p_i dla „zalecanych miejsc z pożywieniem", na podstawie których są uaktualnione lokalizacje preferowane przez pszczoły obserwatorki:

$$p_i = \frac{F_{fit}(\overrightarrow{x_i})}{\sum_{j=1}^{N_s} F_{fit}(\overrightarrow{x_j})}, \tag{1.129}$$

– wysłanie pszczół obserwatorek do miejsc w pobliżu wybranych źródeł pożywienia i wyznaczenie ilości nektaru w tych miejscach:

$$\begin{aligned} v_{ij} &= x_{ij} + (rand[1,-1]) \cdot (x_{ij} - x_{kj}), \\ k &= \lfloor (rand[0,1] \cdot N_s) \rfloor + 1, j \epsilon [1, \dots, N_D], \end{aligned} \tag{1.130}$$

gdzie: N_0 – wymiar problemu.

– zaprzestanie eksploatacji źródeł porzuconych,

– wysłanie pszczół zwiadowców w celu odkrycia w sposób losowy nowych źródeł pożywienia (nektaru), a nowe położenie i-tej pszczoły w przestrzeni poszukiwań $[x_j^{min}, x_j^{max}]$ wynosi:

$$x_{ij}^{min} = x_j^{min} + (x_j^{max} - x_j^{min}) \cdot \text{rand}[0,1], \tag{1.131}$$

– zapamiętanie najlepszego znalezionego do tej pory źródła pożywienia.

3. Etap wyboru i prezentacja najlepszego rozwiązania zadania optymalizacji.

Kod algorytmu ABC został zaprezentowany w projekcie Yarpiz [342] (tab. 1.18).

Tabela 1.18. Przykład implementacji algorytmu ABC [342]

```
% Copyright (c) 2015, Mostapha Kalami Heris & Yarpiz
% (www.yarpiz.com)
% All rights reserved. Please read the "LICENSE" file for license
% terms.
% Project Code: YPEA114
% Project Title: Implementation of Artificial Bee Colony in MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
% Developer: Mostapha Kalami Heris (Member of Yarpiz Team)
% Cite as:
% Mostapha Kalami Heris, Artificial Bee Colony in MATLAB (URL:
% https://yarpiz.com/297/ypea114-artificial-bee-colony), Yarpiz,
% 2015.
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
clear; close all; clc;
CostFunction = @(x) Birdfcn(x);       % przykładowa funkcja kosztu
% Definicja problemu
% Ustawienia algorytmu ABC
MaxIt = 200;              % Maksymalna liczba iteracji
nPop = 100;               % Rozmiar populacji (rozmiar kolonii)
nOnlooker = nPop;         % Liczba pszczół obserwatorów
L = round(0.6*nVar*nPop); % Parametr limitu porzucenia (limit
                          % próbny)
a = 1;                    % Górna granica współczynnika
                          % przyspieszenia
% Inicjalizacja
% Pusta struktura “pszczół”
% Inicjaliacj tablicy populacji
```

```
pop = repmat(empty_bee, nPop, 1);
% Zainicjalizowanie najlepszego znalezionego rozwiązania
% Utworzenie początkowej populacji
for i = 1:nPop
    pop(i).Position = unifrnd(VarMin, VarMax, VarSize);
    pop(i).Cost = CostFunction(pop(i).Position);
    if pop(i).Cost <= BestSol.Cost
        BestSol = pop(i);
    end
end
% Licznik porzuceń/rezygnacji
% Tablica przechowująca najlepsze wartości kosztów
% Pętla główna algorytmu ABC
for it = 1:MaxIt
    % Zrekrutowane pszczoły
    for i = 1:nPop
        % Wybieranie losowo k, nie równe i
        K = [1:i-1 i+1:nPop];
        k = K(randi([1 numel(K)]));
        % Zdefiniowanie współczynnika przyspieszenia
        phi = a*unifrnd(-1, +1, VarSize);
        % Nowa pozycja pszczół
        newbee.Position      =      pop(i).Position+phi.*(pop(i).Position-
pop(k).Position);
        % Zastosowanie ograniczeń
        newbee.Position = max(newbee.Position, VarMin);
        newbee.Position = min(newbee.Position, VarMax);
        % Ewaluacja/ocena
        newbee.Cost = CostFunction(newbee.Position);
        % Porównanie
        if newbee.Cost <= pop(i).Cost
            pop(i) = newbee;
        else
            C(i) = C(i)+1;
        end

    end
    % Obliczanie wartości dopasowania i prawdopodobieństwa selekcji
    MeanCost = mean([pop.Cost]);
    for i = 1:nPop
        F(i) = exp(-pop(i).Cost/MeanCost); % Konwersja kosztów na
                                           % dopasowanie
    end
    P = F/sum(F);
    % Pszczoły obserwatorzy
    for m = 1:nOnlooker
        % Wybranie miejsca “źródłowego”
```

```
        i = RouletteWheelSelection(P);
        % Wybranie losowo k, nie równe i
        K = [1:i-1 i+1:nPop];
        k = K(randi([1 numel(K)]));
        % Zdefiniowanie współczynnika przyspieszenia
        phi = a*unifrnd(-1, +1, VarSize);
        % Nowa ppozycja pszczoły
        newbee.Position      =      pop(i).Position+phi.*(pop(i).Position-
pop(k).Position);
        % Zastosowanie ograniczeń
        % Ewaluacja/ocena
        newbee.Cost = CostFunction(newbee.Position);
        % Porównanie
        if newbee.Cost <= pop(i).Cost
            pop(i) = newbee;
        else
            C(i) = C(i) + 1;
        end

    end
    % Pszczoły zwiadowcze
    for i = 1:nPop
        if C(i) >= L
            pop(i).Position = unifrnd(VarMin, VarMax, VarSize);
            pop(i).Cost = CostFunction(pop(i).Position);
            C(i) = 0;
        end
    end
% Aktualizacja najlepszego rozwiązanie, jakie kiedykolwiek
% wyszukano
    for i = 1:nPop
        if pop(i).Cost <= BestSol.Cost
            BestSol = pop(i);
        end
    end
% Najlepsza wartośc funkcji kosztu, jaką kiedykolwiek znaleziono
    BestCost(it) = BestSol.Cost;
    % Wyświetlanie informacji o iteracji
end
% Wynki
% Wymagana funkcja RouletteWheelSelection()
function i = RouletteWheelSelection(P)
    r = rand;
    C = cumsum(P);
    i = find(r <= C, 1, 'first');
end
```

1.4.9. Algorytm optymalizacji rojem cząstek PSO

Algorytm optymalizacji rojem cząstek PSO, zwany też **algorytmem ptasim**, został zaproponowany w 1995 roku przez J. Kennedy'ego i R. Eberharta [275], [276]. Algorytm ten bazuje na tak zwanych zachowaniach całej populacji, czyli położeniach poszczególnych osobników (cząstek), ocenianych za pomocą funkcji kryterialnej F_c lub funkcji dopasowania F_{fit}. W ramach populacji cząstki komunikują się między sobą i dzielą informacjami o swoich zachowaniach. Każda z nich w m-tym kroku działania algorytmu ma określone położenie x^m oraz kierunek i przyrost zmiany tego położenia, nazywany [428] prędkością v^m (*velocity*) [12]. Cząstki, po odnalezieniu lepszego rozwiązania (zachowania), przemieszczają się do nowych miejsc – położeń, zmieniając przy tym prędkość poszukiwań rozwiązania optymalnego. Każda cząstka ma wiedzę o swoich sąsiadach, tzn. pamięta nie tylko swoje najlepsze położenia, ale także położenia swoich sąsiadów, a także wartości funkcji F_c (F_{fit}) dla wszystkich swoich położeń i położeń sąsiadów.

Algorytm PSO opiera swoje działanie na pięciu etapach [430]:

1. Losowej inicjalizacji położenia i prędkości startowych cząstek.
2. Ocenie położenia cząstek za pomocą funkcji F_c (F_{fit}).
3. Porównaniu zachowania każdej cząstki z jej i sąsiadów najlepszym dotychczasowym zachowaniem i wyłonieniu lidera roju.
4. Uaktualnieniu prędkości każdej cząstki w kolejnym *m*–tym kroku iteracji (*pseudo time step*), zgodnie ze wzorem:

$$v_i^m = \omega v_i^{m-1} + c_1 r_1 (p_i^{m-1} - x_i^{m-1}) + c_2 r_2 (p_d^{m-1} - x_i^{m-1}), \tag{1.132}$$

gdzie:

v_i^m – prędkość *i*-tej cząstki w *m*-tym kroku,
ω – współczynnik bezwładności algorytmu,
c_1 – waga określająca, tak zwaną „świadomość roju",
c_2 – waga, tak zwanego „myślenia społecznego",
r_1, r_2 – liczby losowe z przedziału [0, 1],
p_i^{m-1} – najlepsze, dotychczas znalezione przez danego osobnika, rozwiązanie,
p_d^{m-1} – najlepsze rozwiązanie znalezione przez cały rój.

5. Uaktualnieniu w *m*-tym kroku położenia każdej z cząstek w roju, według wzoru:

$$x_i^m = x_i^{m-1} + v_i^m. \tag{1.133}$$

Rozwiązanie zadania z **przykładu 4** (1.114) z użyciem funkcji *particleswarm* z *Global Optimisation Toolbox* zaimplementowana w środowisku Matlab [431] (tab.1.19).

Tabela 1.20. Przykład zastosowania algorytmu PSO

```
clear; close all; clc;
rng default
nvars = 2;
CostFunction = @(x) Birdfcn(x);     % przykładowa funkcja kosztu
lb = [-2*pi,-2*pi];
ub = [ 2*pi, 2*pi];
options = optimoptions('particleswarm','SwarmSize',100,'MaxIterations',
1000,'SelfAdjustmentWeight', 2.0,'SocialAdjustmentWeight', 2.0);
[x,fval,exitflag,output] =
particleswarm(CostFunction,nvars,lb,ub,options)
% Opymalizowana funkcja kosztu/ funkcja kryterialna
function z = Birdfcn(x)
   z=sin(x(1))*exp((1-cos(x(2)))^2)+ cos(x(2))*exp((1-sin(x(1)))^2)+
(x(1)- x(2))^2;
end
```

Kod algorytmu PSO został zaprezentowany w projekcie Yarpiz [432] (tab. 1.20)

Tabela 1.21. Przykład implementacji algorytmu PSO [432]

```
% Copyright (c) 2015, Mostapha Kalami Heris & Yarpiz
% (www.yarpiz.com)
% All rights reserved. Please read the "LICENSE" file for license
% terms [435].
% Project Code: YPEA102
% Project Title: Implementation of Particle Swarm Optimisation in
% MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
% Developer: Mostapha Kalami Heris (Member of Yarpiz Team)
% Cite as:
% Mostapha Kalami Heris, Particle Swarm Optimisation in MATLAB
% (URL: https://yarpiz.com/50/ypea102-particle-swarm-Optimisation), %
Yarpiz, 2015.
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
% CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED % TO,
THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
% PARTICULAR PURPOSE
```

```
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
% CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, % OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT % OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
% BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
% WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
% OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
% ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
clear; close all; clc;
% Definicja problemu
CostFunction = @(x) Sphere(x);                %Funkcja kosztów
% Inne przkładowe – zaimplementowane funkcje w środowisku Matlab
% @(x) rastriginsfcn(x); @(x) Michalewicz(x); @(x) Birdfcn(x);
nVar = 10;            % Liczba zmiennych decyzyjnych
VarSize = [1 nVar];   % Rozmiar macierzy zmiennych decyzyjnych
VarMin = -10;         % Dolne ogranicznie zmiennych
VarMax = 10;          % Górne ogranicznie zmiennych
% Parametry algorytmu PSO
MaxIt = 1000;     % Makstmalna liczba itracji
nPop = 100;       % Rozmiar populacji (rozmiar roju)
% Parametry algorytmu PSO
w = 1;            % waga “bezwładosci” (Inertia Weight)
wdamp = 0.99;     % waga współczynnika tłumienia (Inertia Weight
                  % Damping Ratio)
c1 = 1.5;         % Indywidualny współczynnik uczenia się
c2 = 2.0;         % Globalny wspóczynnik uczenia się
% współczynnik ograniczeń
% phi1 = 2.05;
% phi2 = 2.05;
% phi = phi1+phi2;
% chi = 2/(phi-2+sqrt(phi^2-4*phi));
% w = chi;        % waga bezwładności
% wdamp = 1;      % waga współczynnika tłumienia (Inertia Weight
                  % Damping Ratio)
% c1 = chi*phi1;  % Indywidualny współczynnik uczenia się
% c2 = chi*phi2;  % Globalny wspóczynnik uczenia się
% Limity prędkości
VelMax = 0.1*(VarMax-VarMin);
VelMin = -VelMax;
% Initializcja
empty_particle.Position = [];
```

```
empty_particle.Cost = [];
empty_particle.Velocity = [];
empty_particle.Best.Position = [];
empty_particle.Best.Cost = [];
particle = repmat(empty_particle, nPop, 1);
GlobalBest.Cost = inf;
for i = 1:nPop
    % Inicjalizacja pozycji
    particle(i).Position = unifrnd(VarMin, VarMax, VarSize);
    % Inicjalizacja prędkości
    particle(i).Velocity = zeros(VarSize);
    % Ewaluacja
    particle(i).Cost = CostFunction(particle(i).Position);
    % Aktualizacja najlepszego wyniku osobistego
    particle(i).Best.Position = particle(i).Position;
    particle(i).Best.Cost = particle(i).Cost;
    % Aktualizacja najlepszego wyniku globalnego
    if particle(i).Best.Cost<GlobalBest.Cost
        GlobalBest = particle(i).Best;
    end
end
BestCost = zeros(MaxIt, 1);
% Pętla główna algorytmu PSO
for it = 1:MaxIt
    for i = 1:nPop
        % Aktualizacja prędkości
        particle(i).Velocity = w*particle(i).Velocity ...
            +c1*rand(VarSize).*(particle(i).Best.Position-
particle(i).Position) ...            +c2*rand(VarSize).
*(GlobalBest.Position-particle(i).Position);
        % Zastosowanie limitów prędkości
        particle(i).Velocity = max(particle(i).Velocity, VelMin);
        particle(i).Velocity = min(particle(i).Velocity, VelMax);
        % Aktualizacja pozycji
        particle(i).Position = particle(i).Position +
particle(i).Velocity;
        % Efekt lustrzanego odbicia prędkości (Velocity Mirror
        % Effect)
        IsOutside = (particle(i).Position<VarMin | particle(i).
Position>VarMax);
        particle(i).Velocity(IsOutside) = -
particle(i).Velocity(IsOutside);
        % Zastosowanie limitów prędkości
        particle(i).Position = max(particle(i).Position, VarMin);
        particle(i).Position = min(particle(i).Position, VarMax);
        % Ewaluacja
        particle(i).Cost = CostFunction(particle(i).Position);
```

```
        % Aktualizacja najlepszego wyniku osobistego
        if particle(i).Cost<particle(i).Best.Cost
            particle(i).Best.Position = particle(i).Position;
            particle(i).Best.Cost = particle(i).Cost;
            % Aktualizacja najlepszego wyniku globalnego
            if particle(i).Best.Cost<GlobalBest.Cost
                GlobalBest = particle(i).Best;
            end
        end
    end
    BestCost(it) = GlobalBest.Cost;
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);
    w = w*wdamp;
end
BestSol = GlobalBest;
% Wyniki końcowe
figure;
semilogy(BestCost, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;

% Zastosowana przykładowa/testowa funkcja celu
function z = Sphere(x)
    z = sum(x.^2);
end
```

1.4.10. Algorytm świetlika FA

W pochodzącym z 2007 roku **algorytmie świetlika FA** [433], opracowanym przez Xin-She Yanga z Uniwersytetu Cambridge, rozwiązanie zadania optymalizacji oparto na różnicy intensywności światła, która jest proporcjonalna do wartości funkcji kryterialnej F_c. Każdy jaśniejszy świetlik przyciąga do siebie pozostałe osobniki, co pozwala na zintensyfikowanie, a zatem bardziej skuteczne badanie przestrzeni przeszukiwań.

Algorytm świetlika FA opiera się na trzech założeniach:

- niezależności od, tak zwanej „płci" osobnika – oznaczającej, że wszystkie świetliki mogą się wzajemnie przyciągać i są równie atrakcyjne,
- atrakcyjności osobnika, która jest proporcjonalna do jasności świecenia i maleje wraz z odległością między świetlikami – oznaczającej, że jeżeli wszystkie świetliki są tak samo atrakcyjne, to poruszają się wówczas w sposób losowy,
- intensywności świecenia osobnika, określonej przez wartość funkcji kryterialnej F_c.

Ruch świetlika, czyli jego położenie w kolejnym m-tym kroku działania algorytmu FA, określa się za pomocą terminu eksploatacji – podczas, której świetlik stara się zbliżyć do najintensywniej świecącego osobnika oraz terminu eksploracji – czyli losowego błądzenia świetlika po obszarze poszukiwań optimum funkcji kryterialnej.

Ruch świetlika, czyli jego położenie w m–tym kroku działania algorytmu FA, określa formuła [12]:

$$x_i^m = x_i^{m-1} + \beta_0 e^{-\gamma d_{ij}^2}\left(x_j^{m-1} - x_i^{m-1}\right) + \alpha_{fa}\left(rand[0,1] - \frac{1}{2}\right), \tag{1.134}$$

gdzie:

x_i^{m-1} – bieżące położenie i-tego świetlika, tzn. położenie osiągnięte w (m-1) kroku,

d_{ij} – odległość między i-tym oraz j-tym świetlikami,

β_0 – parametr określający atrakcyjność świetlika,

α_{fa} – parametr, określający losowe przemieszczenie się świetlika, przyjmujący wartości z przedziału [0, 1],

γ – współczynnik absorpcji.

Kod algorytmu FA został zaprezentowany w serwisie MATLAB Central File Exchange [332] (tab. 1.21).

Tabela 1.22. Przykład implementacji algorytmu FA [332]

```
% The Firefly Algorithm (FA) for unconstrained function
% Optimisation     %
% by Xin-She Yang (Cambridge University) @2008-2009
% Programming dates: 2008-2009, then revised and updated in Oct
% 2010 %
% References -- citation details: ---------------------------------
% (1) Xin-She Yang, Nature-Inspired Metaheuristic Algorithms,
%     Luniver Press, First Edition, (2008).
% (2) Xin-She Yang, Firefly Algorithm, Stochastic Test Functions
% and     %
%     Design Optimisation, Int. Journal of Bio-Inspired
% Computation,     %
%     vol. 2, no. 2, 78-84 (2010).
% -------- Start the Firefly Algorithm (FA) main loop -------------
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
% CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED % TO,
THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
% PARTICULAR %PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
% CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, % OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT % OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
% BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
% WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
% OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
% ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
function fa_ndim_new
format long;
n=20;                   % Rozmiar populacji (liczba świetlików)
alpha=1.0;              % Siła losowości 0-1 (wysoka)
beta0=1.0;              % Stała atrakcyjności
gamma=0.01;             % Współczynnik absorpcji
theta=0.97;                 % Współczynnik redukcji losowości theta=10^
(-5/tMax)
d=10;                   % Liczba wymiarów
tMax=500;               % Maksymalna liczba iteracji
Lb=-10*ones(1,d);       % Dolne ograniczenia/limity
Ub=10*ones(1,d);        % Górne ograniczenia/limity
% Generowanie początkowych lokalizacji n świetlików
```

```
for i=1:n,
   ns(i,:)=Lb+(Ub-Lb).*rand(1,d);         % Losowość
   Lightn(i)=cost(ns(i,:));               % Ocena funkcji celu
end
%%%%%%%%%%%%%%%% Rozpoczęcie iteracji (pętla główna)
%%%%%%%%%%%%%%%%%%%%%%
for k=1:tMax,
 alpha=alpha*theta;     % Zmniejszenie alpha o współczynnik theta
 scale=abs(Ub-Lb);      % Skala problemu optymalizacji
% Dwie pętle nad wszystkimi n świetlikami
for i=1:n,
    for j=1:n,
      % Ocena obiektywnych wartości bieżących rozwiązań
      Lightn(i)=cost(ns(i,:));           % Wywołanie funkcji celu
      % Aktualizacja ruchów
      if Lightn(i)>=Lightn(j),           % Jaśniejszy/bardziej
                                         % atrakcyjny
         r=sqrt(sum((ns(i,:)-ns(j,:)).^2));
         beta=beta0*exp(-gamma*r.^2);    % Atrakcyjność
         steps=alpha.*(rand(1,d)-0.5).*scale;
       % Równanie FA dla aktualizacji wektorów pozycji
            ns(i,:)=ns(i,:)+beta*(ns(j,:)-ns(i,:))+steps;
      end
   end
end
% Sprawdzenie, czy nowe rozwiązania/lokalizacje mieszczą się
% w limitach/granicach
ns=findlimits(n,ns,Lb,Ub);
% Uszeregowanie świetlików według ich intensywności światła/celów
[Lightn,Index]=sort(Lightn);
nsol_tmp=ns;
for i=1:n,
   ns(i,:)=nsol_tmp(Index(i),:);
end
% Znalezienie aktualniego najlepszego rozwiązania i wyświetlenie
% wyniku
fbest=Lightn(1), nbest=ns(1,:)
% fbestvector(k)=fbest
end
%plot(fbestvector);
% Upewnienie się, że nowe świetliki mieszczą się w
% granicach/limitach
function [ns]=findlimits(n,ns,Lb,Ub)
for i=1:n,
  nsol_tmp=ns(i,:);
  % Zastosowanie dolnej granicy
  I=nsol_tmp<Lb;  nsol_tmp(I)=Lb(I);
```

```
  % Zastosowanie górnej granicy
  J=nsol_tmp>Ub;  nsol_tmp(J)=Ub(J);
  % Zaktualizowanie nowego ruchu
  ns(i,:)=nsol_tmp;
end
% Zdefiniowanie funkcji celu lub funkcji kosztu
function z=cost(x)
% Zmodyfikowana funkcja sfery: z=sum_{i=1}^D (x_i-1)^2
z=sum((x-1).^2); % Globalne minimum fmin=0 w (1,1,...,1)
% ---------------------------------------------------------------
```

1.4.11. Algorytm kukułki CS

Jednym z nowszych algorytmów optymalizacji, bo z 2009 roku, jest **algorytm kukułki CS,** który zaproponowany został przez Xin-She Yanga i Suash Deba. W algorytmie tym naśladowane są zachowania niektórych gatunków kukułki, które wykorzystują gniazda innych ptaków do wychowywania potomstwa.

Algorytm kukułki CS można opisać, tzw. pseudokodem [11], [12], uwzględniającym:

1. Losowe wygenerowanie początkowej populacji gniazd N_{neast} i związanych z nimi kukułek.
2. Prowadzenie w kolejnych iteracjach, aż do spełnienia warunku stopu (osiągnięcia N_{iter}^{max}), następujących działań:

– losowe wybranie i–tej kukułki/gniazda i wygenerowanie nowego rozwiązania x_i^m, np. za pomocą, tzw. lotu Lévy'ego:

$$x_i^m = x_i^{m-1} + \alpha_{cs}\, s_{levy} \tag{1.135}$$

gdzie:

m – numer kroku, czyli kolejnej iteracji,

x_i^m – rozwiązanie uzyskane w m-tym kroku dla i-tej kukułki,

α_{cs} – współczynnik skali, którego wartość zależy od rozmiaru problemu,

s_{levy} – długość kroku, wyznaczona na podstawie rozkładu prawdopodobieństwa Lévy'ego [12],

– obliczenie dla rozwiązania x_i^m wartości funkcji kryterialnej F_c^i i w przypadku, gdy wartość tej funkcji jest lepsza od F_c^j, obliczonej dla j–tej kukułki, to należy zastąpić rozwiązanie dla j-tego gniazda nowym x_i^m,

– porzucenie z arbitralnie przyjętym prawdopodobieństwem p_α części gorszych rozwiązań, czyli gniazd/kukułek i zastąpienie ich nowymi,
– posortowanie uzyskanych rozwiązań i zapamiętanie oraz przekazanie najlepszego z nich do dalszych iteracji.

3. Wskazanie najlepszego rozwiązania – gniazda/kukułki, po osiągnięciu kryterium stopu, na przykład, gdy liczba iteracji przyjmie wartość N_{iter}^{max}.

Model matematyczny

Mechanizmem eksploatacyjnym algorytmu CS (a także innych algorytmów OPA oraz dużej liczby algorytmów hybrydowych) są ruchy lokalne, a mechanizmem eksploracyjnym są loty Lévy'ego, oparte na rozkładzie prawdopodobieństwa Lévy'ego wyrażonym wzorem:

$$L(s_{Levy}, \lambda) = \frac{\lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\frac{1}{{s_{Levy}}^{1+\lambda}}, \tag{1.136}$$

gdzie: Γ jest funkcją gamma.

Długość kroku w algorytmie CS może być obliczona zgodnie z algorytmem Mantenga:

$$s_{Levy} = \frac{U}{|V|^{\frac{1}{\lambda}}}, \tag{1.137}$$

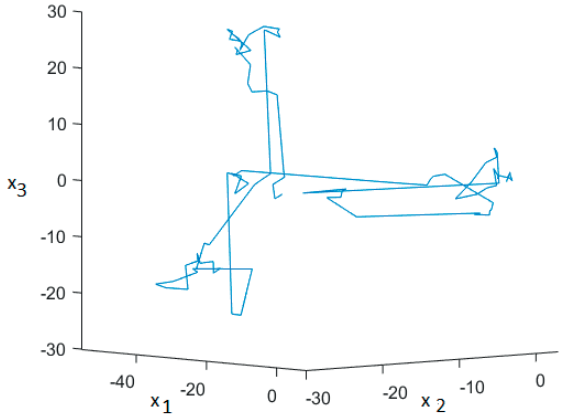
gdzie:

$$\mathrm{U} \sim \mathrm{N}(0, \sigma_u), \mathrm{V} \sim (0, \sigma_v^2), \qquad 1 < \lambda < 2, \tag{1.138}$$

i:

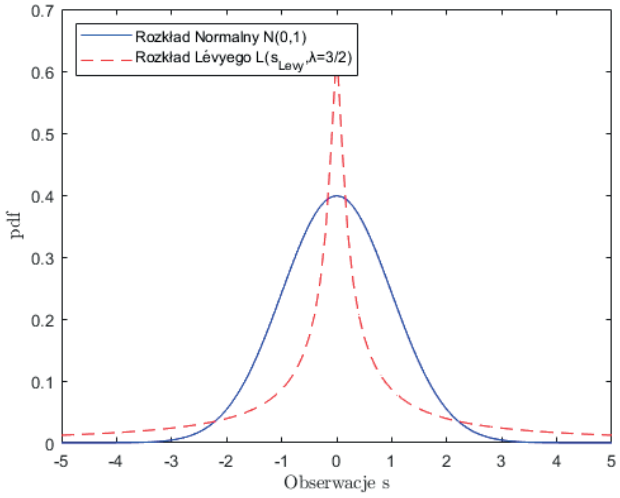
$$\sigma_v = 1, \sigma_u = \left(\frac{\Gamma(1+\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\Gamma\left[\frac{1+\lambda}{2}\right]\lambda 2^{\frac{\lambda-1}{2}}}\right)^{\frac{1}{\lambda}}. \tag{1.139}$$

Analizując kroki generowane w zastosowanym algorytmie CS, można zauważyć, że wśród dużej liczby małych kroków algorytmu od czasu do czasu występują duże skoki zwane lotami Lévy'ego (rys. 1.11), od nazwiska francuskiego matematyka Paula Pierre'a Lévy'ego. Nadają sie one dobrze do eksploracji (przeszukiwania) nieznanych dużych przestrzeni np. w sieci czujników bezprzewodowych WSN (*Wireless Sensor Networks*) [126].



Rysunek 1.11. Wizualizacja przykładowego lotu Lévy'ego (λ=1.3) w przestrzeni Eukliderowej R^3

Charakterystyczną cechą tego rozkładu Lévy'ego są długie „ogony", które występują dla dużych wartości – w przeciwieństwie do rozkładu normalnego (Gaussa) (rys. 1.12).



Rysunek 1.12. Porównanie rozkładów gęstości prawdopodobienstwa (pdf – probability density functions)

Kod algorytmu CS został zaprezentowany w serwisie MATLAB Central File Exchange [282] (tab. 1.22).

Tabela 1.22. Przykład implementacji algorytmu CS [282]

```
% Cuckoo Search (CS) algorithm by Xin-She Yang and Suash Deb      %
% Programmed by Xin-She Yang at Cambridge University              %
% Programming dates: Nov 2008 to June 2009                       %
% Last revised: Dec  2009   (simplified version for demo only)    %
% Papers -- Citation Details:
% Copyright (c) 2010, Xin-She Yang All rights reserved.
% Redistribution
% and          use in source and binary forms, with or without
% modification,
% are permitted provided that the following conditions are met:  *
% Redistributions of source code must retain the above copyright
% notice, this
%   list of conditions and the following disclaimer.  *
% Redistributions in
% binary form must reproduce the above copyright notice,this list of
% conditions and the following disclaimer in the documentation
% and/or other
% materials provided with the distribution
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
% CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED % TO,
THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
% A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
% CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
% OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT % OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
% BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
% WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
% OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
% ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
function [bestnest,fmin]=cuckoo_search(n)
   tStart = tic;
if nargin<1,
% Liczba gniazd (lub różnych rozwiązań)
n=25;
end
% Wskaźnik wykrywania obcych jaj/rozwiązań
pa=0.25;
```

```
%% Można zmieniać ten parameter w celu uzyskania lepszych wyników
% Tolerancja
Tol=1.0e-5;
% Proste granice dziedziny wyszukiwania
% Dolne granice
nd=15;
Lb=-5*ones(1,nd);
% Górne granice
Ub=5*ones(1,nd);
% Losowe rozwiązania początkowe
for i=1:n,
nest(i,:)=Lb+(Ub-Lb).*rand(size(Lb));
end
% Uzyskanie aktualnego najlepszego wyniku
fitness=10^10*ones(n,1);
[fmin,bestnest,nest,fitness]=get_best_nest(nest,nest,fitness);
N_iter=0;
% Rozpoczęcie iteracji
while (fmin>Tol),
% Generowanie nowych rozwiązań (ale z zachowaniem tych najlepszych)
     new_nest=get_cuckoos(nest,bestnest,Lb,Ub);
     [fnew,best,nest,fitness]=get_best_nest(nest,new_nest,fitness);
    % Aktualizacja licznika
      N_iter=N_iter+n;
    % Wykrywanie i losowość
      new_nest=empty_nests(nest,Lb,Ub,pa);
    % Ocena tego zestawu rozwiązań
      [fnew,best,nest,fitness]=get_best_nest(nest,new_nest,fitness);
    % Ponowna aktualizacja licznika
      N_iter=N_iter+n;
    % Odszukanie najlepszego dotychczasowego rozwiązania
    if fnew<fmin,
        fmin=fnew;
        bestnest=best;
    end
end %% Koniec iteracji
% Przetwarzanie po zakończeniu optymalizacji
% Wyświetlenie wszystkich gniazd
disp(strcat('Total number of iterations=',num2str(N_iter)));
fmin
bestnest
tEnd = toc(tStart)
figure
fsurf(fobj ,[Lb(1) Ub(1) ])
hold on
plot3(bestnest(1),bestnest(2),fmin,'-
o','Color','r','MarkerSize',10,'MarkerFaceColor','#D9FFFF')
```

```
% ---------------- Zastosowane funkcje ------------------
% Uzyskaj kukułki przez losowy spacer
function nest=get_cuckoos(nest,best,Lb,Ub)
% Loty Lévy'ego
n=size(nest,1);
% Wykładnik i współczynnik Lévy'ego
% Aby uzyskać szczegółowe informacje, równanie (2.21), strona 16
%   (rozdział 2) książki% X. S. Yang, Nature-Inspired Metaheuristic %
Algorithms, 2nd Edition, Luniver Press, (2010).
alpha=3/2; % λ =1.5
sigma=(gamma(1+alpha)*sin(pi*alpha/2)/(gamma((1+alpha)/2)*alpha*2^((alpha-
1)/2)))^(1/alpha);
for j=1:n,
    s=nest(j,:);
    % Jest to prosty sposób na wdrożenie lotów Lévy'ego
    % W przypadku standardowych spacerów losowych należy użyć step=1;
    % Loty Lévy'ego według algorytmu Mantegny
    u=randn(size(s))*sigma;
    v=randn(size(s));
    step=u./abs(v).^(1/alpha);
    % W następnym równaniu współczynnik różnicy (s-best) oznacza, że
    % w przypadku najlepszego rozwiązania pozostaje ono niezmienione.
    stepsize=0.01*step.*(s-best);
    % Współczynnik 0,01 wynika z faktu, że L/100 powinno być typowym
    % rozmiarem kroku (spacerów/lotów), gdzie L jest typową skalą
    % długości.
    % W przeciwnym razie loty Lévy'ego mogą stać się zbyt
    % agresywne/wydajne,
    % co sprawia, że nowe rozwiązania mogą nawet wyskakiwać poza
    % obszar poszukiwań
    % Faktyczne (aktualne)losowe spacery lub loty
    s=s+stepsize.*randn(size(s));
   % Zastosowanie prostych ograniczeń/limitów
   nest(j,:)=simplebounds(s,Lb,Ub);
end
% Znalezienie aktualnie najlepszego gniazda
% Wymagane funkcje oraz ich realizacje
function [fmin,best,nest,fitness]=get_best_nest(nest,newnest,fitness)
% Ocena wszystkich nowych rozwiązań
for j=1:size(nest,1),
    fnew=fobj(newnest(j,:));
    if fnew<=fitness(j),
       fitness(j)=fnew;
       nest(j,:)=newnest(j,:);
    end
end
% Znalezienie aktualnie najlepszego rozwiązania
```

```
[fmin,K]=min(fitness) ;
best=nest(K,:);
%   Zastąpienie   niektórych   gniazd   poprzez   skonstruowanie   nowych
rozwiązań/gniazd
function new_nest=empty_nests(nest,Lb,Ub,pa)
% Ułamek gorszych gniazd jest wykrywany z prawdopodobieństwem pa
n=size(nest,1);
% Odkryty lub nie – wektor stanu (status)
K=rand(size(nest))>pa;
% W prawdziwym świecie, jeśli kukułcze jajo jest bardzo podobne do % jaja
gospodarza, %to jest mniejsze prwdopodobieństwo jego odkrycia, % a zatem
wartość funkcji %dopoasowania (fitness) powinna być
% odniesiona do różnicy w rozwiązaniach.
% Dobrym pomysłem jest wykonanie spaceru w tendencyjny sposób z
% losowymi  rozmiarami kroków.
% Nowe rozwiązanie w postaci wybiórczych/selektywnych
% (biased/selective) spacerów losowych
stepsize=rand*(nest(randperm(n),:)-nest(randperm(n),:));
new_nest=nest+stepsize.*K;
for j=1:size(new_nest,1)
    s=new_nest(j,:);
  new_nest(j,:)=simplebounds(s,Lb,Ub);
end
% Zastosowanie prostych ograniczeń
function s=simplebounds(s,Lb,Ub)
  % Zastosowanie dolnych ograniczeń
  ns_tmp=s;
  I=ns_tmp<Lb;
  ns_tmp(I)=Lb(I);

  % Zastosowanie górnych ograniczeń
  J=ns_tmp>Ub;
  ns_tmp(J)=Ub(J);
  % Aktualizacja tego nowego ruchu
  s=ns_tmp;
% Następujące funkcje można zastąpić własnymi
% D-wymiarowa funkcja celu
function z=fobj(x)
% D-wymiarowa funkcja celu sum_j=1^d (u_j-1)^2.
%  z minimum w (1,1, ...., 1);
z=sum((x-1).^2);
```

1.4.12. Algorytm nietoperza BA

Algorytm nietoperza BA, zaproponowany w 2010 przez Xin-She Yanga, zainspirowany został echolokacją nietoperzy z podrzędu *Microchiroptera*.

Parametrami sterującymi pracą tego algorytmu są: głośność A_i oraz szybkość emisji r_i impulsów echolokacyjnych.

Na trzy, następujące po sobie, etapy algorytmu nietoperza BA [12] składają się:

1. Losowe rozmieszczenie na obszarze poszukiwanych rozwiązań, przyjętej od górnie liczby N_{bat} nietoperzy i przypisanie im odpowiednich szybkości v_i, a także ustalenie minimalnej f_{min} i maksymalnej f_{max} wartości częstotliwości impulsów echolokacyjnych oraz głośności A_i i szybkości r_i ich emisji.
2. Wykonywanie w ramach kolejnych iteracji, aż do spełnienia warunku stopu (osiągnięcia N_{iter}^{max}), następujących działań:
   wyznaczenie nowego rozwiązania/położenia x_i^{m+1} i–tego nietoperza:

$$x_i^{m+1} = x_i^m + v_i^m + (x_i^m - x_*)[f_{min} + (f_{max} - f_{min})\beta_{BA}], \tag{1.140}$$

   gdzie:
   x_i^m – bieżące położenie i-tego nietoperza,
   v_i^m– szybkość i-ego nietoperza,
   β_{BA} – wartość losowana z przedziału [0,1],
   x_* – aktualnie najlepsze rozwiązanie,

– jeżeli wylosowana wartość o rozkładzie jednostajnym z przedziału [0,1] jest większa od r_i, to należy:
  - wybrać najlepsze dotychczasowe rozwiązanie,
  - wygenerować lokalne rozwiązanie w sąsiedztwie bieżącego rozwiązania,

– wygenerowanie nowego rozwiązania przez losowy lot,

– jeżeli wylosowana wartość o rozkładzie jednostajnym z przedziału [0,1] jest mniejsza od A_i i spełniona jest nierówność: $F_c(x) < F_c(x^*)$ to należy:
  - zaakceptować nowe rozwiązanie,
  - zwiększyć szybkość emisji impulsów echolokacyjnych r_i oraz zmniejszyć ich głośność A_i.

3. Odszukanie najlepszego rozwiązania x^* ocenę nietoperzy.

Tabela 1.22. Przykład implementacji algorytmu BA [236]

```
% ========================================================== %
% Files of the Matlab programs included in the book:       %
% Xin-She Yang, Nature-Inspired Metaheuristic Algorithms,  %
% Second Edition, Luniver Press, (2010).   www.luniver.com %
% ========================================================== %
% Bat-inspired algorithm for continuous Optimisation (demo)%
% Programmed by Xin-She Yang @Cambridge University 2010    %
% ---------------------------------------------------------- %
% Usage: bat_algorithm([20 1000 0.5 0.5]);                 %
% This is a simple demo version only implemented the basic
% idea of the bat algorithm without fine-tuning the parameters,
% Then, though this demo works very well, it is expected that
% this demo is much less efficient than the work reported in
% the following papers:
% (Citation details):
% 1) Yang X.-S., A new metaheuristic bat-inspired algorithm,
%    in: Nature Inspired Cooperative Strategies for Optimisation
%    (NISCO 2010) (Eds. J. R. Gonzalez et al.), Studies in
%    Computational Intelligence, Springer, vol. 284, 65-74 (2010).
% 2) Yang X.-S., Nature-Inspired Metaheuristic Algorithms,
%    Second Edition, Luniver Presss, Frome, UK. (2010).
% 3) Yang X.-S. and Gandomi A. H., Bat algorithm: A novel
%    approach for global engineering Optimisation,
%    Engineering Computations, Vol. 29, No. 5, pp. 464-483 (2012).
% ----------------------------------------------------------------------
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
% CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED % TO,
THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
% PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
% CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, % OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT % OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
% BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
% WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
% OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
% ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
function [best,fmin,N_iter]=bat_algorithm(para)
% Wyświetlenie pomocy
```

```
help bat_algorithm.m
% Parametry domyślne
if nargin<1,  para=[20 1000 0.5 0.5];  end
n=para(1);      % Rozmiar populacji nietoperzy, typowo 10 do 40
N_gen=para(2);  % Liczba pokoleń
A=para(3);      % Głośność (stała lub malejąca)
r=para(4);      % wskaźnik/częstotliwość impulsu Pulse rate (stała
                % lub malejąca)
% Ten zakres częstotliwości określa skalowanie
Qmin=0;         % minimalna wartość częstotliwości
Qmax=2;         % maksymalna wartość częstotliwości
% Paranetry iteracji
N_iter=0;       % Łączna liczba obliczeń funkcji
% Wymiar poszukiwanych zmiennych
d=10;           % Liczba wymiarów
% Dolna granica wektora
Lb=-2*ones(1,d);
% Górna granica wektora
Ub=2*ones(1,d);
% Inicjalizacja tablic
Q=zeros(n,1);   % Częstotliwość
v=zeros(n,d);   % Prędkości
% Inicjalizacja populacji/rozwiązań
for i=1:n,
  Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);
  Fitness(i)=Fun(Sol(i,:));
end
% Znajdź początkowe najlepsze rozwiązanie
[fmin,I]=min(Fitness);
best=Sol(I,:);
% =====================================================================
% Uwaga: Ponieważ jest to wersja demonstracyjna, nie
% zaimplementowano      %
% funkcj zmniejszającej głośność i zwiększającej szybkości emisji
% impulsów.%
% Zainteresowani czytelnicy mogą przeprowadzić badania
% parametryczne       %
% a także wdrażanie różnych zmian A i r itp.
% =====================================================================
% Rozpoczęcie iteracji -- Algorytm nietoperza (zasadnicza część)
for t=1:N_gen,
% Pętla nad wszystkimi nietoperzami/rozwiązaniami
        for i=1:n,
          Q(i)=Qmin+(Qmin-Qmax)*rand;
          v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);
          S(i,:)=Sol(i,:)+v(i,:);
```

```
          % Zastosuj proste ograniczenia/limity
          Sol(i,:)=simplebounds(Sol(i,:),Lb,Ub);
          % Częstotliwość impulsu
          if rand>r
          % Współczynnik 0,001 ogranicza rozmiary kroków losowych
          % spacerów
          S(i,:)=best+0.001*randn(1,d);
          end
     % Ocena nowych rozwiązań
           Fnew=Fun(S(i,:));
     % Zaktualizuj, jeśli rozwiązanie się poprawi lub nie będzie
     % zbyt głośne
           if (Fnew<=Fitness(i)) & (rand<A) ,
                Sol(i,:)=S(i,:);
                Fitness(i)=Fnew;
           end
          % Zaktualizuj bieżące najlepsze rozwiązanie
          if Fnew<=fmin,
                best=S(i,:);
                fmin=Fnew;
          end
        end
        N_iter=N_iter+n;
end
% Informacje wyjściowe/wyświetlenie
disp(['Number of evaluations: ',num2str(N_iter)]);
disp(['Best =',num2str(best),' fmin=',num2str(fmin)]);
% Zastosowanie prostych limitów/ograniczeń
function s=simplebounds(s,Lb,Ub)
  % Zastosowanie wektora dolnego ograniczenia
  ns_tmp=s;
  I=ns_tmp<Lb;
  ns_tmp(I)=Lb(I);

  % Zastosowanie wektora górnego ograniczenia
  J=ns_tmp>Ub;
  ns_tmp(J)=Ub(J);
  % Aktualizacja tego nowego ruchu
  s=ns_tmp;
%=====================================================================%
Przykładowa funkcja celu:
function z=Fun(u)
% Funkcja Sphere z fmin=0 w punkcie (0,0,...,0)
z=sum(u.^2);
```

1.4.13. Wybrane problemy optymalizacji wielokryterialnej MOO

Podczas próby sformułowania – w miarę wiernie odzwierciedlającego rzeczywistość – zadania optymalizacji może okazać się koniecznym uwzględnienie w takim zadaniu nie jednej, a kilku – niekiedy wzajemnie przeciwstawnych – funkcji kryterialnych.

Zazwyczaj otaczającą nas rzeczywistość, w tym w szczególności, na przykład, sieć WLAN standardu IEEE 802.11 z infrastrukturą, nie jesteśmy w stanie dobrze scharakteryzować tylko i wyłącznie za pomocą jakiejś jednej, wybranej właściwości, a do pełnej oceny musimy wykorzystać większy zbór–zestaw najważniejszych cech. Cechy te wraz ze wskaźnikami jakości, czyli funkcjami zależnymi od tych cech składają się określony zbiór funkcji celu. Opierając zadanie optymalizacji, nie na jednej, a na kilku funkcjach celu, problem poszukiwania optymalnego rozwiązania zostaje przeniesiony z przestrzeni parametrów, czyli liczb, do przestrzeni kryteriów, czyli funkcji [2].

Zazwyczaj w problemie optymalizacji wielokryterialnej MOO (*Multiobjective Optimisation*), poszukujemy minimum/maksimum wielu funkcji kryterialnych (celu). Na przykład, dla problemu minimalizacji z ograniczeniami zbiór równań opisujących zadanie optymalizacji może przyjąć postać następujących wyrażeń:

$$\min_x \{F_{c_1}(x), F_{c_2}(x), \dots, F_{c_K}(x)\}, \tag{1.141}$$

$$g_i(x) = 0, i = 1, \dots, m_e, \tag{1.142}$$

$$g_i(x) \leq 0, i = m_{e+1}, \dots, m_o, \tag{1.143}$$

$$x_l \leq x \leq x_u, \tag{1.144}$$

gdzie:

x – zmienna niezależna,

$F_{c1} \dots F_{cK}$ – zbiór K funkcji kryterialnych,

g_i – funkcje ograniczeń,

x_t, x_u – dolne i górne ograniczenia na wartości zmiennych niezależnych.

Istnieje szereg klasycznych sposobów, metod, technik sprowadzania problemu optymalizacji wielokryterialnej MOO do problemu optymalizacji jednokryterialnej SOO (*Single-objective Optimisation*). Jedna z takich technik jest metoda, w której poszczególne funkcje celów łączy się w jedną funkcję wyrażoną jako:

$$F_c(x) = \sum_{i=1}^{K} w_i F_{c_i}(x), \qquad \text{gdzie } w_i \in [0,1], \sum_{i=1}^{K} w_i = 1. \tag{1.145}$$

W tej metodzie mogą istnieć punkty w zbiorze Pareto, które nie zostaną znalezione, ponieważ leżą, w tzw. „zagłębieniu frontu" Pareto.

Inną metodą sprowadzenie problemu MOO do SOO jest technika oparta na funkcji odległości, w której określa się wektor popytu *y*, a sama funkcja kryterialna przyjmuje postać:

$$F_c(x) = \left(\sum_{i=1}^{K} \left| F_{c_i}(x) - y_i \right|^n \right)^{\frac{1}{n}}. \tag{1.146}$$

Dla odległości euklidesowej, we wzorze (6.18), przyjmuje się n=2.

Kolejną techniką zamiany zadania wielokryterialnego na zadanie optymalizacji jednokryterialnej jest metoda ε –ograniczeń [2], [434]. Polega ona na ustanowieniu, jednej z wielu funkcji kryterialnych, „główną" i jedyną funkcją celu, podczas gdy z pozostałych funkcji celu tworzony jest zbiór ograniczeń.

Znanych jest wiele różnych metod rozwiązywania wielokryterialnych zadań optymalizacji (np. mini–max [435]). Jednak do najważniejszych należą metody – algorytmy ewolucyjne, w tym między innymi, takie jak: VEGA (*Vector Evaluated Genetic Algorithm*) [436], HLGA (*Hajela, Lin Genetic Algorithm*), FFGA (*Fonseca, Fleming Genetic Algorithm*) [437], NPGA (*Niched-Pareto Genetic Algorithm*), SPEA (*Strong Pareto Evolutionary Algorithm*) [438], SPEA2 [439], NSGA (*Non-dominated Sorting Genetic Algorithm*) [440], NSGAII [441] (z wariantami [445]), NSGAIII [15], [443], [444], PESA–II (*Pareto Envelope-based Selection Algorithm),* Micro–GA [445], AMGA2 (*Archive-based Micro Genetic Algorithm*) czy MOEA/D (*MultiObjective Evolutionary Algorithm Based on Decomposition*).

W procesie optymalizacji wielokryterialnej stosowane są także różnego rodzaju metaheurystyki „rojowe", na przykład: MOPSO (*MultiObjective Particle Swarm Optimisation*) [446], [447], MOFA (*MultiObjective Firefly Algorithm*) [433], MOCS (*MultiObjective Cuckoo Search*) [448], MOBA (*MultiObjective Bat Algorithm*) [12] i inne [194].

Rozwiązania zadania optymalizacji wielokryterialnej można sklasyfikować, jako zdominowane i niezdominowane (Pareto optymalne) [438].

Wektor $u \in F$ (przestrzeni poszukiwań – parametrów) dominuje wektor v [12]:

$$u \prec v \text{ czyli } \forall\, i \in \{1, \ldots, N_D\}: u_i \leq v_i \wedge \exists i \in \{1, \ldots, N_D\}: u_i < v_i, \tag{1.147}$$

W zadaniu minimalizacji, rozwiązanie (x) jest zdominowane, jeżeli istnieje dopuszczalne rozwiązanie (y) nie gorsze od (x) dla wszystkich funkcji celu: F_{c1} (i=1, ... N_D), co można zapisać jako [80]:

$$F_{c_i}(y) \leq F_{c_i}(x) \; \forall \; 1 \leq i \leq N_D, \tag{1.148}$$

Jeżeli rozwiązanie nie jest zdominowane przez żadne inne rozwiązanie dopuszczalne, to nazywane jest rozwiązaniem niezdominowanym (Pareto optymalnym) [80].

Front Pareto P może być zdefiniowany, jako zbiór niezdominowanych rozwiązań, co w przestrzeni poszukiwań można wyrazić, jako:

$$\mathcal{P} = X_p = \{x \in \mathcal{F} \mid \nexists \, x' \in \mathcal{F} : \; F_c(x') \prec F_c(x)\}. \tag{1.149}$$

Wszystkie rozwiązania Paraeto optymalne mogą być ważne i dlatego powinny zostać przeanalizowane.

Wybrane algorytmy MOO

Jedną z zaawansowanych metod rozwiązywania zadań wielokryterialnych MO jest metoda **programowania celowego GAM** (*Goal Attainment Method*), w którym otrzymywane jest zadanie postaci:

$$\min_{x,\gamma} \gamma, \tag{1.150}$$

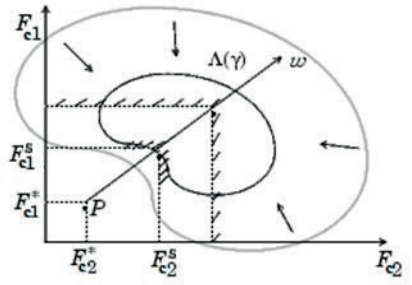
$$F_{c_1}(x) - w_1 \cdot \gamma \leq F^*_{c_1}, \tag{1.151}$$

$$F_{c_2}(x) - w_2 \cdot \gamma \leq F^*_{c_2}, \tag{1.152}$$

$$\cdots$$

$$F_{c_K}(x) - w_K \cdot \gamma \leq F^*_{c_K}, \tag{1.153}$$

gdzie: F^*_{c1}, F^*_{c1}, ..., $F^*_{c_K}$ – współrzędne wektora ($\mathbb{F}c^*$) określającego cel poszukiwań, w_1, w_2, …, w_K – współrzędne wektora (waga – *weight w*) określającego kierunek poszukiwań, γ – zmienna swobodna (*slack variable*) optymalizacji wielokryterialnej.



Rysunek 1.13. Geometryczna reprezentacja metody programowania celowego [435]

Po takim przekształceniu, poszukiwany jest punkt ze zbioru rozwiązań dopuszczalnych $(\Lambda(\gamma))$, w którym wartości kryteriów są najbliższe pewnym idealnym wartościom (punkt P na rys. 1.13) określonym wektorem $\mathbb{F}c^*$. Elementy wektora $(\mathbb{w})$ określają ważność poszczególnych kryteriów, natomiast wektor $\mathbb{F}c^*$definiuje ukierunkowania decyzyjne (dążenia). Podczas optymalizacji, zmienna swobodna γ zmienia rozmiar obszaru rozwiązań dopuszczalnych, a granice zbiegają się w unikalnym rozwiązaniu $F_{c_1}^s, F_{c_2}^s, \ldots, F_{c_K}^s$ (rys. 1.13).

W pakiecie pakiecie Matlab, metoda ta jest zaimplementowana jako funkcja programowania celowego *fgoalattain*.

Przykład 5. Rozważona została dwukryterialna optymalizacja funkcjami jednowymiarowymi zdefiniowana jako:

$$F_c = \begin{bmatrix} 2 + (x-3)^2 \\ 5 + \dfrac{x^2}{4} \end{bmatrix}. \tag{1.154}$$

Tabela 1.22. Zastosowanie optymalizacji MOO dla przykładu 5

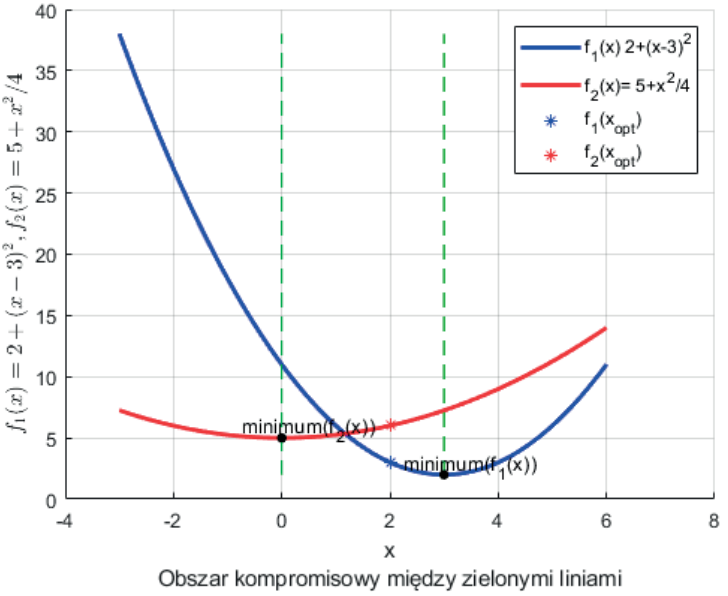
```
N = 50; % Liczba punktów do wyrysowania
x0=1;
x1 = fminsearch(@(x)pickindex(x,1),x0)
x2 = fminsearch(@(x)pickindex(x,2),x0)
d=max(x1,x2)-min(x1,x2);
x_min=min(x1,x2)-d;
x_max=max(x1,x2)+d;
t=linspace(x_min,x_max);
k = 1;
[min1,minfn1] = fminbnd(@(x)pickindex(x,k),x_min,x_max);
k = 2;
[min2,minfn2] = fminbnd(@(x)pickindex(x,k),x_min,x_max);
goal = [minfn1,minfn2];
weight = [1,1];
options =
optimoptions('fgoalattain','PlotFcn','optimplotx','Display','iter-
detailed','EqualityGoalCount',2)
[xopt,fval,attainfactor,exitflag,output,lambda] =
fgoalattain(@functions,x0,goal,weight,[],[],[],[],[],[],[],options)

figure(2)
hold on; grid on;
f=functions(t);
p1=plot(t,f(1,:),'b-','LineWidth',2);
```

```
p2=plot(t,f(2,:),'r-','LineWidth',2);
xlabel({'x';'Obszar kompromisowy między zielonymi liniami'})
ylabel('$f_1(x)= 2+(x-3)^2, f_2(x)= 5+x^2/4$','interpreter','latex')
p3=plot(xopt,fval(1),'b*');
p4=plot(xopt,fval(2),'r*');
plot([min(min1,min2),min(min1,min2)],[min(minfn1,minfn2),max(max(f(:,1)),ma
x(f(:,2)))],'g--');
plot([max(min1,min2),max(min1,min2)],[min(minfn1,minfn2),max(max(f(:,1)),ma
x(f(:,2)))],'g--');
plot([min1,min2],[minfn1,minfn2],'k.','MarkerSize',15);
deltapol=0.75;
text(min1-deltapol,minfn1+deltapol,'minimum(f_1(x))')
text(min2-deltapol,minfn2+deltapol,'minimum(f_2(x))')
legend([p1     p2     p3     p4     ],     {'f_1(x)     2+(x-3)^2','f_2(x)=
5+x^2/4','f_1(x_{opt})','f_2(x_{opt})'})

onen = 1/N;
x = zeros(N+1,1);
f = zeros(N+1,nf)   ;
x0 = 1.0;
options = optimoptions('fgoalattain','Display','off');
for r = 0:N
tw = onen*r; % między 0 a 1
weight = [tw,(1-tw)];
% Wygenerowanie frontu Pareto
[x(r+1,:),f(r+1,:)] = fgoalattain(@functions,x0,goal,weight,...
[],[],[],[],[],[],[],options);
end
figure(3)
hold on; grid on;
p1=plot(f(:,1),f(:,2),'ko');
x=round(x,4);
xopt=round(xopt,4);
index=find(x==xopt);
p2=plot(f(index,1),f(index,2),'r*');
xlabel('$f_1(x)= 2+(x-3)^2$','interpreter','latex')
ylabel('$f_2(x)= 5+x^2/4$','interpreter','latex')
legend([p1 p2], {'Front Pareto','x_{opt}'})
function f = functions(x)
f(1,:) = 2+(x-3).^2;
f(2,:) = 5+x.^2/4;
end
function z = pickindex(x,k)
z = functions(x); % Ocena/obliczenie wartości wszystkich funkcji celu
z = z(k);         % Zwrócenie wartości k-tej funkcji celu
end
```

Na rysunku 1.14 oraz 1.15 przedstawiono rozwiązanie dla przykładu 5 z zastosowaniem programowania celowego w optymalizacji wielokryterialnej.



Rysunek 1.14. Rozwiązanie programowania celowego MOO dla przykładu 5

W przypadku optymalizacji wielokryterialnej MOO w dalszej części monografii zastosowano także algorytm NSGAIII [15], [443], [444], który jest rozszerzeniem algorytmu NSGAII [435], [442], będącego ulepszoną wersją algorytmu NSGA.

W algorytmie NSGA (*Non–dominated Sorting Genetic Algorithm*) opracowanym przez Srinivasa i Deba, zastosowano autorską wersję procedury nadawania rang a także graficzną metodę porównania wyników [440]. Kilka lat później S. Deb dokonał modyfikacji w algorytmie NSGAII [441], który uważany jest za jeden z bardziej efektywnych algorytmów ewolucyjnych, który może zastać zastosowany do wyznaczania rozwiązań różnorodnych zadań optymalizacji wielokryterialnej [449].

Ze względu na efektywność wyszukiwania rozwiązania z wykorzystaniem algorytmu CS i PSO w optymalizacji jednokryterialnej [10], [11], w wybranych scenariuszach badawczych rozważona została także optymalizacja wielokryterialna MOCS [448] oraz MOPSO [446], [447].

Wielokryterialny algorytm kukułki MOCS [448] jest jednym z nowszych algorytmów, w którym w porównaniu z jednokryterialnym algorytmem kukułki CS:

- każda kukułka składa jednorazowo K jaj, i podrzuca je do losowo wybranego gniazda, gdzie każde jajko k odpowiada rozwiązaniu k-tej funkcji kryterialnej,
- każde gniazdo jest opuszczane z prawdopodobieństwem p_a i budowane jest, zgodnie z podobieństwami /różnicami w jajkach, nowe gniazdo z K jajami.

Wielokryterialny algorytm kukułki MOCS można opisać, tzw. pseudokodem [448]:

Etap 1: Zainicjowanie wszystkich zastosowanych funkcji kryterialnych, $(F^*_{c1}, \ldots F^*_{cK})$ oraz losowe wygenerowanie początkowej populacji gniazd N_{neast} – każdej z K jajkami odpowiadającej liczbie funkcji kryterialnych.

Etap 2: Wykonanie iteracji dopóki nie jest spełniony warunek stopu (np. osiągnięcia N^{max}_{iter}), w których następuje:

- obliczenie położenia i-tej kukułki np. przez lot Leviego obliczenie dla niej wartości funkcji kryterialnych, ocenę i sprawdzenie czy rozwiązanie jest Pareto optymalne,
- wybranie losowo j-te gniazdo z pośród N_{neast},
- ocenę K rozwiań dla j-tego gniazda.

- gdy nowe rozwianie dla gniazda j dominuje rozwiązanie i-tej kukułki, to należy zastąpić zawartość i-tego gniazda nowym zbiorem rozwiązań dla gniazda j,
- porzucenie części gorszych gniazd (z prawdopodobieństwem p_a) i zastąpienie je nowymi gniazdami,
- przechowanie najlepszych rozwiązań (lub gniazd z niezdominowanym zbiorem rozwiązań),
- posortowanie osobników i odszukanie bieżących najlepszych rozwiań Pareto optymalnych.

Etap 3: Jeśli spełnione zostało kryterium stopu, to wskazać najlepsze rozwiązanie i je zwizualizować.

Tabela 1.26. Przykład zastosowania optymalizacji MOO-MOCS [450]

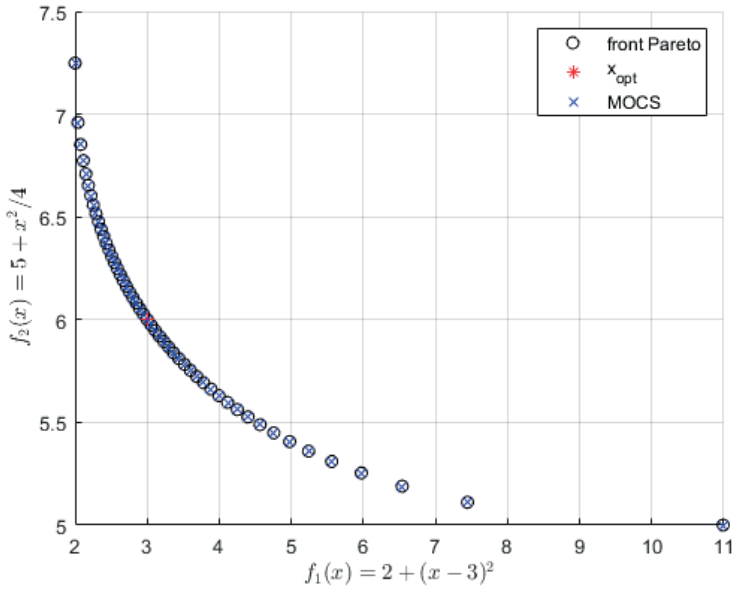
```
% Cuckoo Search (CS) algorithm by Xin-She Yang and Suash Deb     %
% Programmed by Xin-She Yang at Cambridge University              %
% Programming dates: Nov 2008 to June 2009                       %
% Last revised: Dec  2009   (simplified version for demo only)    %
% Multiobjective cuckoo search (MOCS) added in July 2012,         %
% Then, MOCS was updated in Sept 2015.                    Thanks %
% ---------------------------------------------------------------
% References -- Citation Details:
% 1) X.-S. Yang, S. Deb, Cuckoo search via Lévy flights,
% in: Proc. of World Congress on Nature & Biologically Inspired
% Computing (NaBIC 2009), December 2009, India,
% IEEE Publications, USA,  pp. 210-214 (2009).
% http://arxiv.org/PS_cache/arxiv/pdf/1003/1003.1594v1.pdf
% 2) X.-S. Yang, S. Deb, Engineering Optimisation by cuckoo search,
% Int. J. Mathematical Modelling and Numerical Optimisation,
% Vol. 1, No. 4, 330-343 (2010).
% http://arxiv.org/PS_cache/arxiv/pdf/1005/1005.2908v2.pdf
% 3) X.-S. Yang, S. Deb, Multi-objective cuckoo search for
% Design Optimisation, Computers & Operations Research,
% vol. 40, no. 6, 1616-1624 (2013).
% ---------------------------------------------------------------%
% This demo program only implements a standard version of         %
% Cuckoo Search (CS), as the Lévy flights and generation of       %
% new solutions may use slightly different methods.               %
% The pseudo code was given sequentially (select a cuckoo etc),   %
% but the implementation here uses Matlab's vector capability,    %
% which results in neater/better codes and shorter running time.  %
% This implementation is different and more efficient than the    %
% the demo code provided in the book by
%    "Yang X. S., Nature-Inspired Optimisation Algoirthms,        %
%     Elsevier Press, 2014.  "                                    %
% ---------------------------------------------------------------- %
% ================================================================ %
% Notes:                                                          %
% 1) The constraint-handling is not included in this demo code.   %
% The main idea to show how the essential steps of cuckoo search  %
% and multi-objective cuckoo search (MOCS) can be done.           %
% 2) Different implementations may lead to slightly different     %
% behavour and/or results, but there is nothing wrong with it,    %
% as it is the nature of random walks and all metaheuristics.     %
% ---------------------------------------------------------------- %
function [bestnest,fmin]=mocs_new(inp)
if nargin<1,
inp=[100 1000 0.25]; % pop_size, #iteraion, pa
end
```

```
% Liczba gniazd (lub liczebność populacji)
n=inp(1);
% Liczba iteracji/generacji
N_IterTotal=inp(2);
% Wskaźnik wykrywalności obcych jaj/rozwiązań
pa=inp(3);
d=1;   % Rozmiar (wymiarowość) problemu
% Proste dolne ograniczenia
Lb=-5*ones(1,d);
% Proste górne ograniczenia
Ub=5*ones(1,d);
% Liczba kryteriów (funkcji kryterialnych)
m=2;
%% Inicjalizacja (zainicjowanie) populacji
for i=1:n,
   Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);
   f(i,1:m) = obj_funs(Sol(i,:), m);
end
% Przechowywanie wartości dopasowania lub wartości funkcji celu
f_new=f;
% Sortowanie zainicjalizowanej populacji
x=[Sol f];  % połączone w jedno wejście
% Sortowanie rozwiązań nie zdowminowanych dla początkowej populacji
Sorted=solutions_sorting(x, m,d);
% Rozkład na rozwiązania, kondycję, rangę i odległości
nest=Sorted(:,1:d);
f=Sorted(:,(d+1):(d+m));
RnD=Sorted(:,(d+m+1):end);
% Rozpoczęcie iteracji
for t=1:N_IterTotal,
% Generowanie nowych rozwiązań (ale z zachowaniem tych najlepszych)
     new_nest=get_cuckoos(nest,nest(1,:), Lb,Ub);
     % new_nest=nest;
     % Wykrywanie i wprowadzenie losowości
     new_nest=empty_nests(nest,Lb,Ub,pa) ;

     % Ocena tego zestawu rozwiązań
      for i=1:n,
      % Ocena wartości dopasowania/funkcji nowej populacji
        f_new(i, 1:m) = obj_funs(new_nest(i,1:d),m);

        if (f_new(i,1:m) <= f(i,1:m)),
            f(i,1:m)=f_new(i,1:m);
            nest(i,:)=new_nest(i,:);
        end
     % Zaktualizuj bieżący najlepszy wynik (zapisany w pierwszym
```

```
     % wierszu)
        if (f_new(i,1:m) <= f(1,1:m)),
            nest(1,1:d) = new_nest(i,1:d);
            f(1,:)=f_new(i,:);
        end
      end
% Połączona populacja składa się zarówno ze starych, jak i nowych
% rozwiązań%% Zatem całkowita liczba rozwiązań dla sortowania
% wynosi 2*n
% Bardzo ważne jest połączenie obu populacji, w przeciwnym razie
% wyniki mogą wyglądać dziwnie i będą bardzo nieefektywne. !
       X(1:n,:)=[new_nest f_new];    % Połączenie nowych rozwiązań
       X((n+1):(2*n),:)=[nest f];    % Połączenie starych rozwiązań
       Sorted=solutions_sorting(X, m, d);
       % Wybór n rozwiązań z połączonej populacji 2*n rozwiązań
       new_Sol=Select_pop(Sorted, m, d, n);
       % Zdekomponowanie posortowanych rozwiązań na rozwiązania,
       % dopasowanie i ranking
       nest=new_Sol(:,1:d);     % Posortowane rozwiązania/zmienne
       f=new_Sol(:,(d+1):(d+m));% osortowane wartości funkcji celów
       RnD=new_Sol(:,(d+m+1):end);% Posortowane rangi i odległości
% Wyświetlanie przy każdych 100 iteracjach
   if ~mod(t,100),
     disp(strcat('Iterations t=',num2str(t)));
     plot(f(:, 1), f(:, 2),'rs','MarkerSize',3);
     f(:, 1)
     f(:, 2)
%    axis([0 10 -0.8 10]);
     xlabel('f_1'); ylabel('f_2');
     drawnow;
   end
end % Zakończenie iteracji
% Wszytskie wymagne funkcje potrzebne do uruchomienia przykładu- % %
mocs_new  %
% function nest=get_cuckoos(nest,best,Lb,Ub)
% function new_nest=empty_nests(nest,Lb,Ub,pa)
% function s=simplebounds(s,Lb,Ub)
% function f = obj_funs(x, m)
% function new_Sol = Select_pop(nest, m, ndim, npop)
% function sorted_x = solutions_sorting(x, m, ndim)
% cześc z nich przedstawiono w tabeli 1.21 w przykładzie
% zastosowania CS-- %
```



Rysunek 1.15. Rozwiązanieoptymalizacji MOO dla przykładu 5 dla programowania celowego

Na rysunku 1.15 przedstawiono rozwiązanie dla przykładu 5 z zastosowaniem programowania celowego [451] w optymalizacji wielokryterialnej, frontu Pareto [452] oraz optymalizacji z wykorzystaniem algorytmu MOCS [450]. Można zauważyć, że otrzymano dokładnie taki sam front Pareto (rysunek 1.15) z wykorzystaniem programowania celowego, jak i algorytmu MOCS. Wyszczególnione rozwiązanie (rys. 1.15) otrzymano z zasosowaniem programu z tabeli 1.24, w którym ustawiono równe wagi dla obu kryteriów, a jako cel (goal) przyjęto minima tych funkcji.

Wielokryterialny algorytm ptasi MOPSO (Multi-Objective Particle Swarm Optimisation) zaproponowany w 2002 roku przez: C.A. Coello Coello, C. I M. Salazar Lechuga, M. [449] jest wielokryterialną wersją algorytmu PSO, w której w celu określenia kierunku lotu cząstki zastosowano pomysł dominacji w sensie Pareto.

Wszystkie niezdominowane cząstki w roju (wektory wartości funkcji kryterialnych), gromadzone są w globalnej podgrupie zwanej repozytorium, a każda cząstka wybiera swój najlepszy „cel" spośród członków tego repozytorium do przeprowadzenia własnego lotu, stosując probabilistyczne zasady dominacji.

Analogicznie jak w algorytmie ptasim PSO [11], [448], cząstki roju w algorytmie MOPSO dzielą się informacjami i poruszają się w kierunku globalnie najlepszych cząstek oraz w kierunku zapamiętanej swojej najlepszej cząstce.

1.5. Podsumowanie

Ponieważ rozwiązanie zadania optymalizacji w przypadku zastosowania nieliniowych funkcji kryterialnych może nastręczać wiele problemów, dlatego rozważono różne algorytmy optymalizacji, dla kilku zdefiniowanych zadań optymalizacji.

Warto zauważyć, że algorytmy inspirowane przyrodą, a zwłaszcza najnowsze algorytmy rojowe są efektywnym narzędziem w optymalizacji globalnej różnych zadań optymalizacji.

W trzecim i czwartym rozdziale niniejszej monografii, w scenariuszach testowych przeprowadzono porównanie rozwiązań uzyskanych z użyciem wybranych algorytmów optymalizacji, dla zadań bazujących na funkcjach kryterialnych zdefiniowanych w rozdziale drugim.

2. PLANOWANIE I OPTYMALIZACJA SIECI WLAN STANDARDU IEEE 802.11 Z INFRASTRUKTURĄ

W obecnych czasach obserwuje się wzrost znaczenia lokalnych systemów i sieci telekomunikacyjnych, które mogą pracować zarówno w terenie otwartym, w warunkach gęstej zabudowy miejskiej, jak i we wnętrzach budynków. W ostatnim z wymienionych środowisk coraz większą popularność zyskują lokalne sieci bezprzewodowe WLAN (*Wireless Local Area Network*), charakteryzujące się łatwą i szybką instalacją, wygodnym użytkowaniem, ciągle doskonalonym i taniejącym sprzętem i oprogramowaniem, prostotą konfiguracyjną oraz dostępem do nielicencjonowanych pasm ISM (*Industry, Science, Medicine*) i UNII (*Unlicensed National Information Infrastructure*).

2.1. Wstęp

Uniezależnienie transmisji danych we wnętrzach budynków od mediów przewodowych pozwoliło na lepsze zagospodarowanie przestrzeni biurowych, pomieszczeń laboratoryjnych jednostek naukowych i dydaktycznych, powierzchni dużych sklepów, magazynów, dworców czy lotnisk itp. obiektów.

Istnieje szereg standardów telekomunikacyjnych, które wykorzystują nielicencjonowane pasma radiowe. W pasmach ISM/UNII działają systemy takich standardów takich jak: IEEE 802.11, HomeRF, Bluetooth/802.15.1, IEEE 802.15.4/ ZigBee, WirelessHART (*Highway Addressable Remote Transducer Protocol*), RFID (*Radio-Frequency Identification*), WDCT (*Worldwide Digital Cordless Telecommunications*), DECT (*Digital Enhanced Cordless Telephony*), WiMAX (*Worldwide Interoperability for Microwave Access*) czy ANT+ (*Advanced Network Tools*).

Wiele z urządzeń opartych na wyżej wymienionych standardach lub działających w wymienionych pasmach to nie tylko komputery osobiste i osprzęt peryferyjny pracujący na ich rzecz (myszki, klawiatury, prezentery), ale także: telefony komórkowe, tablety PC, netbooki, smartfony, e-booki,

palmptopy (PDA), radia WiFi, głośniki bezprzewodowe, odbiorniki telewizyjne DLNA (*Digital Living Network Alliance*), kamery bezprzewodowe dla systemów telewizji przemysłowej i monitoringu – CCTV (*Closed-Circuit TeleVision*), bezprzewodowe modemy PLC (*Programmable Logic Controller*), elektroniczne nianie z czujnikami ruchu/oddechu/kamerą wideo, czytniki kodów kreskowych – kolektory danych/inwentaryzatory, urządzenia sterujące drogą radiową RPV (*Remotely Piloted Vehicle*) zabawkami, miniaturowymi samolotami/dronami (Wi-fli, Wi-Spi), kuchenki mikrofalowe, pasywne czujniki podczerwieni PIR (*Passive Infra Red*), urządzenia diatermii mikrofalowej czy lotnicze i meteorologiczne systemy radarowe.

Ciągle rosnąca popularność lokalnych komputerowych sieci bezprzewodowych WLAN standardu IEEE 802.11, spowodowała wzrost liczby punktów dostępu AP (*Access Point*) dostępu do zasobów sieci transmisji danych i w konsekwencji wzrost poziomu mocy zakłóceń interferencyjnych. Ponadto w ostatnich latach pojawiło się szereg nowych urządzeń i systemów, pracujących w paśmie ISM 2.4 GHz, stanowiących dodatkowe źródło zakłóceń sieci WLAN.

Wzrost interferencji w paśmie ISM 2.4 GHz w znaczący sposób wpłynął na rozwój nowych metod planowania sieci WLAN oraz rozwiązań sprzętowych zarządzania pasmem częstotliwości – np. CCA (*Cisco CleanAir*). W tym kontekście bardzo ważnym zadaniem, stawianym przed projektantami obecnych i przyszłych sieci WLAN, jest wybór liczby punktów dostępu AP oraz ich wzajemnego rozmieszczenia czy wreszcie przypisanie im odpowiednich kanałów radiowych oraz mocy nadawczych, które wpływają na zasięg oraz na osiągane przepustowości sieci.

W planowaniu sieci WLAN można wyróżnić metody ukierunkowane na zasięg (*coverage oriented*) oraz pojemność (*capacity oriented*) sieci [456].

W zasięgowej metodzie planowania sieci dąży się do zapewnienia odpowiedniego pokrycia sygnałem radiowym zadanego obszaru, przy jak najmniejszej liczbie punktów dostępu AP.

W drugim sposobie maksymalizowana jest pojemność sieci, mierzona liczbą użytkowników posługujących się odpowiednim sprzętem – stacjami ST (*STation*), którym należy zagwarantować odpowiednie przepustowości.

W przypadku metod planowania sieci WLAN standardu IEEE 802.11, opartych na kryterium pojemności, liczba aktywnych połączeń w sieci wynika przede wszystkim z użytego protokołu dostępu do kanału radiowego – CSMA/CA (*Carrier Sense Multiple Access/Collision Avoidance*) oraz techniki dostępu do łącza – którą jest funkcja DCF (*Distributed Coordination*

Function), określająca i porządkująca zasady dostępu do sieci, jednak nie eliminująca ewentualnych kolizji pakietów.

Planowanie zorientowane na pojemność sieci jest zdecydowanie trudniejszym i dużo bardziej złożonym zadaniem od planowania ukierunkowanego na zasięg. Niesie ono ze sobą konieczność oszacowania zapotrzebowania na określony rodzaj przesyłanych danych. Przykładowo, inne wymagania, ze względu na wartość opóźnienia czy szybkość transmisji [64], stawiane są strumieniowaniu informacji multimedialnych, a inne prostym transferom plików.

Analizując metody planowania sieci, można zauważyć, że samo zapewnienie odpowiedniego zasięgu ich działania może być niewystarczające [454]. Dodatkowo, ze względu na wieloaspektowość procesu planowania rozmieszczenia punktów dostępu AP, samo sformułowanie problemu optymalizacyjnego może być równie skomplikowane, jak jego rozwiązanie [64].

Na wstępnym etapie automatycznego projektowania sieci WLAN, w celu określenia jakości poszukiwanego rozwiązania, należy zdefiniować odpowiedni wskaźnik oceny nazywany funkcją kryterialną.

W literaturze tego przedmiotu funkcje kryterialne uwzględniają między innymi takie parametry sieci WLAN jak: zasięg radiowy [455], odległość między punktem dostępu AP a stacjami ST [64], [456], tłumienie fali radiowej na trasie punkt dostępu AP-stacja ST [457], [458], [459], [460], [461], [462], stosunek mocy nośnej do mocy zakłóceń interferencyjnych *SIR* (*Signal to Interference Ratio*) na wejściu odbiornika [455], stosunek mocy nośnej do sumy mocy zakłóceń interferencyjnych i szumów własnych *SINR* (*Signal to Interference and Noise Ratio*), bitową stopę błędów *BER* (*Bit Error Rate*) [463], [464], [465], pakietową stopę błędów *PER* (*Packet Error Rate*) [466], moc sygnału nośnej na wejściu odbiornika stacji ST, moc interferencji [457], przepływność (szybkość transmisji) [467], wskaźniki sprawiedliwego wykorzystania zasobów *JFI* (*Jain's Fairness Index*) i *CFI* (*Capacity Fairness Index*) [468], a także szereg innych wielkości i wskaźników [1].

Zazwyczaj funkcje kryterialne są nieliniowe, a poszukiwanie ich wartości optymalnych wymaga stosowania złożonych algorytmów optymalizacyjnych OPA (Optimisation *Algorithms*).

Metody optymalizacji można podzielić na: oparte na algorytmach deterministycznych [3] i niedeterministycznych, w tym na algorytmach przybliżonych (*approximation algorithms*), nazywanych też heurystycznymi.

Metody bazujące na wyborze przypadkowym wykorzystano w algorytmie Metropolisa (1953 r.), nazywanym algorytmem symulowanego wyżarzania SA (*Simulated Annealing*) (1982 r.) oraz innych, dla których inspiracją były procesy zachodzące w przyrodzie. Należą do nich algorytmy ewolucyjne EA (*Evolutionary Algorithms*), wykorzystujące, tzw. miękką optymalizację komputerową (*soft computing*). Metoda ta została zaproponowana przez Rechenberga w pracy [469] dotyczącej strategii ewolucyjnej w latach 60. XX w. Jego podstawą była teoria ewolucji Darwina, jako jeden z możliwych i skutecznych sposobów odnajdywania akceptowalnych rozwiązań dla prawdziwych, często bardzo złożonych problemów obliczeniowych. Z czasem idea ta została rozwinięta, a dzisiaj algorytmy ewolucyjne można podzielić na trzy kategorie: strategie ewolucyjne [469], programowanie ewolucyjne [470] oraz algorytmy genetyczne GA (*Genetic Algorithm*) [89].

Inną grupę metod stanowią algorytmy rojowych nazywanych także algorytmami stadnymi. Są to algorytmy, które do rozwiązania złożonych problemów optymalizacyjnych stosują paradygmaty zjawisk występujących w stadzie zwierząt (ptaków, ryb), w roju insektów czy owadów.

Można zauważyć, że znanych jest kilkaset algorytmów wykorzystujących zjawiska zachowań istniejące w naturze w świecie zwierząt czy roślin.

Jednym z rozważanych algorytmów, algorytm pszczeli BA (*Bee Algorithm*) umożliwia przeszukiwanie przestrzeni rozwiązań problemu wyrażonego funkcją kryterialną w sposób naśladujący zdobywanie nektaru przez rój pszczół.

Warto wspomnieć, że prace nad symulacją roju pszczół powstały już na przełomie lat 70-tych i 80-tych ubiegłego wieku [7]. Obecnie istnieje duża liczba modyfikacji i rozszerzeń dla podstawowego algorytmu.

Algorytm optymalizacji rojem cząstek PSO (*Particle Swarm Optimisation*) zwany algorytmem ptasim, zaproponowany został przez J. Kennedy'ego i R. Eberharta w 1995 r. Ten algorytm rojowy bazuje na zachowaniu całej populacji, w której istnieje możliwość komunikowania się między osobnikami (cząstkami) i dzielenia się informacjami.

W stosunkowo nowym, bo z 2007 roku, algorytmie świetlika FA *(Firefly Algorithm*) rozwiązanie problemu optymalizacji oparte jest na różnicy intensywności światła, która jest proporcjonalna do wartości funkcji kryterialnej. Każdy jaśniejszy świetlik przyciąga do siebie pozostałe, co pozwala na skuteczne badanie przestrzeni przeszukiwań [428].

Jedną z nowszych metod optymalizacji jest algorytm kukułki CS (*Cukko Search*). W algorytmie tym naśladowane są mechanizmy niektórych gatun-

ków kukułki, które wykorzystują gniazda innych ptaków wychowywania potomstwa.

W ostatnich latach można zauważyć rozwój metod optymalizacji wielokryterialnej [8], które obok metod deterministycznych używają algorytmów rojowych, oraz algorytmów ewolucyjnych [471]. Są to metody wnioskowania, w których poszukiwane jest takie niezdominowane rozwiązanie optymalne, które może zostać zaakceptowane z punktu widzenia każdego z kryteriów. Metody ewolucyjne używają technik takich jak: NSGAII (*Non-Dominated Sorting Genetic Algorithm*) [441], Micro-GA [445], SPEA (*Strength Pareto Evolutionary*) [438], SPEA2 [439] czy AMGA2 (*Archive-based Micro Genetic Algorithm*).

2.2. Rozwój sieci WLAN

Rozwój sieci WLAN rozpoczął się w 1997 roku, kiedy organizacja IEEE (*Institute of Electrical and Electronics Engineers*) opublikowała dokumenty standaryzacyjne dla lokalnej sieci bezprzewodowej, pracującej w nielicencjonowanym paśmie ISM (*Industry, Science, Medicine*) – 2.4 GHz. Standard ten otrzymał oznaczenie IEEE 802.11 i początkowo oferował szybkości transmisji: 1 oraz 2 Mb/s. Po wprowadzeniu w 1999 r. standardu IEEE 802.11b szybkość transmisji podniesiono do 11 Mb/s. Wzrost zapotrzebowania na duże szybkości transmisji w sieciach WLAN jedynie w niewielkim stopniu spełnił opublikowany w tym samym roku standard IEEE 802.11a, który oparto na paśmie UNII (*Unlicensed National Information Infrastucture)* – 5.2 GHz. Standard ten oferował maksymalną szybkość transmisji na poziomie 54 Mb/s, ale nie był kompatybilny z IEEE 802.11b. W roku 2002 opracowano IEEE 802.11g, w którym zastosowano podobne rozwiązania, jak w IEEE 802.11a. Zachowana została także kompatybilność wsteczna ze standardem IEEE 802.11b.

W 2002 roku rozpoczęto pracę nad standardem IEEE 802.11n [472] – zatwierdzonym we wrześniu 2009 roku, który wykorzystywał zarówno pasmo ISM 2.4 GHz, jak i UNII 5.2 GHz. Maksymalną szybkość transmisji – 600 Mb/s osiągnięto dzięki zastosowaniu: techniki wieloantenowej MIMO (*Multiple Input, Multiple Output*) o czterech antenach nadawczych i czterech odbiorczych, połączeniu dwóch kanałów 20 MHz w jeden o szerokości 40 MHz, krótszym o połowę odstępie ochronnym (upływającym po zakończeniu transmisji każdego symbolu OFDMA) SGI (*Short Guard*

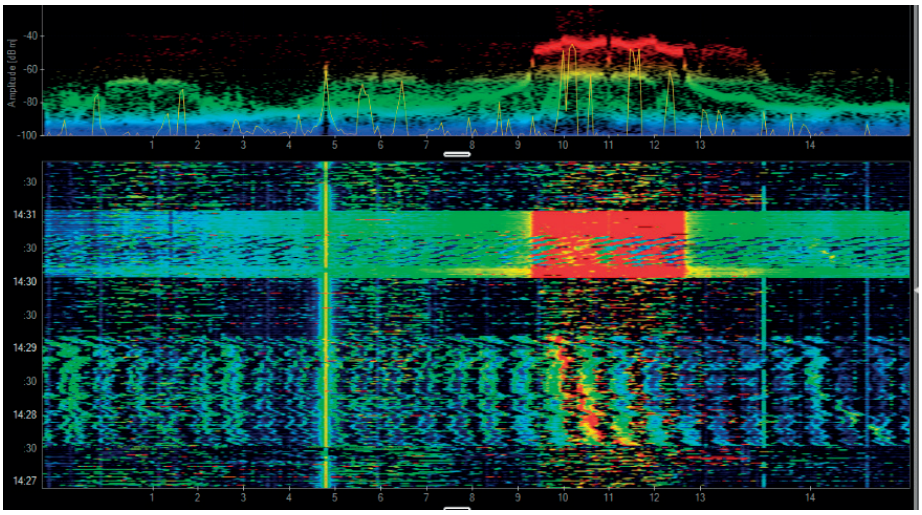
Interval) – 400 ns, agregacji ramek A-MPDU (*Aggregate MAC Protocol Data Unit*) oraz A-MSDU (*Aggregate MAC Service Data Unit*).

Zatwierdzonym w styczniu 2014 r., wiodącymi standardami sieci WLAN w paśmie UNII 5.2 GHz zostały: IEEE 802.11ac [473], w 2021 r. IEEE 802.11ax [474] a od 2024 r. IEEE 802.11be [475].

W standardzie tym zwiększono szybkość transmisji do 867 Mb/s, a przy wykorzystaniu techniki MU-MIMO (*Multi-User MIMO*) nawet do 6,93 Gb/s. Zastosowano w nim: modulację 256-QAM połączoną z kodowaniem o sprawności (*code rate*) 5/6, zwiększono szerokość pasma kanału radiowego do 160 MHz oraz zastosowano 8 strumieni przestrzennych (dla punktu dostępu AP), a także 4 strumienie przestrzenne dla stacji abonenckiej ST. W innych, firmowych rozwiązaniach, w tym, np. w ASUS, zastosowano w paśmie ISM 2.4 GHz modulacje 256QAM (TurboQAM) oraz 1024QAM (NitroQAM). W standardzie IEEE 802.11ax [474] w każdym z dwóch pasm UNII 5.2 GHz można uzyskać szybkości transmisji 4,80 Gb/s, czyli sumarycznie 9,61 Gb/s oraz dodatkowo 1,15 Gb/s w paśmie ISM 2.4 GHz (inne parametry przedstawiono w tabeli 2.8). W 2024 r. wprowadzono kolejny standard IEEE 802.11be [475] w którym oprócz pasm 2.4 GHz oraz 5.2 GHz zastosowano możliwość pracy w pasmie 6 GHz, a dokładniej UNII–5 w zakresie 5,945 – 6,425 GHz. W standardzie IEEE 802.11be [475] zastosowano: modulacje 4096QAM, szerokość pasma 320/160+160 MHz i 240/160+80 MHz, a maksymalna szybkośc transmisji wzrosła do 23,059 Gb/s, a sumarycznie nawet do 46,118 Gb/s [476].

Analizując właściwości lokalnych sieci bezprzewodowych WLAN, można stwierdzić, że do ich głównych zalet należy zaliczyć: możliwość ich szybkiej rozbudowy i modyfikacji, duży zasięg oraz nomadyczność stacji abonenckich.ST. Ważnym atutem tego typu sieci jest ich działanie w nielicencjonowanych pasmach: ISM 2.4 GHz, UNII 5.2 GHz oraz 6 GHz.

Brak licencjonowania transmisji w w/w pasmach może prowadzić do ich bardzo dużej zajętości (zwłaszcza w paśmie ISM 2.4 GHz). Wynika to z powszechnego wykorzystania tych pasm w urządzeniach różnych standardów, przeznaczonych nie tylko do komunikacji bezprzewodowej, co w sposób naturalny prowadzi do wzrostu interferencji (rys. 2.1) i tym samym ogranicza zasięg i pośrednio także przepustowości sieci WLAN.



Rysunek 2.1. Przykładowy obraz widma fal elektromagnetycznych w paśmie ISM 2.4 GHz z kilkoma pracującymi punktami dostępu AP, kuchenką mikrofalową oraz mikrofonem bezprzewodowym

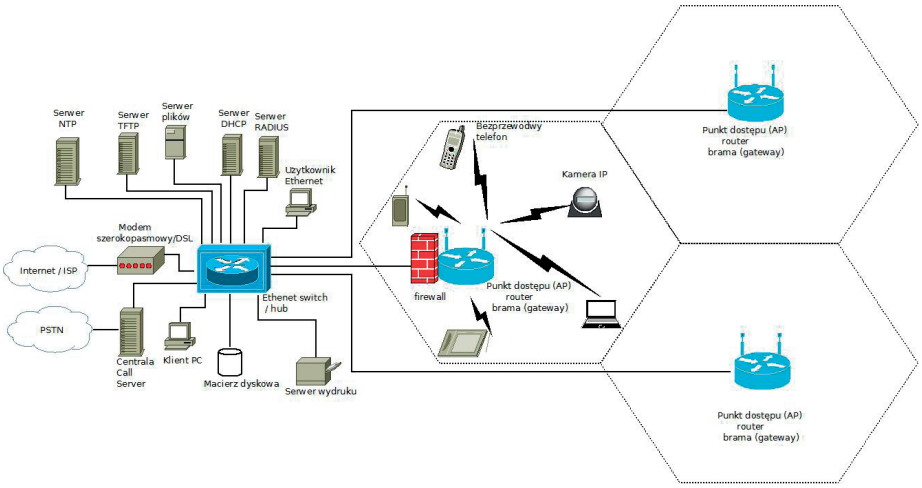
2.3. Elementy funkcjonalne sieci WLAN

Sieci WLAN standardu IEEE 802.11 mogą pracować w trzech podstawowych trybach:

- IBSS (*Independent Basic Service Set*) – sieć okazjonalna lub tymczasowa (*ad–hoc*),
- BSS (*Basic Service Set*) – sieć z infrastrukturą,
- ESS (*Extended Service Set*) – sieć powstała z połączenia, co najmniej dwóch podsieci BSS, w której punkty dostępu AP są połączone ze sobą przewodowo lub za pomocą bezprzewodowego systemu dystrybucyjnego, np. WDS (*Wireless Distribution System*).

Urządzenia w sieci WLAN standardu IEEE 802.11 mogą działać w kilku różnych topologiach, np.: gwiazdy, rozszerzonej gwiazdy, hierarchicznej, magistrali, pierścienia, podwójnego pierścienia lub kraty ESS (*Mesh Networking*).

Sieć pracująca w trybie BSS, czyli z infrastrukturą, będąca przedmiotem niniejszej monografii, wymaga specjalizowanej stacji, nazywanej punktem dostępu AP. Ponieważ stacje abonenckie ST – należące do wybranej sieci BSS – nie komunikują się bezpośrednio między sobą, dlatego punkt dostępu AP jest centralnym węzłem komunikacyjnym (rys. 2.2).



Rysunek 2.2. Przykładowa architektura sieci WLAN [477]

Punkt dostępu AP może być urządzeniem realizującym cały zakres funkcji sieciowych albo posiadać ograniczone możliwości ich realizacji. Sieć z infrastrukturą wykorzystująca, punkty dostępu, tzw. lekkiej konstrukcji, LAP (*Lightweight Access Point*) o ograniczonych funkcjach sieciowych wymaga dodatkowego urządzenia, którym może być sterownik/kontroler sieci bezprzewodowej WLC (*Wireless Lan Controller*). Jest to urządzenie, z którym punkty LAP komunikują się za pomocą dedykowanego protokołu, np. LWAPP (*Lightweight Access Point Protocol*) RFC 5412 i w których jest realizowane:

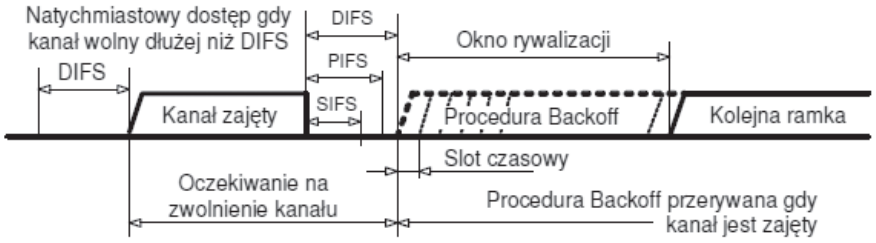
- wykrywanie interferencji oraz przydział kanału radiowego,
- równoważenie obciążenia w sieci,
- dynamiczny przydział mocy oraz wykrywanie i korekcja braku zasięgu.

Do zarządzania urządzeniami sieciowymi może być używany także protokół SNMP (*Simple Network Management Protocol*). Całość może tworzyć bezprzewodowy system sterowania (*wireless control system*).

2.4. Techniki dostępu do kanału radiowego

Dla potrzeb sieci WLAN opracowano szereg protokołów warstwy MAC, które oparto na nasłuchiwaniu nośnej w kanale radiowym, w celu unikania kolizji z aktualnie trwającą transmisją. Wszystkie znane i stosowane protokoły wielodostępu nasłuchujące nośną z rodziny CSMA (*Carrier Sense Multiple Access*) zostały szczegółowo omówione w pracy [478].

Przyjętą przez organizację IEEE metodą dostępu do kanału radiowego w sieci standardu IEEE 802.11 jest DCF (*Distributed Coordination Function*) [453]. Realizuje ona założenia techniki CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), który określa zasady dostępu do sieci stacji abonenckich ST (rys. 2.3).



Rysunek 2.3. Diagram dostępu do kanału radiowego w metodzie DCF

Stacja abonencka ST rywalizująca o dostęp do łącza bezprzewodowego sprawdza jego zajętość, wykorzystując w tym celu procedurę CCA (*Clear Channel Assessment*), realizującą dwie funkcje: wykrywania nośnej CCA-CS (*CCA–Carrier Sense*) oraz detekcję energii CCA-ED (*CCA-Energy Detect*).

Funkcja detekcji energii CCA-ED pozwala wykryć w odbiorniku stacji abonenckiej ST transmisję realizowaną w danym kanale radiowym w ramach innej sieci WLAN niż ta, w której ona pracuje i tym samym wstrzymywania się z wysyłaniem danych. Z kolei funkcja wykrywania nośnej CCA-CS umożliwia wykrycie i analizę w odbiorniku stacji ST wszystkich pakietów oraz dekodowania sygnału w warstwie fizycznej.

Uzupełnieniem metody DCF dostępu do kanału radiowego jest opcjonalna technika RTS/CTS (*Request To Send/Clear To Send*). Ogranicza on kolizje ramek wynikające z problemu występowania w sieciach WLAN, tzw. stacji „ukrytych", a w niektórych scenariuszach rozwiązuje także problem stacji „odkrytej". Jednakże jej stosowanie może wiązać się ze wzrostem opóźnień transmisji oraz zmniejszeniem przepustowości sieci.

Dodatkowo w standardzie IEEE 802.11e zdefiniowano technikę EDCA (*Enhanced Distributed Channel Access)*, która rozszerza możliwości DCF i zapewnia, tzw. czas możliwości realizacji transmisji TXOP (*Transmit Opportunity*). TXOP jest to okres, w którym stacja abonencka ST może nadawać bez konieczności rywalizacji o kanał radiowy. Ponadto w technice EDCA, w celu zapewnienia odpowiedniej jakości usług QoS (*Quality of Service*) świadczonych w sieci WLAN, wprowadzono cztery klasy ruchu: głosu VO, wideo VI, BE (*best effort)* i tła BK [453].

Wcześniejsze generacje standardów IEEE 802.11, takie jak IEEE 802.11n [472] oraz IEEE 802.11ac [473], koncentrowały się głównie na poprawie maksymalnej przepustowości, w nowszych generacjach tych standarów dąży się do: poprawy wydajności w gęstych sieciach (IEEE 802.11ax [474]), zwiększenia przepustowości i niezawodności, zmniejszenia opóźnień (IEEE 802.11be [475]) oraz rozważa się kolejną klasę ruchu wrażliwą na opóźnienia TS (*Time Sensitive*) [482]. TSN (*Time Sensitive Networking*) jest to zestaw nowych technologii (IEEE 1588 z kolejnymi wersjami, IEEE 802.1AS, IEEE 802.1Qbu, IEEE 802.3br, P802.1Qcc, IEEE 802.Qbv, IEEE 802.1CB, IEEE 802.1Qci, IEEE 802.1Qdj) opracowany w celu poprawy komunikacji czasu rzeczywistego we współczesnych przemysłowych sieciach Ethernet, umożliwiający sterowanie przesyłaniem danych, ustalanie priorytetów oraz uwzględnienie wymagań poszczególnych aplikacji, takie jak gwarantowana przepustowość, synchronizacja czasu czy niskie opóźnienia poszczególnych aplikacji [479], [481].

2.5. Proces planowania sieci WLAN w środowisku wewnątrzbudynkowym

W procesie projektowania i budowy sieci WLAN z infrastrukturą można wyróżnić szereg etapów [482]. Do najważniejszych z nich należy:

- ustalenie liczby stacji abonenckich ST współpracujących z punktami dostępu AP wraz z ich szczegółową charakterystyką, w tym wymaganiami i rodzajem realizowanych usług telekomunikacyjnych,
- wybór techniki transmisji oraz standardu sieci,
- przeprowadzenie wstępnych testów empirycznych, np. umożliwiających oszacowanie zasięgu radiowego sieci,
- wykonanie projektu sieci,

– przeprowadzenie testów zasięgu i/lub przepustowości zaprojektowanej sieci,
– sporządzenie dokumentacji z przeprowadzonych testów w sieci oraz dokonanie analizy osiągniętych wyników w odniesieniu do wstępnych założeń oraz ewentualnej modyfikacji projektu sieci,
– budowa i uruchomienie sieci WLAN,
– utrzymanie sieci i jej ewentualna optymalizacja.

Główne zadania związane z planowaniem sieci WLAN na etapie jej projektowania koncentrują się na:
– określeniu liczby punktów dostępu AP oraz ich rozmieszczenia na obszarze sieci,
– wstępnej ocenie zasięgu radiowego sieci we wnętrzu budynku,
– przydziale numerów kanałów radiowych (ch_{nr}), ustaleniu mocy nadawczych poszczególnych punktów dostępu AP oraz parametrów działania punktów dostępu AP.

Rozpoczynając planowanie i projektowanie sieci, zaleca się, aby punkty dostępu AP [483]:
– leżały powyżej przeszkód, występujących na drodze propagacji i mogących zakłócić bezpośrednią widoczność anten stacji abonenckich ST i punktów AP,
– znajdowały się w miarę możliwości w centrum rozważanego obszaru działania projektowanej sieci i najlepiej bezpośrednio pod sufitem,
– były rozmieszczone w pomieszczeniu, w którym zakłada się przebywanie dużej liczby potencjalnych użytkowników sieci.

Sformułowanie i rozwiązanie zadania optymalizacji, którego rezultatem jest wskazanie najlepszych lokalizacji punktów AP na obszarze działania sieci WLN zazwyczaj wymaga [6]:
– wyboru i zdefiniowania funkcji kryterialnej (F_c) na której zostanie oparty proces optymalizacji rozmieszczenia punktów AP,
– opracowania modelu matematycznego analizowanej sieci WLAN,
– zastosowania jednego z wielu algorytmów optymalizacji OPA, poszukiwania rozwiązania sformułowanego zadania optymalizacji.

Określenie liczby punktów dostępu AP

Liczba stacji abonenckich ST możliwych do obsłużenia przez pojedynczy punkt dostępu AP jest relatywnie duża (np. 256 stacji abonenckich ST) i zależy od rozmiarów tablicy MAC w punkcie dostępu AP. W dobrze

zwymiarowanych i efektywnie działających sieciach WLAN liczba stacji abonenckich ST przypadająca na jeden punkt dostępu AP nie powinna być większa niż 30 [453].

Aby wstępnie oszacować dla zadanej liczby stacji abonenckich N_{ST} wymaganą liczbę N_{AP} punktów dostępu AP, można posłużyć się wzorem [64]:

$$N_{AP} = \frac{\sum_{i=1}^{N_{ST}} M_{TR_i} \cdot \nu_i}{M_{TR}^{max} \cdot TUL} = \frac{\sum_{i=1}^{N_{ST}} M_{TR_i} \cdot \nu_i}{S_{TUL}}, \tag{2.1}$$

gdzie:

M_{TRi} – wymagana szybkość transmisji i-tej stacji abonenckiej ST,

ν_i –współczynnik, określający aktywności użytkownika posługującego się i–tą stacją abonencką ST,

M_{TR}^{max} – maksymalna szybkość transmisji ST/AP,

TUL – wykorzystanie punktu AP (górne pasmo przepustowości dla punktu AP),

S_{TUL} – górna granica przepustowości dla punktu AP.

Dokładne oszacowanie wymaganej liczby N_{AP} punktów dostępu AP dla zadanej liczby stacji abonenckich N_{ST} przedstawiono w rozdziale 2.12.6 [487].

Określenie położenia punktów dostępu AP

Analizując zagadnienie planowania rozmieszczenia punktów dostępu AP w sieci WLAN [453], [458], można zauważyć, że dominujące procedury związane z planowaniem są zorientowane na zasięg (*coverage oriented*), rzadziej na pojemność (*capacity oriented*) sieci.

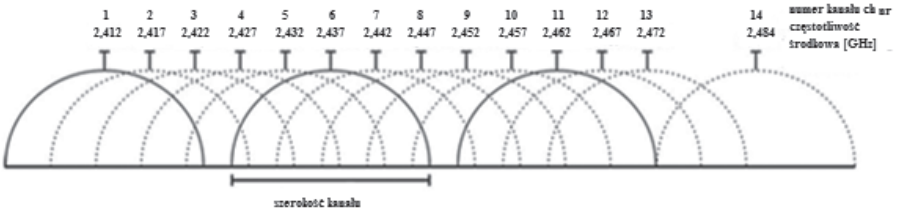
W podejściu zasięgowym, czyli zorientowanym na zasięg działania sieci, dąży się do zapewnienia jak najlepszego pokrycia sygnałem radiowym rozważanego obszaru, przy minimalnej liczbie punktów dostępu AP.

W podejściu pojemnościowym, czyli zorientowanym na pojemność sieci, optymalizowana jest szybkość transmisji każdej stacji abonenckiej ST działającej w sieci. Należy zaznaczyć, że planowanie zorientowane na pojemność niesie ze sobą konieczność oszacowania zapotrzebowania na określony typ przesyłanych danych (np. aplikacji strumieniowych lub pakietowych) oraz ich wrażliwość na opóźnienia czy szybkości transmisji [64].

Rozmiary komórek w sieci zorientowanej na pojemność są zazwyczaj mniejsze w porównaniu z sieciami opartymi na kryterium zasięgowym, co niesie za sobą konieczność zwiększenia liczby punktów dostępu AP.

Przydział kanałów radiowych

Nielicencjonowane pasmo ISM 2.4 GHz dla standardu IEEE 802.11 zostało zdefiniowane przez ITU-R i podzielone przez FCC (USA) i ETSI (Europa) w zależności od kraju na: 11 (USA), 13 (Europa) lub 14 (Japonia, dla IEEE 802.11b) kanałów radiowych – każdy o szerokości pasma 22 MHz.



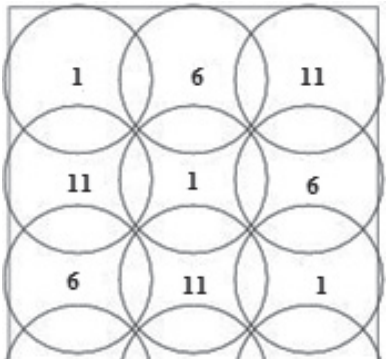
Rysunek 2.4. Plan kanałów radiowych w pasmie ISM 2.4 GHz dla rodziny standardów IEEE 802.11

Wzajemne zakłócanie się punktów dostępu AP, czyli tzw. interferencje sąsiedniokanałowe wynikają z faktu, że sieci WLAN standardu IEEE 802.11 działające w paśmie ISM 2.4 GHz mają do dyspozycji tylko trzy ortogonalne, nienachodzące na siebie, kanały radiowe ch_{nr}, np. (1, 6, 11) oraz cztery kanały o numerach (1, 5, 9, 13), tylko w nieznaczny sposób nakładające się na siebie (rys. 2.4).

W paśmie ISM 2.4 GHz we wszystkich przypadkach maksymalna moc zastępcza promieniowana izotropowo *EIRP* (*Effective/Equivalent Isotropically Radiated Power*) może wynosić co najwyżej 100 mW [485].

Przydział kanału radiowego danemu punktowi AP może zostać przeprowadzony na kilka sposobów, w tym w szczególności, np. przy użyciu funkcji kryterialnej z przeprowadzeniem procesu optymalizacji, którego celem może być: minimalizacja poziomu interferencji, maksymalizacja pojemności systemu, maksymalizacja przepływności systemu, maksymalizacja indeksu sprawiedliwości itd.

W zależności od przyjętego rozwiązania stosowana jest automatyczna selekcja kanałów z uwzględnieniem: poziomu mocy odbieranego sygnału, interferencji, stanu wykorzystania kanału i ruchu w sieci czy liczby stacji ST przypisanych do punktu dostępu. Ten ostatni sposób realizowany jest przez algorytm LCCS (*Least Congested Channel Search*) stosowany w urządzeniach Cisco.



Rysunek 2.5. Przykładowe rozmieszczenie nakładających się komórek o kanałach radiowych nr 1, 6 i 11 odseparowanych częstotliwościowo oraz przestrzennie

W celu zapewnienia poruszającym się stacjom abonenckim ST możliwości przenoszenia połączeń między punktami dostępu AP, obszary komórek (zasięgów radiowych) o różnych kanałach mogą na siebie nachodzić (rys. 2.5).

Konfiguracja i zarządzanie wybranymi parametrami sieci WLAN

W systemach scentralizowanych zarządzanie zasobami radiowymi można prowadzić także we działającej już sieci bezprzewodowej. Taki typ zarządzanie może polegać, m.in. na automatycznym wykrywaniu punktów dostępu AP, doborze mocy nadawczych i przydziale kanału radiowego punktom dostępu AP, równoważeniu obciążenia (*load balacing*) punktów AP, wykrywaniu obcych punktów dostępu AP oraz nieuprawnionych do pracy stacji abonenckich ST, a także kształtowanie diagramów kierunkowych anten (*beamforming*) w punktach dostępu AP.

W procesie konfiguracji i zarządzania sieciami WLAN można wyróżnić szereg parametrów, które mogą mieć istotny wpływ na efektywność funkcjonowania sieci. Można do nich zaliczyć:

- parametry protokołu wielodostępu do kanału radiowego,
- typ i parametry protokołów warstwy transportowej ISO/OSI,
- parametry usług i generowanego przez nie pakietowego ruchu sieciowego,
- rozmiar maksymalnej jednostki transmisji danych *MTU* (*Maximum Transmission Unit*).

2.6. Optymalizacja sieci WLAN z zastosowaniem funkcji kryterialnych

Analizując wymagania stawiane lokalnym sieciom bezprzewodowym WLAN, można zauważyć, że samo zagwarantowanie odpowiedniego zasięgu radiowego może okazać się niewystarczające do ich efektywnego działania [457]. Z kolei zadanie optymalizacji, dotyczące rozmieszczenia i konfiguracji punktów dostępu AP w sieci WLAN z infrastrukturą, może być równie skomplikowane, jak próby jego rozwiązania [64].

Przystępując do procesu planowania sieci WLAN, należy przede wszystkim rozpocząć od sformułowania właściwej funkcji kryterialnej F_c (*criterion function*), nazywanej również wskaźnikiem jakości, funkcjonałem jakości, kryterium optymalizacji czy kryterium jakości. Funkcja kryterialna powinna gwarantować z punktu widzenia potrzeb i oczekiwań użytkowników sieci WLAN optymalne rozmieszczenie oraz konfigurację punktów dostępu AP, w konkretnym środowisku radiokomunikacyjnym.

Funkcja kryterialna F_c może uwzględniać takie parametry systemu transmisji, jak:

- odległość między punktem dostępu AP a stacją abonencką ST, która przekłada się na tłumienie sygnału w torze radiowym,
- stosunek poziomu mocy sygnału do poziomu mocy zakłóceń interferencyjnych i mocy szumów własnych odbiornika *SINR* (*Signal to Interference and Noise Ratio*),
- bitową stopę błędów *BER* (*Bit Error Rate*) i/lub pakietową stopę błędów PER (*Packet Error Rate*),
- poziom mocy nośnej *RSSI* (*Received Signal Strength Indicator*) na wejściu odbiornika stacji abonenckiej ST,
- wartości indeksów sprawiedliwego przydziału zasobów *JFI* (*Jain's Fairness Index*) oraz *CFI* (*Capacity Fairness Index*),
- szybkość transmisji,
- inne, ważne z punktu widzenia optymalizacji jednokryterialnej lub wielokryterialnej, parametry sieci WLAN [1], [486], [487], [488].

Jedną z najprostszych funkcji kryterialnych jest, tzw. funkcja zasięgowa F_{c1}, uwzględniająca jedynie odległości między punktami dostępu AP a stacjami abonenckimi ST:

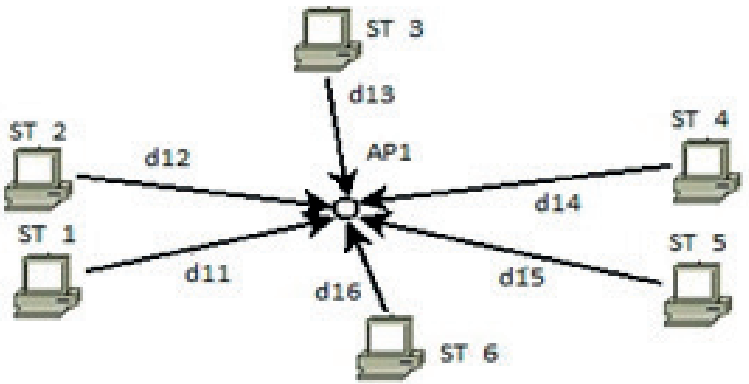
$$F_{c_1} = \sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} d_{ji}, \tag{2.2}$$

gdzie:

N_{AP} – liczba punktów dostępu AP,

N_{STj} – liczba stacji abonenckich ST obsługiwanych przez j-ty punkt dostępu AP,

d_{ij} – odległość między j-tym punktem dostępu AP, a stacją i-tą stacją abonencką ST.



Rysunek 2.6. Przykład optymalnego, z punktu widzenia funkcji kryterialnej F_{c_1}, rozmieszczenia stacji ST wokół punktu AP

Dla funkcji F_c, zapisanej wzorem (2.2), planowanie sieci WLAN opartej na jednym tylko punkcie dostępu AP sprowadza się do takiego jego umiejscowienia, dla którego suma odległości między stacjami abonenckimi ST a tym punktem jest jak najmniejsza (rys. 2.6).

Zasięgowa funkcja kryterialna może zostać zdefiniowana również tak, by uwzględniała wartość tłumienia fali radiowej na drodze jej propagacji między punktem dostępu AP a stacją ST. W takim podejściu, funkcja kryterialna może zostać wyrażona jako [460], [461]:

$$F_{c_2} = \Psi \cdot F_{c_2}^1 + (1 - \Psi) \cdot F_{c_2}^2 \tag{2.3}$$

gdzie:

Ψ jest współczynnikiem sterującym, spełniającym nierówność $1 \leq \Psi \leq 0$, a występujące we wzorze (2.3) funkcje $F_{c_2}^1$ i $F_{c_2}^2$ przyjmują postać:

$$F_{c_2}^1 = \sum_{i=1}^{N_{ST}} w_i \cdot \left(L_i^j + \mu_p \cdot max\{0, L_i^j - L_{i,max}\}\right), \tag{2.4}$$

$$F_{c_2}^2 = max\left(L_i^j + \mu_p \cdot max\{0, L_i^j - L_{i,max}\}\right), \tag{2.5}$$

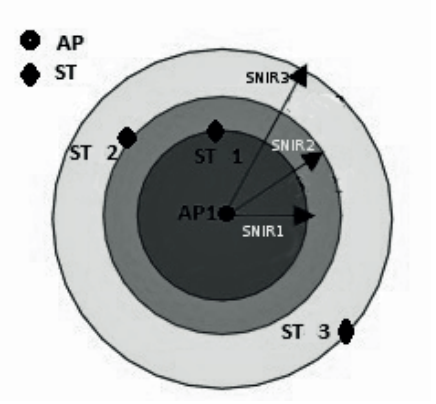
gdzie:

L_j^i – tłumienie toru radiowego między i-tą stacją abonencką ST a j-tym punktem dostępu AP,

$L_{i,max}$ – maksymalne dopuszczalne tłumienie sygnału,

μ_p – współczynnik kary $\mu_p > 0$.

Dodatkowo, wagi w_i we wzorze (2.4) spełniają warunek normalizacyjny: $\sum_{i=1}^{N_{ST}} w_i = 1$.



Rysunek 2.7. Koncepcja podziału obszaru zasięgu radiowego punktu dostępu AP z punktu widzenia funkcji F_{c_3} i F_{c_4}

Dwie kolejne funkcje kryterialne realizują koncepcję podziału zasięgu radiowego punktu dostępu AP na współśrodkowe obszary, w których transmisja odbywa się z szybkościami zależnymi od stosunku mocy *SINR* (rys. 2.7). Funkcje te uwzględniają zarówno zasięg radiowy, jak i szybkości transmisji (F_{c_3}) bądź przepustowości (F_{c_4}) osiągane przez poszczególne stacje abonenckie ST wymieniające dane z punktami dostępu AP sieci WLAN.

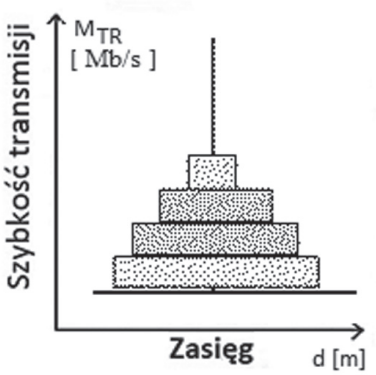
Wartość *SINR* można obliczyć według zależności:

$$SINR = \overline{RSL} - 10 \cdot log_{10}(N_{power} + I_t) - NF = \\ = \overline{RSL} - 10 \cdot log_{10}(k_B \cdot T_{emp} \cdot B + I_t) - NF, \tag{2.6}$$

gdzie:

$\overline{RSL}$ (*Received Signal Level*) – średni poziom mocy nośnej na wejściu odbiornika stacji abonenckiej ST,
N_{power} – poziom mocy szumów własnych odbiornika,
I_t – całkowita moc szumów interferencyjnych,
k_B – stała Boltzmana,
T_{emp} – temperatura fizyczna odbiornika w kelwinach,
B – szerokość pasma kanału radiowego,
NF – współczynnik szumów odbiornika (*Noise Figure*).

W literaturze przedmiotu, taki uproszczony model dla wielu szybkości transmisji określany jest terminem „Wieży Hanoi" (*Hanoi Tower*) [489]. Średnica poszczególnych elementów wieży odzwierciedla zasięg radiowych, a ich wysokość – szybkość transmisji (rys. 2.8).



Rysunek 2.8. Kształt „Wieży Hanoi" uwzględniający zasięgi i szybkości transmisji

Pierwszą z funkcji kryterialnych, wykorzystującą w/w model można wyrazić jako:

$$F_{c_3} = \sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} M_{TR_i} \cdot \left(d_{ji}^2 - d_{j(i-1)}^2\right), \tag{2.7}$$

gdzie:

N_{STj} – liczba stacji abonenckich ST obsługiwanych przez j-ty punktu dostępu AP,

d_{ji} – odległość między i-tą stacją abonencką ST a j-tym punktem dostępu AP,

M_{TRi} – szybkość transmisji i-tej stacji abonenckiej ST.

W przypadku kryterialnej funkcji zasięgowej opartej na przepustowości grupy stacji abonenckich ST, wykorzystano modele matematyczne sieci WLAN, będące, zaproponowanym przez Bianchiego [490] dla funkcji DCF, rozwinięciem modelu Markowa, których szczegółowy opis zamieszczono w czwartym rozdziale niniejszej monografii.

Funkcja kryterialna bazująca na przepustowościach S_k została wyrażona wzorem:

$$F_{c_4} = \sum_{j=1}^{N_{AP}} \sum_{k=1}^{L_{TR_j}} \frac{S_k}{N_{ST_{jk}}} \left(d_{jk}^2 - d_{j(k-1)}^2\right), \tag{2.8}$$

gdzie:

L_{TRj} – liczba rozważanych szybkości transmisji dla j-tego punktu dostępu AP,

S_k – całkowita przepustowość stacji abonenckich ST dla k-tej szybkości transmisji,

d_{jk} – odległość między najdalszą stacją abonencką ST realizującą transmisję z k-tą szybkością transmisji a j-tym punktem dostępu AP.

Analogicznie, jak w przypadku funkcji kryterialnej F_{c_4} (2.8), wskaźnik wydajności *PM* (*Performance Metric*) sieci WLAN [494] zdefiniowano jako:

$$PM = \sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} S_i \cdot \left(d_{ji}^2 - \mathrm{d}_{j(i-1)}^2\right), \tag{2.9}$$

Przedstawiona wzorem 2.8 funkcja kryterialna F_{c_4} oraz PM (2.9) łączą w sobie zasięg radiowy oraz przepustowość sieci WLAN [491], a przez to są zależne od techniki dostępu do kanału radiowego [492].

2.7. Pozostałe funkcje kryterialne

W pracy [493] zdefiniowano dla potrzeb planowania i optymalizacji sieci WLAN funkcję kryterialną opartą na wykorzystaniu odpowiednio dużej liczby N_{ST} stacji abonenckich ST, losowo rozmieszczonych wokół punktów AP. Stacje te są traktowane jako swego rodzaju punkty pomiarowe ([494], [92]), umożliwiające wyznaczenie granic zasięgu działania sieci WLAN czy też jej poszczególnych punktów dostępu AP rozmieszczonych na danym obszarze.

W tak zdefiniowanej funkcji kryterialnej może zostać także ujęta informacja o tym, jaka część rozważanych stacji abonenckich ST – punktów pomiarowych znajduje się w zasięgu konkretnego punktu dostępu AP.

W sytuacji, gdy samo zagwarantowanie zasięgu radiowego w planowanej sieci WLAN nie wyczerpuje wszystkich oczekiwań użytkowników, w funkcji kryterialnej można uwzględnić szybkości transmisji wymagane dla poszczególnych stacji abonenckich ST:

$$F_{c_5} = \Psi \cdot \left(\frac{\sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} M_{TR_{ij}}^{max}}{N_{ST} \cdot M_{TR}^{max}} \right) + (1 - \Psi) \cdot \left(\frac{\sum_{j=1}^{N_{AP}} N_{ST_j}}{N_{ST}} \right), \tag{2.10}$$

gdzie:

$M_{TR_{ij}}^{max}$ – maksymalna szybkość transmisji możliwa do osiągnięcia przez i–tą stację abonencką ST współpracującą z j-tym punktem dostępu AP, dla którego zarejestrowano największą wartości *SINR*,

M_{TR}^{max}– maksymalna szybkość transmisji obsługiwana przez punkty dostępu AP,

Ψ – współczynnik ($1 \leq \Psi \leq 0$) sterujący wyborem między optymalizacją nastawioną na zasięg lub na szybkość transmisji.

We wzorze 2.10, dla wagi: $\Psi = 0$ funkcja kryterialna F_{c_5} będzie definiować zadanie optymalizacji nastawione tylko i wyłącznie na zasięg działania sieci, natomiast dla $\Psi = 1$ na szybkości transmisji osiągane w poszczególnych lokalizacjach stacji abonenckich ST.

We wzorze 2.10, dla wagi: $\Psi = 0$ funkcja kryterialna F_{c_5} będzie definiować zadanie optymalizacji nastawione tylko i wyłącznie na zasięg działania sieci, natomiast dla $\Psi = 1$ na szybkości transmisji osiągane w poszczególnych lokalizacjach stacji abonenckich ST.

Kolejną funkcję kryterialną, uwzględniającą przepustowość sieci, a przez to pośrednio i właściwości funkcji DCF można zdefiniować, jako:

$$F_{c_6} = \Psi \cdot \left(\frac{\sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} S_{ij}}{N_{AP} \cdot M_{TR}^{max}} \right) + (1 - \Psi) \cdot \left(\frac{\sum_{j=1}^{N_{AP}} N_{ST_j}}{N_{ST}} \right), \tag{2.11}$$

gdzie pierwszy składnik sumy we wzorach 2.10 i 2.11 określa odpowiednio całkowitą: szybkość transmisji 2.10 i przepustowość 2.11.

Decyzja o przydzieleniu do punktu dostępu AP i-tej stacji abonenckiej ST warunkowana może być także zapewnieniem minimalnej szybkości transmisji dla rozważanego typu usługi.

Drugi składnik sumy zarówno we wzorze 2.10, jak i 2.11 odpowiada za zasięgowy charakter zadania optymalizacji. W przypadku funkcji kryterialnych F_{c_5} i F_{c_6} oraz $\Psi = 0$, zliczane są stacje abonenckie ST, które znalazły się w zasięgu działania sieci.

Jeszcze innym podejściem optymalizacyjnym, które uwzględnia jednocześnie kilka różnych kryteriów planowania sieci WLAN z infrastrukturą, np. dotyczących zarówno zasięgu radiowego, jak i przepustowości, może być sformułowanie zadania optymalizacji wielokryterialnej MOO (*Multi–objectives Optimisation*).

W tym typie planowania sieci WLAN poszukuje się rozwiązania z punktu widzenia kilku, niekiedy nawet przeciwstawnych kryteriów optymalizacji [2], [6].

Dla potrzeba takiego typu poszukiwań, obok wcześniej wymienionych funkcji kryterialnych, można zdefiniować szereg dodatkowych funkcji uwzględniających, takie elementy jak [1]:

- określoną liczbę stacji abonenckich ST, dla których tłumienie propagacyjne przekracza założoną wartość,
- wymaganą (minimalną) liczbę punktów dostępu AP,
- całkowity koszt infrastruktury sieciowej,
- dopuszczalne tłumienia nośnej dla każdej stacji ST z osobna,
- dokładność lokalizacji GDOP/HDOP (*Geometric/Horizontal Dilution of Precision*) stacji ST [486], [487], [488],
- wartości indeksu sprawiedliwego przydziału zasobów JFI (*Jain's Fairness Index*):

$$JFI = \frac{\left(\sum_{i=1}^{N_{ST}} S_i\right)^2}{N_{ST} \cdot \sum_{i=1}^{N_{ST}} S_i^2}, \tag{2.12}$$

gdzie:

N_{ST} – liczba stacji abonenckich ST,

S_i – wartość przepustowości $S_i$dla i–tej stacji ST,

- wartości pojemnościowego indeksu sprawiedliwego przydziału zasobów CFI (*Capacity Fairness Index*) [468]:

$$CFI = \left(\sum_{i=1}^{N_{ST}} S_i\right) \cdot JFI. \tag{2.13}$$

Ponadto dla potrzeb optymalizacji wielokryterialnej MOO pod uwagę brane są funkcje kryterialne, które uwzględniają:

- całkowitą szybkość transmisji wszystkich stacji abonenckich ST:

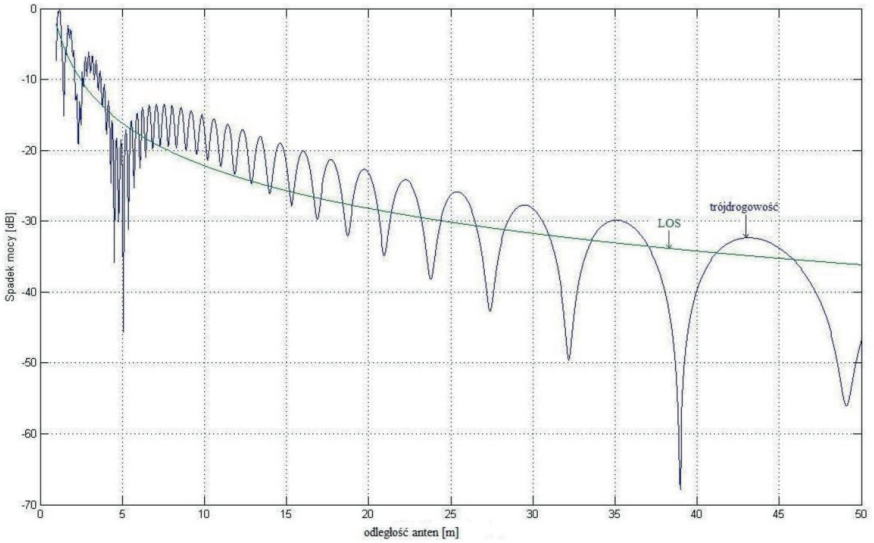
$$F_{c_7} = \sum_{j=1}^{N_{AP}} \sum_{k=1}^{N_{ST_j}} M_{TR_k}, \tag{2.14}$$

- sumaryczną przepustowość wszystkich stacji abonenckich ST:

$$F_{c_8} = \sum_{j=1}^{N_{AP}} \sum_{k=1}^{N_{ST_j}} S_k. \tag{2.15}$$

2.8. Wybrane modele propagacji fal radiowych pasma ISM 2.4 GHz

To, jak rozchodzą się fale radiowe, zależy od ich częstotliwości, polaryzacji oraz środowiska radiokomunikacyjnego. Szczegółowe rozpoznanie właściwości propagacyjnych środowiska czy to w terenie otwartym, czy we wnętrzach budynków ma istotny wpływ na poprawne zaprojektowanie i następnie na wydajną eksploatację bezprzewodowych lokalnych sieci komputerowych WLAN. To właściwości toru radiowego decydują o wzajemnym rozmieszczeniu i konfiguracji węzłów, w tym punktów dostępu AP, w sieciach łączności bezprzewodowej.



Rysunek 2.9. Przykładowy względny spadek mocy nośnej w funkcji odległości w warunkach propagacji LOS i trójdrogowości [495]

Warunki propagacyjne, występujące w określonej przestrzeni wewnątrzbudynkowej, determinują przede wszystkim możliwy do osiągnięcia zasięg radiowy i z tego względu stanowią niezwykle ważny element procesu planowania i optymalizacji sieci WLAN.

Do poprawnego zaplanowania systemu łączności bezprzewodowej w środowisku wewnątrz budynku (*indoor*) konieczne jest rozpoznanie właściwości toru radiowego, w tym geometrii budynku oraz tworzących go pomieszczeń, a także znajomość rozmieszczenia elementów wystroju i wyposażenia wnętrz oraz osób przebywających na obszarze działania systemu (rys. 3.1).

Propagację fal elektromagnetycznych zazwyczaj opisuje się za pomocą modeli matematycznych opartych na prawach optyki geometrycznej i jednolitej teorii dyfrakcji. Takie podejście pozwala uwzględnić większość najważniejszych zjawisk fizycznych, takich, jak: dyfrakcja, rozproszenie czy odbicie [496], towarzyszących rozchodzeniu się fal radiowych w obecności różnego typu przeszkód.

Do najważniejszych matematycznych modeli propagacyjnych, stosowanych w zagadnieniach planowania i optymalizacji sieci bezprzewodowych, zaliczyć należy modele deterministyczne i stochastyczne oraz

modele deterministyczne i stochastyczne o współczynnikach dobieranych empirycznie. Charakterystycznymi cechami wszystkich tych modeli powinna być ich stosunkowo niewielka złożoność obliczeniowa, a także brak wymogu posiadania dokładnych informacji o geometrii pomieszczeń oraz o parametrach elektrycznych (przewodności właściwej, przenikalności elektrycznej) i magnetycznych (przenikalności magnetycznej) obiektów–przeszkód znajdujących się na trasie propagacji fali radiowej.

Metody oparte na modelach deterministyczno-empirycznych, a wśród nich metody śledzenia promieni (*ray tracing*) i wiązek (*beam tracing*) są metodami bardzo dokładnymi, które kosztem dużych nakładów obliczeniowych pozwalają bardzo precyzyjnie określić parametry toru radiowego w dowolnym miejscu analizowanego obszaru przestrzeni. Jednak dużą niedogodnością w ich stosowaniu są: znacznie rozbudowane algorytmy obliczeniowe, konieczność dysponowania szczegółowa bazą danych na temat środowiska propagacyjnego, w tym w szczególności trudności w ustaleniu wartości przenikalności elektrycznych i przewodności właściwych materiałów, z których zbudowane są przeszkody leżące w obszarze propagacji.

Ponieważ rozchodzeniu się fal radiowych we wnętrzach budynków towarzyszy zjawisko wielodrogowości, dlatego dokładny deterministyczny model matematyczny propagacji fal w tym środowisku radiokomunikacyjnym powinien uwzględniać:

- odbicia od ścian oraz elementów wystroju i wyposażenia pomieszczeń,
- przenikanie fal przez stropy i ściany budynków,
- dyfrakcję fal na krawędziach elementów wystroju i wyposażenia pomieszczeń,
- efekt prowadnicy falowodowej w długich korytarzach,
- obecność oraz przemieszczanie się ludzi we wnętrzach pomieszczeń.

Poziom mocy czynnej wydzielonej w obciążeniu w pełni dopasowanym do anteny odbiorczej, można obliczyć ze wzoru [497]:

$$P_r = S \cdot A_e = S \cdot \frac{G_r \lambda^2}{4\pi}, \tag{2.16}$$

gdzie:

S – strumień gęstości powierzchniowej mocy fali radiowej w miejscu lokalizacji anteny odbiorczej na kierunku jej maksymalnego odbioru w W/m^2,

A_e – maksymalna powierzchnia skuteczna anteny odbiorczej w m^2,

G_r – maksymalny zysk energetyczny anteny odbiorczej w W/W,

λ – długość fali radiowej w metrach.

Przy uwzględnieniu propagacji wielodrogowej, czyli promieni docierających do anteny odbiorczej różnymi drogami, poziom mocy sygnału Pr na wejściu odbiornika, może zostać wyrażony jako:

$$P_r[\text{dBW}] = 10log_{10}\left(\left|\sum_{i=1}^{N_{ray}} \hat{E}_i\right|^2 \frac{A_e}{2Z_0}\right), \tag{2.17}$$

gdzie:

N_{ray} – liczba promieni docierających po różnych drogach do anteny odbiorczej,

Z_0 – impedancja właściwa próżni (120π Ω),

$\hat{E}_i$ – amplituda zespolona natężenia pola elektrycznego i-tego promienia fali EM równa [498]:

$$\hat{E}_i = \sqrt{\frac{P_t Z_0 G_t}{2\pi}} \prod_j R_{ij} \prod_k T_{ik} \prod_l D_{il} \frac{e^{-j\frac{2\pi f_c d_i}{c}}}{d_i^t}, \tag{2.18}$$

gdzie:

P_t – moc czynna dostarczona do zacisków wejściowych anteny nadawczej w watach

G_t – maksymalny zysk energetyczny anteny nadawczej w W/W,

j – liczba odbić i-tego promienia od przeszkód leżących na trasie jego propagacji,

R – współczynnik odbicia [499], [500], [501] pola elektrycznego,

k – liczba przejść i-tego promienia przez przeszkody leżące na trasie jego propagacji,

T – współczynnik transmisji [499] pola elektrycznego,

l – liczba ugięć i-tego promienia na przeszkodach leżących na trasie jego propagacji,

D – współczynnik dyfrakcji [502] pola elektrycznego,

d_i – długość trasy propagacji i-tego promienia w metrach,

c – prędkość światła w próżni w m/s,

f_c – częstotliwość środkowa kanału radiowego należącego do planu kanałów z nielicencjonowanego pasma ISM 2.4 GHz (*Industrial, Scientific, Medical*) obliczona na podstawie wzoru i wyrażona w hercach:

$$f_c = (2{,}412 + (ch_{nr} - 1) \cdot 0{,}005) \cdot 10^9 \text{ Hz}, \tag{2.19}$$

gdzie:

ch_{nr} – numer kanału radiowego w paśmie ISM 2.4 GHz,

d^t – wypadkowa długość trasy propagacji i-tego promienia [498], [503], która dla odbicia i transmisji wynosi $d^t = d_i$, natomiast dla pojedynczej dyfrakcji na prostokątnych brzegach (framugi drzwiowe, strukturalne załamania ścian):

$$d_i^t = \frac{d_1}{d_2(d_1 + d_2)}, \tag{2.20}$$

gdzie:

d_1 – długość trasy i-tego promienia od nadajnika do punktu dyfrakcji,

d_2 – długość trasy i-tego promienia od punktu dyfrakcji do odbiornika.

W przypadku systemu LOS (*Line Of Sight*), gdy na trasie propagacji fali radiowej w odległości $d^t = d$ między umieszczonymi w próżni bezstratnymi i izotropowymi ($G_t = G_r = 1$) antenami nie występują przeszkody ($j = k = l = 0$), straty mocy PL_{FSL} (*path loss*) dla fali bezpośredniej ($N_{ray} = 1$), nazywane stratami wolnej przestrzeni FSL (*Free Space Loss*), zgodnie ze wzorem (2.17) i (2.18) wynoszą:

$$PL_{FSL}(d) = \frac{P_t}{P_r} = 10log_{10}\left(\frac{4\pi d}{\lambda}\right)^2. \tag{2.21}$$

2.9. Przegląd modeli propagacji fal radiowych w paśmie ISM 2.4 GHz

Do wyznaczenia we wnętrzach budynków wokół punktów dostępu AP sieci WLAN z infrastrukturą [504] precyzyjnych rozkładów natężenia pola elektromagnetycznego można by wykorzystać bardzo dokładne modele propagacji fal radiowych, oparte na technikach śledzenia:
- stożków [498],
- wiązek [505],
- promieni w przód FRT (*Forward Ray Tracing*) [499], [500], [506], [507],
- promieni w tył BRT (*Backward Ray Tracing*) przy zastosowaniu metody obrazów IM (*Image Method*) [498], [508],
- promieni w sposób inteligentny [509].

Ponieważ jednak wszystkie te modele wymagają bardzo długich czasów obliczeń, zatem ich stosowanie w złożonych zadaniach optymalizacji, dla potrzeb planowania sieci WLAN, może powodować zupełnie niepotrzebny, nadmierny wzrost nakładów obliczeniowych.

Problem zbyt długich czasów obliczeń tłumienia mocy sygnałów we wnętrzach budynków sprawił, że znacznie częściej i chętniej sięga się przy planowaniu sieci WLAN po jeden z następujących modeli empirycznych:
- 1SM (*One-slope Model*) [510], [511],
- LAM (*Linear Attenuation Model*) [510], [512],
- ITU-R P.1238 [504].

Jeżeli na drodze propagacji wiązki mikrofalowej występują przeszkody, w tym ściany, stropy budynku, elementy wyposażenia wnętrz, to wnoszone przez nie dodatkowe tłumienia oblicza się zazwyczaj za pomocą:
- modelu Motleya-Keenana [496] oraz jego różnych wariantów,
- modelu MWM (*Multi-Wall Model*) [513] opracowanego w ramach projektu COST 231 [504], [514],
- modeli będących rozszerzeniem lub modyfikacją modeli Motleya-Keenana i MWM, w tym w szczególności modeli Seidela-Rappaporta i Devasirvathama [515],
- modelu dominującej ścieżki DP (*Dominant Path*) [511].

2.9.1. Model *One-slope*

Model *One-Slope* (1SM) zakłada logarytmiczną zależność między długością drogi fali bezpośredniej (toru radiowego), a tłumieniem występującym na tej drodze. Jedynym parametrem empirycznym, w tym modelu, jest współczynnik propagacji n^{1SM}.

Tłumienie medianowe $L_{50}^{prop}(d)$ modelu 1SM może zostać obliczone według wzoru:

$$PL_{1SM}(d < d_0) = PL_0(d), \tag{2.22}$$

$$PL_{1SM}(d > d_0) = PL_0(d_0) + 10n_{prop}^{1SM} log_{10}\left(\frac{d}{d_0}\right), \tag{2.23}$$

gdzie:

d_0 – odległość odniesienia, wynosząca zazwyczaj 1 metr,

PL_0 – tłumienie trasy (*path loss*) zmierzone w odległości odniesienia d_0 [516],

d – odległość między anteną nadajnika a odbiornika,

n_{prop}^{1SM} – indeks odległościowego spadku mocy sygnału radiowego.

Tabela 2.1. Parametry modelu 1SM dla różnych środowisk propagacyjnych [510], [511]

Środowisko propagacyjne	PL_0 [dB]	Współczynnik propagacji n_{prop}^{1SM}
Wolna przestrzeń	40	2
Korytarz	39,2 ÷ 41	1,2 ÷2,2
Jedno duże pomieszczenie	44	2
Budynki z dużym zagęszczeniem osób, mebli, ścian	40	3,3 ÷ 4,5
Pomieszczenie na jednym piętrze (biuro)	33 ÷ 34	4,0 ÷ 4,1
Kilkupiętrowy budynek biurowy	41 ÷ 46	5,3 ÷ 5,5

Dla pomieszczeń dużych rozmiarów, w tym magazynów i hal produkcyjnych, opracowano model, stanowiący proste rozszerzenie modelu *One-slope* (1SM), w którym współczynnik propagacji n_{prop}^{1SM} przyjmuje wartości przedziału od 3,3 do 3,5 [511], a odległości odniesienia d_0 wynoszą 5 [64], a nawet 8 metrów [517].

2.9.2. Model LAM

Spośród modeli deterministycznych o współczynnikach empirycznych, do oszacowania tłumienia sygnału wewnątrz budynku bardzo często jest wykorzystywany model LAM.

Tabela 2.2. Przykładowe wartości współczynnika tłumienia α_{LAM} w modelu LAM [512]

Środowisko propagacji	Współczynnik tłumienia α_{LAM} [dB/m]
Korytarz lub jedno duże pomieszczenie	0,3
Pomieszczenia na jednym piętrze	0,7
Budynek biurowy 1 piętro	0,63 ÷1,4
Budynek biurowy (kilka pięter)	2,5 ÷ 2,8
Pomieszczenia na wielu piętrach	3,1

W modelu tym, tłumienie medianowe PL_{LAM}, wyrażone w jednostkach logarytmicznych (decybelach), oblicza się ze wzoru:

$$PL_{LAM}(d) = PL_{FSL}(d) + \alpha_{LAM}\, d, \tag{2.24}$$

gdzie:

α_{LAM} – współczynnik tłumienia, liczony w decybelach na każdy metr odległości, którego wartość zależy od charakteru i konstrukcji budynku (tab. 2.2).

2.9.3. Model ITU-R P.1238

Model ITU-R P.1238 został opracowany przez Międzynarodowy Związek Telekomunikacyjny ITU (*International Telecommunication Union*) i zaprezentowany w zaleceniu ITU-R P.1238 [504]. Umożliwia on obliczenie tłumienia mocy sygnału wewnątrz budynku w bardzo szerokim zakresie częstotliwości – od 900 MHz do 100 GHz.

Model ITU-–R P.1238 uwzględnia wpływ charakteru budynku oraz liczby stropów, występujących na trasie propagacji fali między nadajnikiem a odbiornikiem, na wartość tłumienia PL_{ITU} wyznaczanego na podstawie zależności:

$$PL_{ITU}(d) = 20log_{10}(f_c[MHz]) + Nlog_{10}(d[m]) + L_f(n_f) - 28, \quad (2.25)$$

gdzie:

N – współczynnik tłumienia,
d – odległość między antenami nadajnika i odbiornika ($d > 1$ m),
n_f – liczba stropów na drodze propagacji sygnału,
L_f – wartość tłumienia stropów (tab. 2.3),
f_c – częstotliwość środkowa kanału radiowego.

Tabela 2.3. Parametry modelu ITU–R P.1238 dla pasma ISM 2.4 GHz [504]

Typ budynku	Współczynnik tłumienia – N	L_f [dB]
Budynek mieszkalny	28	$4n_f$
Budynek biurowy	30	$15 + 4(n_f - 1)$
Budynek komercyjny Hala fabryczna	22	$6 + 3(n_f - 1)$

2.9.4. Model Motleya-Keenana

Model Motleya-Keenana jest modelem propagacyjnym przeznaczonym do obliczania tłumienia mocy sygnału w torze radiowym przebiegającym między kilkoma kondygnacjami budynku. W modelu tym uwzględnia się tylko jedną kategorię ścian/stropów [496], a tłumienie sygnału radiowego PL_{MK} wyraża się wzorem:

$$PL_{MK}(d) = PL(d_0) + 10n_{prop}^{MK} log_{10}\left(\frac{d}{d_0}\right) + n_w L_w + n_f L_f, \quad (2.26)$$

gdzie:

d_0 – odległość odniesienia, równa 1 metr,
n_{prop}^{MK} – współczynnik propagacji modelu Motleya-Keenana,
L_w –tłumienie wnoszone przez pojedynczą ścianę,
n_w– liczba ścian na drodze propagacji sygnału,
L_f– tłumienie wnoszone przez pojedynczy strop,
n_f– liczba stropów na drodze propagacji sygnału.

W literaturze funkcjonuje modyfikacja wzoru (2.26), uwzględniająca przy wyznaczaniu tłumienia PL_{MKm} dodatkowo jeszcze grubości i typy ścian, przez które przechodzi fala radiowa [64]:

$$PL_{MKm}(d) = PL(d_0) + 10n_{prop}^{MKm} log_{10}\left(\frac{d}{d_0}\right) + \sum_{i=0}^{I} n_{w_i} L_{wo_i} 2^{log_3\left(\frac{e_i}{e_{oi}}\right)}, \qquad (2.27)$$

gdzie:

d_0 – odległość odniesienia wynosząca 1 metr,
n_{prop}^{MKm} – współczynnik propagacji zmodyfikowanego modelu Motleya-Keenana,
I – liczba wszystkich typów ścian od $i = 0$ do $i = I$,
L_{wo^i} –tłumienie ściany odniesienia typu i,
n_{w^i} – liczba ścian typu i,
e_i – grubości ścian typu i,
e_{oi} – grubość ściany odniesienia typu i.

Tabela 2.4. Zestawienie współczynników tłumienia różnych materiałów, z których mogą być wykonane ściany lub stropy w budynku [518]

Rodzaje ścian lub/i jej elementów	Współczynniki tłumienia [dB]
Ściana działowa, płyta gipsowa z watą szklaną 7 cm	2
Drzwi drewniane 4 cm	2 ÷ 2,5
Ściany gipsowe	3,5
Oszklenie dwuszybowe: 2 x szyba+ 1cm przerwy	4,5
Ściany betonowe	7
Ściana zewnętrzna – cegła 30 cm	9
Ściana wewnętrzna – cegła 10 cm	7
Strop betonowy 30 cm	11 ÷ 16,2
Stropy (inne)	26,2 ÷ 46,4
Cegła perforowana 36 cm	41,5

2.9.5. Model MWM

Model MWM stanowi połączenie modelu *One–slope* i Motleya-Keenana, w którym do wyznaczenia tłumienia toru radiowego PL_{MWM} wymagana jest znajomość, nie tylko liczby ścian czy stropów, ale również wartości ich tłumień [511], [514] oraz innych przeszkód występujących na drodze propagacji sygnału. Całkowite tłumienie medianowe sygnału nośnej wyrażone jest za pomocą wzoru:

$$PL_{MWM}(d) = PL(d_0) + 10n_{prop}^{MWM} log_{10}\left(\frac{d}{d_0}\right) + \overbrace{\sum_{i=0}^{I} n_{w_i} L_{w_i}}^{WAF} + \overbrace{\sum_{j=0}^{J} n_{f_j} L_{f_j}}^{FAF} + \sum_{k=0}^{K} n_{o_k} L_{o_k}, \tag{2.28}$$

gdzie:

d_0 – odległość odniesienia wynosząca 1 metr,
n_{prop}^{MWM} – współczynnik propagacji modelu Multi Wall,
I – liczba wszystkich typów ścian,
J– liczba wszystkich typów stropów na drodze propagacji sygnału,
K – liczba wszystkich typów przeszkód, innych niż strop czy ściana,
L_{w_i}– współczynnik tłumienia ścian typu i,
n_{w_i} – liczba ścian typu i,
L_{f_j} – współczynnik tłumienia stropu typu j,
n_{f_j} – liczba stropów typu j na drodze sygnału (tab. 2.4),
WAF– tłumienie ścian (*Wall Attenuation Factor*),
FAF– tłumienie stropów (*Floor Attenuation Factor*),
L_{o_k} – współczynnik tłumienia przeszkody (*obstacle*) typu k,
n_{o^k} – liczba przeszkód typu k na drodze propagacji sygnału.

2.9.6. Model MWM – COST 231

Rozwinięciem modelu MWM (2.28) jest model wielościanowy COST 231 (*European Cooperation in Science and Technology*) w którym zakłada się, że do anteny odbiorczej dociera tylko fala bezpośrednia [504], [513].

Zgodnie z tym modelem tłumienie medianowe może zostać obliczone jako:

$$PL_{MWMc}(d)[dB] = PL(d_0) + 10n_{prop}^{MWMc} log_{10}\left(\frac{d}{d_0}\right) + \sum_{i=1}^{I} n_{w_i} L_{w_i} + n_f^{\frac{n_f+2}{n_f+1} - b_c} L_f. \quad (2.29)$$

Tabela 2.5. Wartości współczynników dla modelu MWM [510], [511], [512], [513], [519]

Nr	Środowisko propagacyjne	PL_0 [dB]	n_{prop}^{MWM}	L_{w1} [dB]	L_{w2} [dB]	L_f [dB]	b_c
1.	Pomieszczenia na jednym piętrze	34	4,1	4,1	7,5	9,1	0,5
2.	Pomieszczenia na dwóch piętrach	23	5,3				
3.	Pomieszczenia na wielu piętrach	46	5,5				
4.	Jedno duże pomieszczenie	44	2,0				
5.	Bardzo duże pomieszczenie	39	2,1				
6.	Budynki biurowe (COST 231)	37	3	3,4	6,9	18,3	0,46

W tabeli 2.5. zestawiono wartości parametrów modelu MWM (nr 1÷5) oraz COST 231 (nr 6).

2.10. Bilans energetyczny łącza radiowego wykorzystujący do planowania sieci WLAN standardu IEEE 802.11 w środowisku wewnątrzbudynkowym model propagacyjny MWM

W każdym punkcie, miejscu budynku, w którym ma być zapewniony zasięg radiowy sieci WLAN standardu IEEE 802.11, należy zagwarantować odpowiednio dużą wartość średnią mocy nośnej $P_r = \overline{RSL}$ (*Received Signal Level*), którą można wyznaczyć z zależności:

$$\overline{RSL} = \overbrace{P_t + G_t - l_t}^{EIRP} - PL + G_r - l_r, \tag{2.30}$$

gdzie:

G_r – maksymalny zysk energetyczny anteny odbiorczej,
l_r – straty mocy nośnej w fiderze odbiornika,
G_t – maksymalny zysk energetyczny anteny nadawczej,
l_t – straty mocy nośnej w fiderze nadajnika,
PL – mediana tłumienia mocy nośnej w torze radiowym,
$EIRP$ – moc zastępcza promieniowana izotropowo (*Effective Isotropic Radiated Power*).

W niniejszej monografii do obliczenia wartości mediany tłumienia PL we wzorze (3.15) zdecydowano się wykorzystać model empiryczny MWM, dla którego $PL = PL_{MWM}$ określa wzór (2.28).

Dla potrzeb niczym niezakłóconej, nieprzerwanej transmisji danych realizowanej z zadaną szybkością M_{TR}, chwilowa wartość poziomu mocy nośnej $RSL(d)$ na wejściu odbiornika stacji abonenckiej położonej w odległości d od punktu dostępu AP, powinna przewyższać, określony przez producenta dla zadanej stopy błędów BER, próg P_S^{thr} czułości odbiornika:

$$\overbrace{P_t + G_t - l_t}^{EIRP} - PL_{sig}(d) + G_r - l_r = RSL(d) > P_S^{thr}(M_{TR}, BER), \tag{2.31}$$

gdzie:

$P_s^{thr}(M_{TR}, BER)$ – czułość odbiornika gwarantująca transmisję danych z określoną szybkością M_{TR} przy zadanej stopie błędów BER,
$PL_{sig}(d)$ – wartość chwilowa tłumienia mocy nośnej wyznaczona w odległości d od punktu AP.

Ponieważ podawana przez producenta w dokumentacji technicznej, wartość czułości odbiornika $P_s^{thr}(M_{TR}, BER)$ jest wyznaczana tylko i wyłącznie w odniesieniu do szumów własnych odbiornika w temperaturze 290 K, zatem w celu jej urealnienia należy w równaniu (3.16) uwzględnić jeszcze dla konkretnego środowiska radiokomunikacyjnego dodatkowy margines mocy *FM* (*Fade Margin*) na, tzw. zaniki wolne (*slow-fading*) i wówczas:

$$\begin{aligned} EIRP - PL_{sig}(d) - FM + G_r - l_r = RSL(d) - FM = \\ = RSL_{FM}(d) > P_S^{thr}(M_{TR}, BER), \end{aligned} \tag{2.32}$$

Pomiary poziomu mocy nośnej, prowadzone w różnych środowiskach radiokomunikacyjnych [499], wskazują, że tłumienie $PL_{sig}(d)$ jest zmienną losową [516] z rozkładu logarytmiczno-normalnego (*log-normal shadowing*) o wartości średniej zależnej od odległości d stacji abonenckiej ST od punktu dostępu AP.

Dla potrzeb niniejszej monografii losowy charakter zmian tłumienia $PL_{sig}(d)$ nośnej w miejscu lokalizacji odbiornika stacji ST w przypadku propagacji wielodrogowej i przy wykorzystaniu modelu MWM – wzór (3.13) – zapisano za pomocą wzoru [499]:

$$\begin{aligned} PL_{sig}(d) = PL(d_0) + 10n_{prop}^{MWM} log_{10}\left(\frac{d}{d_0}\right) + \\ + \overbrace{\sum_{i=0}^{I} n_{w_i} L_{w_i}}^{WAF} + \overbrace{\sum_{j=0}^{J} n_{f_j} L_{f_j}}^{FAF} + \sum_{k=0}^{K} n_{o_k} L_{o_k} + X_\sigma(0, \sigma), \end{aligned} \tag{2.33}$$

gdzie:

$X_\sigma(0, \sigma)$ – zmienna losowa o rozkładzie normalnym, zerowej wartości średniej i wariancji σ^2 [dB].

Z kolei występujący we wzorze (3.17), margines mocy FM, zapewniający – z prawdopodobieństwem $U(P_S^{thr}) = 1 - Pr[RSL(d) \le P_S^{thr}(M_{TR}, BER)]$ – stacji ST, znajdującej się w odległości d od punktu AP, transmisję danych (dostępność połączenia) z szybkością M_{TR} i stopą błędów BER, wyznaczano w monografii, rozwiązując numerycznie, ze względu na FM, równanie [499], [513], [520], [520]:

$$U(P_S^{thr}) = \frac{1}{2}\left(1 - erf(a_U) + e^{\left(\frac{1-2a_U b_U}{{b_U}^2}\right)} \cdot \left[1 - erf\left(\frac{1 - a_U b_U}{b_U}\right)\right]\right), \tag{2.34}$$

gdzie:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{t^2} dt\,, \qquad a_U = \frac{FM}{\sigma\sqrt{2}}, \qquad b_U = \frac{10n_{prop}^{MWM} log_{10} e}{\sigma \cdot \sqrt{2}}\,. \tag{2.35}$$

2.11. Podsumowanie

Wybór funkcji kryterialnej (rozdz. 2) to zaledwie pierwszy krok w sformułowaniu zadania optymalizacji, umożliwiającego optymalne planowanie sieci WLAN standardu IEEE 802.11 w wybranym środowisku radiokomunikacyjnym, w tym w szczególności wewnątrzbudynkowym.

W kolejnym kroku procesu planowania bezprzewodowej sieci lokalnej należy wybrać model propagacyjny, który w sposób najlepszy z możliwych odzwierciedla właściwości torów radiowych ST-AP i AP-ST w konkretnym wewnątrzbudynkowym środowisku radiokomunikacyjnym. To właściwości toru radiowego, zależne od geometrii i materiałów, z których wykonano budynek, liczby i układu przestrzennego pomieszczeń, ich wystroju oraz przedmiotów i osób znajdujących się w budynku, decydują o zasięgach radiowych sieci WLAN, a zatem o liczbie oraz wzajemnym rozmieszczeniu i konfiguracji punktów AP tworzących infrastrukturę teletransmisyjną systemu.

2.12. Szybkość transmisji a przepustowość sieci WLAN

Do poprawnego użycia wybranych funkcji kryterialnych, przedstawionych w rozdziale drugim, istotne jest określenie nie tylko zasięgu (rozdz. 3) ale także przepustowości bezprzewodowej sieci WLAN standardu IEEE 802.11 z infrastrukturą w paśmie ISM 2.4 GHz. W tym celu w niniejszym rozdziale przedstawiono modele matematyczne umożliwiające określenie przepustowości sieci WLAN dla potrzeb zadania optymalizacji opartego na wybranej funkcji kryterialnej (rozdz. 2).

2.12.1. Szybkości transmisji danych w systemach WLAN opartych na technikach DSSS i OFDM

Technika DSSS (*Direct Sequence Spread Spectrum*) w sieciach WLAN standardu IEEE 802.11b polega na kodowaniu danych z warstwy drugiej modelu ISO/OSI pobieranych ze strumienia o szybkości 1 Mb/s i przekształceniu go przez 11-bitową sekwencję rozpraszającą w strumień porcji – czipów (*chips*) o szerokości pasma częstotliwości 11 MHz.

Dla szybkości transmisji 1 Mb/s przyjęto modulację różnicową DBPSK (*Differential Binary Phase Shift Keying*), natomiast dla 2 Mb/s – DQPSK (*Differential Quadrature Phase Shift Keying*). W przypadku DQPSK każdy symbol zawiera dwa chipy, co daje szybkość bitową 2 Mb/s. W celu osiągnięcia szybkości transmisji 5,5 Mb/s oraz 11 Mb/s w standardzie IEEE 802.11b wykorzystywano modulację CCK (*Complementary Code Keying*) z odpowiednio czterema i ośmioma bitami składającymi się na jeden symbol [453].

Natomiast w standardzie IEEE 802.11g w miejsce techniki DSSS zastosowano ortogonalne zwielokrotnienie w dziedzinie częstotliwości OFDM (*Orthogonal Frequency-Division Multiplexing*) wraz z M-wartościową kwadraturową modulacją amplitudowo-fazową MQAM (*M-ary Quadrature Amplitude Modulation*).

Tabela 2.6. Wybrane parametry standardu IEEE 802.11g

Szybkość transmisji M_{TR} [Mb/s]	$M_{TR} \in \{6; 9; 12; 18; 24; 36; 48; 54\}$
Sprawność kodu splotowego – *code rate CR*	$CR \in \{\frac{1}{2}; \frac{2}{3}; \frac{3}{4}\}$
Liczba nośnych N_c	$N_c = 52$
Liczba podnośnych pilota N_{psc}	$N_{psc} = 4$
Czas trwania symbolu OFDM T_{symbol}	$T_{symbol} = 4$ µs
Liczba strumieni przestrzennych N_s	$N_s = 1$

Szybkość transmisji danych M_{TR} (*data rate*) w standardzie IEEE 802.11g [521] można obliczyć ze wzoru:

$$M_{TR} = N_s \cdot \frac{N_{DbpC} \cdot \overbrace{(N_c - N_{psc})}^{N_{sc}} \cdot CR}{T_{symbol}} = N_s \cdot \frac{N_{DbpS}}{T_{symbol}}, \tag{2.36}$$

gdzie:

N_{DbpS} – liczba bitów danych składających się na symbol (tab. 2.7),
N_{DbpC} – liczba bitów na jednej nośnej (tab. 2.7),
N_{sc} – liczba podnośnych danych informacyjnych,
N_c – liczba nośnych,
N_{psc} – liczba podnośnych pilota (tab. 2.6),
CR – sprawność *Code Rate* kodu splotowego,
T_{symbol} – czas trwania jednego symbolu OFDM.

Tabela 2.7. Zestawienie wybranych parametrów standardów IEEE 802.11 b/g

M_{TR} [Mb/s]	54	48	36	24	18	12	11	9	6	5,5	2	1
M_{TR} [Mb/s]	24	24	24	24	12	12	2	6	6	2	2	1
N_{DbpS} [b]	216	192	144	96	72	48	n/d	36	24	n/d	n/d	n/d
N_{DbpC} [b]	6	6	4	4	2	2	n/d	1	1	n/d	n/d	n/d

n/d – nie dotyczy

Z danych zamieszczonych w tabelach 2.6 i 2.7 wynika, że maksymalne szybkości transmisji M_{TR}^{max}, możliwe do osiągnięcia w sieciach WLAN z infrastrukturą, nie przekraczają odpowiednio, dla standardu IEEE 802.11b wartości M_{TR}^{max} =11 Mb/s oraz dla IEEE 802.11g M_{TR}^{max} =54 Mb/s. Wybrane parametry standardów IEEE 802.11n [472], IEEE 8011ac [473], IEEE 8011ax [474] przedstawiono w tabeli 2.8.

Tabela 2.8. Wybrane parametry standardów: IEEE 802.11n, IEEE 8011ac, IEEE 8011ax, IEEE 8011be

Standard / wybrane parametry	IEEE 802.11n	IEEE 802.11ac	IEEE 802.11ax	IEEE 802.11be
Maksymalna liczba strumieni przestrzennych N_s	**4**	3 **8**	**8**	8 **16**
Liczba nośnych N_c (szerokośc pasma)	56 (20 MHz) **114** (40 MHz)	242 (80 MHz); **484** (160, 80+80 MHz)	996 (80 MHz) **1976** (160 MHz)	**3936** (320 MHz)
Liczba podnośnych pilota N_{psc}	{4; **6** }	{8; **16**}	16	16
Guard Interval GI [µs]	0,8; **0,4**	**0,4**	**0,8**/1,6/3,2	**0,8**/1,6/3,2
Czas trwania symbolu OFDM T_{symbol} [µs]	3,2+GI	3,2+GI	12,8+GI	12,8+GI
Sprawność kodu splotowego – *code rate CR*	$\{\frac{1}{2}; \frac{2}{3}; \frac{3}{4}; \mathbf{\frac{5}{6}}\}$	$\{\frac{1}{2}; \frac{2}{3}; \frac{3}{4}; \mathbf{\frac{5}{6}}\}$	$\{\frac{1}{2}; \frac{2}{3}; \frac{3}{4}; \mathbf{\frac{5}{6}}\}$	$\{\frac{1}{2}; \frac{2}{3}; \frac{3}{4}; \mathbf{\frac{5}{6}}\}$
Maksymalna N_{DbpC} [b] /modulacja	**6** / 64−QAM	**8** / 256−QAM	**10** / 1024−QAM	**12** / 4096−QAM
Maksymalna szybkość transmisji M_{TR}^{max} [Mb/s]	**600**	2600 (3 N_s) wave 1 **6933,3** (8 N_s) wave 2	9607,8	23059 **46118**

2.12.2. Modele matematyczne przepustowości sieci WLAN standardu IEEE 802.11

Z informacji zamieszczonych w dokumencie RFC 2544 [522] przepustowość (*throughput*) sieci transmisji danych cyfrowych można zdefiniować jako „liczbę bitów przesłanych w ciągu jednej sekundy przez urządzenie DUT (*Device Under Test*) lub przez sieć bez utraty danych czy gubienia ramek".

Tabela 2.9. Zestawienie wybranych parametrów sieci WLAN

Nazwa parametru	Wartość parametru
Analizowany rozmiar ładunku pakietu danych $L_{DATA\_payload}$	$L_{DATA\_payload}$ = 1472 B oraz ∈{64; 128; 512; 1024} B
Parametry obliczeniowe dla protokołu CSMA/CA – funkcja DCF	
Minimalny rozmiar okna rywalizacji CW_{min}	CW_{min} ∈ {3; 7; 15; 31}
Liczba ponowień procedury *backoff* – m	m ∈ {1; 2; 5; 6}
Maksymalny rozmiar okna rywalizacji CW_{max}	CW_{max} ∈ {15; 31; 63; 1023}
Limit ponowień transmisji ramek Data L_r	$L_r = 7$

Tabela 2.10. Zestawienie wybranych parametrów i wzorów obliczeniowych dla sieci WLAN standardu IEEE 802.11b

Nazwa parametru		Wartość parametru
Długość podstawowej szczeliny czasowej T_{slot}		T_{slot} = 20 µs
Szybkość transmisji danych sterujących (*control rate*) M_{CR}	M_{CR_1}	M_{CR_1} =1 Mb/s
	M_{CR_2}	M_{CR_2} =2 Mb/s
Rozmiar nagłówka PHY L_{PHY}/ PLCP L_{PLCP}		L_{PHY} =6 B / L_{PLCP} =18 B
Czas trwania nagłówków: PHY i PLCP $T_{PHY_{hdr}}$	krótka preambuła	$(L_{PHY}/M_{CR_2} + L_{PLCP}/M_{CR_1})$=96 µs
	długa preambuła	$(L_{PHY} + L_{PLCP})/M_{CR_1}$=192 µs
Czas trwania pakietu potwierdzenia T_{ACK}		$T_{ACK} = T_{PHY_{hdr}} + L_{ACK}/M_{CR}$
Czas trwania całego pakietu danych T_{DATA}		$T_{DATA} = T_{PHY_{hdr}} + (L_{DATA})/M_{TR}$
Odstęp czasu *Extended InterFrame Space* T_{EIFS}		$T_{EIFS} = T_{SIFS} + T_{DIFS} + T_{ACK}$=364 µs

Efektywny próg fragmentacji pakietów w sieciach transmisji danych ustalany jest w warstwie sieciowej za pomocą jednostki *MTU* (*Maximum Transfer Unit*), która określa maksymalny rozmiar pakietu dla danego typu sieci [523]. I tak dla sieci typu Ethernet (IEEE 802.3) jednostka *MTU* wynosi 1500 B [524].

Rozmiary pakietów przesyłanych w ramce Ethernet (IEEE 802.3) mogą przyjmować wartości z przedziału od 64 do 1518 B i nie zawierają preambuły oraz pola znacznika początku ramki SFD (*Start Frame Delimiter*), które łącznie liczą 8 B. Funkcjonują także rozwiązania zwiększające MTU – tzw. *jumbo frames* (9000 B) lub *super jumbo frames* (SJF) o teoretycznej wartości aż do 64 kB.

W tabelach 2.9÷2.12 przedstawiono najważniejsze parametry oraz wybrane wzory obliczeniowe przydatne w zagadnieniach modelowania i planowania sieci WLAN standardów IEEE 802.11b i IEE 802.11g [525], [526].

Tabela 2.11. Zestawienie wybranych parametrów i wzorów obliczeniowych dla sieci WLAN standardu IEEE 802.11g

Nazwa parametru	Wartość parametru
Szczelina czasowa T_{slot}	T_{slot} =9 µs lub 20 µs
Czas transmisji symbolu OFDM T_{symbol}	T_{symbol} =4 µs
Odstęp czasu *Extended Interframe Space* T_{EIFS}	$T_{EIFS} = T_{ACK_{max}}+T_{SIFS}+T_{DIFS}$
Czas transmisji preambuły PHY T_{PRE}	T_{PRE} =16 µs
Czas transmisji nagłówka PHY T_{PHY}	T_{PHY} =4 µs
Rozmiar pola SERVICE/TAIL L_{ser}/ L_{tail}	16 b /6 b
Rozmiar nagłówka UDP/ IP L_{UDP}/ L_{IP}	64 b /160 b
Czas trwania pakietu potwierdzenia T_{ACK}	$T_{PRE}+T_{PHY} + T_{symbol} \cdot \left\lceil \frac{L_{ser}+L_{tail}+L_{ACK}}{L_{DbpS}} \right\rceil$
Czas transmisji pakietu danych T_{DATA}	$T_{PRE}+T_{PHY} + T_{symbol} \cdot \left\lceil \frac{L_{ser}+L_{tail}+L_{DATA}}{L_{DbpS}} \right\rceil$
Odstęp czasu *Extended Interframe Space* T_{EIFS}	$T_{EIFS} = T_{SIFS} + T_{DIFS} + T_{ACK}$=160 µs

Procedura dzielenia pakietów danych, czyli ich fragmentacji, pogarsza przepustowość sieci WLAN, co jest naturalną konsekwencją konieczności przesyłania oprócz fragmentów danych źródłowych także ramek zawierających dane sygnalizacyjne, w tym np. pakiety potwierdzeń oraz dane nagłówkowe wynikające z samej budowy ramek. Rośnie również prawdopodobieństwo utraty pakietów. Stąd w przypadku protokołu IP, dla wszystkich urządzeń biorących udział w wymianie danych, określa się za pomocą techniki PMTUD (*Path MTU Discovery*) [527] najmniejszą wartość jednostki *MTU* [524].

Tabela 2.12. Wspólne parametry i wzory obliczeniowe dla sieci WLAN standardów IEEE 802.11b i IEEE 802.11g

Nazwa parametru	Wartość parametru
Odstęp czasu *Short InterFrame Space* T_{SIFS}	$T_{SIFS} = 10$ µs
Odstęp czasu *DCF InterFrame Space* T_{DIFS}	$T_{DIFS} = T_{SIFS} + 2 \cdot Tslot$
Rozmiar długiego nagłówka MAC $L_{MAC}hdr$	$L_{MAC}hdr = 34$ B
Rozmiar nagłówka MAC pakietu ACK $L_{MAC\_ACK}hdr$	$L_{MAC\_ACK}hdr = 14$ B
Limit czasu nieudanej transmisji $T_{ACK}timeout$	$T_{ACK}timeout = T_{SIFS} + TACK$
Rozmiar pakietu danych L_{DATA}	$L_{DATA} = L_{MAC}hdr + L_{DATA\_payload}$
Rozmiar pakietu potwierdzeń L_{ACK}	$L_{ACK} = L_{MAC\_ACK}hdr$

Dla potrzeb niniejszej monografii, w celu określenia maksymalnej przepustowości analizowanych w niej sieci WLAN, wybrano pakiety danych z ładunkiem $L_{DATA\_payload}$ o rozmiarze 1472 B. Rozmiar pakietu z takim ładunkiem znajduje się w zakresie, w którym może zostać ustalona granica fragmentacji (2346 B) dla sieci WLAN standardu IEEE 802.11 oraz wynika z jego optymalnej wielkości dla rozważanych w monografii sieci WLAN [528].

2.12.3. Górna granica przepustowości sieci WLAN

W celu zdefiniowania górnej granicy przepustowości TUL (Throughput Upper Limit) sieci WLAN [531], [532], [533], [534] założono: brak strat pakietów wskutek kolizji, zerową wartość bitowej stopy błędów BER, ciągłą – niczym nie przerywaną – transmisję pakietów, brak fragmentacji pakietów w podwarstwie MAC (Media Access Control), pomijalnie krótkie czasy trwania ramek zarządzających pracą sieci. Wówczas dla podstawowego typu dostępu do łącza danych i protokołu CSMA/CA, górna granica przepustowości S_TUL sieci WLAN – dla danego rozmiaru L_(DATA_payload) ładunku ramki MSDU – jest stała i wyraża się zależnością [528]:

$$\overline{CW} = \frac{CW_{min} \cdot T_{slot}}{2}. \tag{2.37}$$

gdzie:

CW – średni rozmiar okna rywalizacji dla procedury backoff DCF określony jako:

$$S_{TUL} = \frac{8 \cdot L_{DATA\_payload}}{T_{DATA} + T_{SIFS} + T_{ACK} + T_{DIFS} + \overline{CW}}, \tag{2.38}$$

2.12.4. Prosty model matematyczny przepustowości sieci WLAN

W przypadku sieci WLAN z N_{ST} stacjami ST, pracującymi z taką samą szybkością transmisji M_{TR}, całkowitą przepustowość S_s takiej sieci można opisać za pomocą prostego deterministycznego modelu matematycznego [532], [533]:

$$S_S = M_{TR} \cdot p_S \cdot U_S, \tag{2.39}$$

gdzie:

p_S – współczynnik zdefiniowany w [532] jako:

$$p_S = \frac{T_{DATA}}{T_p} \cdot \frac{L_{DATA_{payload}}}{L_{DATA}}, \tag{2.40}$$

U_S – prawdopodobieństwo dostępu do kanału radiowego – liczone na dostatecznie długim horyzoncie czasu – równe:

$$U_S = \frac{T_p}{N_{ST} \cdot T_p + T_{jam} \cdot (N_{ST} - 1)}. \tag{2.41}$$

Do wyznaczenia, na podstawie wzoru (2.41), wartości prawdopodobieństwa U_S wymagana jest znajomość czasu T_{jam}, reprezentującego, tzw. średnią stratę czasu transmisji, wynikającą ze zjawiska kolizji pakietów, który można obliczyć ze wzoru:

$$T_{jam} = \left(1 - \frac{2}{N_{ST} + 1}\right) \cdot T_p, \tag{2.42}$$

Jeżeli założyć, że czas propagacji sygnału między punktem AP a stacją ST jest do pominięcia (δ = 0), to całkowity czas transmisji pakietu T_p wyniesie:

$$T_p = T_{ov} + T_{DATA} + T_{cont}, \tag{2.43}$$

gdzie:

T_{ov} – czas trwania wszelkiego typu nagłówków występujących w ramce – tzw. narzutu (*overhead*) związanego z transmisją pojedynczego pakietu – równy:

$$T_{ov} = T_{DIFS} + T_{SIFS} + T_{ACK}, \tag{2.44}$$

T_{cont} – czas trwania procedury rywalizacji (*contention*) stacji ST o dostęp do kanału radiowego, który można oszacować na podstawie wzoru [533]:

$$T_{cont} \cong T_{slot} \cdot \frac{1 + p_c}{2 \cdot N_{ST}} \cdot \frac{CW_{min}}{2}, \tag{2.45}$$

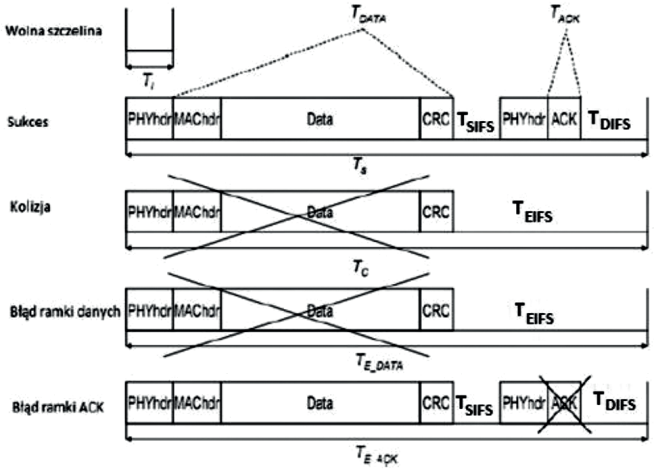
gdzie:

p_c – prawdopodobieństwo wystąpienia kolizji, określone na podstawie liczby poprawnie odebranych pakietów potwierdzeń w podwarstwie MAC, równe [532]:

$$p_c = 1 - \left(1 - \frac{1}{CW_{min}}\right)^{N_{ST}-1}. \tag{2.46}$$

2.12.5. Stochastyczny model matematyczny przepustowości sieci WLAN

Do wyznaczenia przepustowości sieci WLAN standardu IEEE 802.11, w niniejszej monografii, wykorzystano także stochastyczny model matematyczny bazujący na teorii łańcuchów Markowa.



Rysunek 2.10. Pięć stanów kanału radiowego [534]

W modelu opartym na procesach Markowa z czasem dyskretnym całkowitą przepustowość S_M sieci wyznaczono na podstawie wzoru:

$$S_M = \frac{E\left[L_{DATA\_payload}\right]}{E[T]}, \tag{2.47}$$

gdzie:

$E[L_{DATA_{payload}}]$ – średni rozmiar ładunku wszystkich poprawnie odebranych pakietów,

$E[T]$ – średni czas trwania pięciu stanów kanału radiowego (rys. 2.10), występujących podczas jego użytkowania i obliczane ze wzoru:

$$E[T] = \overbrace{\underbrace{(1-p_{tr})}_{p_{idle}} \cdot T_{slot}}^{wolny\ kanał} + \overbrace{p_{tr} \cdot p_s \cdot (1-p_e) \cdot T_s}^{udana\ transmisja} + \overbrace{p_{tr} \cdot (1-p_s) \cdot T_c}^{transmisja\ z\ kolizją} + \\ + \overbrace{\underbrace{p_{tr} \cdot p_s \cdot p_e \cdot T_e}_{(błędy\ w\ ramce\ danych\ lub\ błąd\ ramki\ ACK)}}^{nieudana\ transmisja}, \tag{2.48}$$

gdzie:

p_e – prawdopodobieństwo błędnego odbioru pakietu [535], [536],

p_{idle} – prawdopodobieństwo wystąpienia stanu, w którym kanał radiowy jest wolny,

p_s – całkowite prawdopodobieństwo udanej transmisji, wyrażające się zależnością:

$$p_s = \frac{\sum_{i=0}^{L_{TR}-1} p_{str_i}}{p_{tr}} = \frac{\sum_{i=0}^{L_{TR}-1} N_{ST_i} \cdot \tau_i \cdot (1-\tau_i)^{N_{ST_i}-1} \cdot \prod_{j=0, i\neq j}^{L_{TR}-1} \left(1-\tau_j\right)^{N_{ST_j}}}{1-\prod_{i=0}^{L_{TR}-1}(1-\tau_i)^{N_{ST_i}}} \tag{2.49}$$

gdzie:

p_{str_i} – prawdopodobieństwo udanej transmisji dla i-tej szybkości transmisji TR,

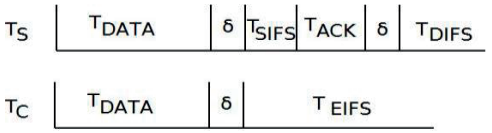
τ_i – prawdopodobieństwo udanej transmisji ramki,

N^i – liczba aktywnych stacji ST, realizujących transmisję z i-tą TR,

L_{TR}=12 – liczba rozważanych szybkości transmisji TR,

p_{tr} – prawdopodobieństwo wystąpienia stanu, w którym kanał radiowy jest zajęty, równe:

$$p_{tr} = 1 - \prod_{i=0}^{L_{TR}-1} (1-\tau_i)^{N_{ST}^i}. \tag{2.50}$$



Rysunek 2.11. Diagramy czasowe podstawowego trybu DCF

Dla podstawowego trybu dostępu DCF [526], [537], czasy trwania: T_s – udanej transmisji [538], T_c – transmisji z kolizją oraz T_e – nieudanej transmisji (z błędami) wynosiły odpowiednio (rys. 2.11):

$$T_s = T_{DATA} + T_{SIFS} + T_{ACK} + T_{DIFS} + 2 \cdot \delta, \tag{2.51}$$

$$T_c = T_e = T^*_{DATA} + \overbrace{T_{DIFS} + T_{ACK_{timeout}}}^{T_{EIFS}} + \delta, \tag{2.52}$$

gdzie:

T^* – średni czas transmisji pakietów biorących udział w kolizji,

T_{EIFS} – opóźnienie stosowane przez stację ST po odebraniu pakietu z błędem, którego nie potrafi zinterpretować,

δ – czas propagacji sygnału między punktem AP a stacją ST.

Znając wartości czasów oraz prawdopodobieństw we wzorze (2.48), całkowitą przepustowość sieci, można wyrazić wzorem:

$$S_M = \frac{\sum_{i=1}^{L_{TR}} p_{str_i} \cdot E\left[L_{DATA\_payload_i}\right] \cdot (1 - p_e)}{(1 - p_{tr}) \cdot T_{slot} + \sum_{i=1}^{L_{TR}} p_{str_i} \cdot T_{s_i+} + p_{tr}\left(1 - \sum_{i=1}^{L_{TR}} p_{str_i}\right) \cdot T_c}, \tag{2.53}$$

gdzie:

T_{si} – średni czas trwania udanej transmisji pakietu strumienia ruchu i–tej TR.

Do wyznaczania całkowitej przepustowość S_M sieci WLAN standardu IEEE 802.11 z infrastrukturą, liczonej według wzoru (2.53), wykorzystano dwa modele matematyczne oparte na łańcuchach Markowa ze statycznym wyborem trybu pracy (*multi–rate*) [539], [540]. Pierwszy z nich $DCF_1(m; CW_{min})$ stanowił proste rozwinięcie modelu Bianchiego dla funkcji DCF [490]. Po uwzględnieniu w modelu $DCF_1(m; CW_{min})$ właściwości protokołu CSMA/CA oraz rozszerzeniu go o limit ponowień L_r nadawania ramek i o ograniczenie rozmiaru okna rywalizacji [541], [542], powstał drugi model $DCF_2(m; CW_{min}; L_r)$.

Oba modele szacowania przepustowości zostały szczegółowo opisane w artykule [492].

W pracy tej znalazły się także zależności umożliwiające obliczanie wartości poszczególnych prawdopodobieństw, występujących we wzorze (2.48).

Wyznaczona na podstawie wzoru (2.53) przepustowość sieci osiąga maksimum w przypadku, między innym, bezbłędnej transmisji oraz dla maksymalnych rozmiarów pakietów danych, czyli takich, które nie ulegają fragmentacji na żadnym etapie transmisji w sieci.

2.12.6. Dokładne oszacowanie wymaganej liczbu punktów dostępu (AP)

Dokładne oszacowanie wymaganej liczby N_{AP} punktów dostępu AP dla zadanej liczby stacji abonenckich N_{ST} [484] można przeprowadzić, bazując na ustaleniach dotyczących górnej granicy przepustowości TUL (*Throughput Upper Limit*) (2.37). Wartość ta może być obliczona ze statystycznie otrzymanego wzoru [484]:

$$N_{AP} = \left\lceil N_{ST} \cdot \left(\frac{R_{MAC}}{0{,}983 \cdot M_{TR}^{max} \cdot \eta_1} \right)^{0{,}894} \right\rceil, \tag{2.54}$$

gdzie:

M^{max}– maksymalna szybkość transmisji ST/AP,

η_1 – efektywność wykorzystania punktu AP [484],

R_{MAC} – wymagana szybkość transmiji w podwarstwie MAC (*Media Access Control*).

Tabela 2.13. Efektywność transmisji dla różnych standardów sieci WLAN [484]

Standard IEEE	802.11b	802.11b/g	802.11g	802.11a	802.11n (20MHz)	802.11n (40MHz)
M_{TR}^{max} [Mb/s]	11	54	54	54	288,9	600
η_1	0,62	0,54	0,79	0,80	0,99	0,98

2.13. Podsumowanie

Szacowanie przepustowości sieci WLAN, obok określenia zasięgu sieci, jest ważnym zagadnieniem, koniecznym do rozwiązania zadania optymalizacji wykorzystującego adekwatną funkcję kryterialną (rozdz. 2).

W celu obliczenia funkcji kryterialnych: F_{C_4} (2.8), F_{C_6}(2.11) czy F_{C_8}(2.15) wymagana jest znajomość przepustowosci sieci, która jest determinowana protokołem wielodostępu do kanału radiowego – CSMA/CA.

W rozważaniach analizowane zostały rozwiązania wykorzystujące standardy sieci WLAN pracujące w paśmie ISM 2.4 GHz. Przedstawione zostały wzory umożliwiające obliczenia maksymalnej przepustowości sieci (prosty model przepustowości, TUL).

Najbardziej rozbudowanym, ale także najbardziej dokładnym z analizowanych modeli, jest ten wykorzystujący łańcuchy Markowa. W modelu tym można uwzględnić statyczny wybór trybu pracy (*multi-rate*) stacji ST.

W celu ograniczenia wpływu technik sterowania przepływem i retransmisją pakietów na przepustowość sieci, w monografii rozważano tylko i wyłącznie transmisję danych z protokołem UDP. Tym samym otrzymane przy wykorzystaniu tego protokołu wartości przepustowości sieci WLAN były wartościami największymi z możliwych do osiągnięcia w tego typu sieciach [491].

W sieciach WLAN z infrastrukturą, obecność stacji ST o znacząco małych szybkościach transmisji M_{TR} prowadzi do istotnego pogorszenia przepustowości. Stacje ST używające z konieczności małych szybkości transmisji ograniczają bowiem możliwości transmisyjne wszystkich pozostałych stacji ST, które mogłyby nadawać z dużo większymi szybkościami.

W kontekście tego zjawiska, nazywanego anomalią wydajności (*Performance Anomaly*) sieci WLAN [532], [533] dużego znaczenia nabiera staranne planowanie sieci WLAN pod kątem dostępnego na rynku sprzętu Wi-Fi oraz wyboru trybu pracy stacji ST. I właśnie taki rodzaj planowania sieci WLAN z infrastrukturą w zakresie 2.4 GHz jest możliwy do przeprowadzenia za pomocą funkcji kryterialnych zaprezentowanych w rozdziale drugim (2.6 i 2.7) niniejszej monografii.

3. ANALIZA I OCENA PORÓWNAWCZA WYBRANYCH ALGORYTMÓW OPTYMALIZACJI W PLANOWANIU SIECI WLAN STANDARDU IEEE 802.11

W niniejszym rozdziale, w jego podrozdziałach 3.1 i 3.2 porównano, omówione w rozdziale pierwszym monografii, wybrane metody optymalizacji jednokryterialnej oraz wielokryterialnej.

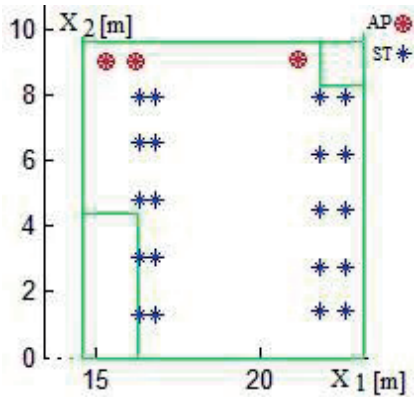
Przeprowadzone porównania, przedstawionych w rozdziale szóstym metod optymalizacji, miały na celu wskazanie jednego, bądź kilku algorytmów optymalizacji OPA, które najlepiej sprawdziłyby się w zadaniach planowania, projektowania i eksploatacji sieci WLAN z infrastrukturą. Algorytmy OPA (rozdz. 1) porównywano w wybranych środowiskach radiokomunikacyjnych, dla z góry zadanej liczby stacji ST obsługiwanych przez określoną liczbę punktów AP, na drodze wyznaczenia:

- wartości funkcji kryterialnej dla różnych parametrów algorytmów OPA,
- niezbędnej liczby iteracji, koniecznych do zakończenia działania poszczególnych algorytmów OPA,
- czasów prowadzonych obliczeń dla porównywanych algorytmów OPA.

Dla potrzeb analizy porównawczej opracowano autorski program komputerowy umożliwiający wyznaczenie ekstremów funkcji kryterialnych zdefiniowanych w rozdziale drugim monografii.

3.1. Porównanie algorytmów optymalizacji jednokryterialnej

Analizę porównawczą wybranych algorytmów optymalizacji jednokryterialnej SOO (*Single Objective Optimisation*) przeprowadzono dla sieci WLAN IEEE 802.11 w środowisku radiokomunikacyjnym przedstawionym na rysunku 3.1, w którym w jednej sali działało 20 stacji ST, komunikujących się z 3 punktami AP, realizującymi transmisję danych w trzech niezakłócających się kanałach radiowych pasma ISM 2.4 GHz o numerach $ch_{nr} \in \{1, 6, 11\}$.



Rysunek 3.1. Rzut poziomy sali z 20 stacjami ST oraz trzema przykładowo rozlokowanymi, punktami AP dla scenariusza SOO_{3AP}^{20ST}

W ramach zadania optymalizacji jednokryterialnej, sformułowanego dla potrzeb planowania sieci WLAN (tab. 3.1) o 20 stacjach ST i 3 punktach AP (rys. 3.1), czyli dla scenariusza SOO^{20ST}, porównano ze sobą sześć rojowych algorytmów OPA: ABC, BA, BeA, CS, FA, PSO oraz trzy inne algorytmy OPA: SA, GA, GA/NM (tab. 3.2). Jako funkcję kryterialną wykorzystano, zapisaną wzorem (2.8), opartą na przepustowości grupy stacji abonenckich ST, funkcję $F_{c_4} = \sum_{j=1}^{N_{AP}} \sum_{k=1}^{L_{TR_j}} \frac{S_k}{N_{ST_{jk}}} \left(d_{jk}^2 - d_{j(k-1)}^2\right)$.

Tabela 3.1. Dane techniczne punktów dostępu AP i stacji abonenckich ST w analizowanej sieci WLAN

Sieć WLAN standardu IEEE 802.11b/g	
Czułości P_S^{thr} odbiornika stacji ST, gwarantujące na jego wyjściu bitową stopę błędów $BER \leq 10^{-5}$ i odpowiadające im szybkości transmisji M_{TR}	$P_S^{thr} = \{-90;\ -89;\ -88;-87;\ -86;\ -85;\ -84;\ -82;\ -79;\ -75;\ -72;\ -68\}$ dBm
	$M_{TR} = \{1{,}0;\ 2{,}0;\ 5{,}5;\ 6{,}0;\ 9{,}0;\ 11{,}0;\ 12{,}0;\ 18{,}0;\ 24{,}0;\ 36{,}0;\ 48{,}0;\ 54{,}0\}$ Mb/s
Moc zastępcza promieniowana izotropowo *EIRP* puntu dostępu AP	$EIRP$ = 20 dBm
Rozmiar ładunku danych $L_{DATA\_payload}$	$L_{DATA\_payload}$ =1472 B

W tym miejscu należy zaznaczyć, że pomimo pewnych ograniczeń samej metody NM [64], co do jej zastosowania w analizowanych w tym rozdziale zadaniach optymalizacji, zdecydowano się jednak mimo wszystko wykorzystać ją w hybrydowym algorytmie genetycznym GA/NM.

Tabela 3.2. Dane opisujące zadanie optymalizacji dla scenariusza SOO_{3AP}^{20ST}

Wymiar problemu N_D	N_D=$N_{AP} \cdot 2$
Kryterium stopu OPA– maksymalna liczba iteracji	N_{iter}^{max} = 1000 oraz 300 dla BeA
ABC – sztucznej kolonii pszczół	
Liczba pszczół N_{pop}	N_{pop}=50
Górna granica współczynnika przyspieszenia a_{ABC}	a_{ABC}={0,10; 0,50; 0,90; 1,00}
BA – algorytm nietoperza	
Liczba nietoperzy N_{bat}=	N_{bat}= 50
Szybkość emisji impulsów *Pulse rate* r_i	r_i={0,50; 0,70; 1,00}
Głośność A_i	A_i={0,00; 0,50; 0,70}
BeA – algorytm pszczeli	
Liczba pszczół skautów N_{sb}	N_{sb}=50
Liczba: wybranych miejsc N_{ss} i pszczół do wybranych miejsc N_{ssb}	$N_{ss} = \lfloor 0{,}5N_{sb} \rfloor$, $N_{ssb} = \lfloor 0{,}5N_{sb} \rfloor$
Liczba elitarnych miejsc N_{es} i pszczół do elitarnych miejsc N_{esb}	$N_{es} = \lfloor 0{,}4N_{ss} \rfloor$, N_{esb}=$2N_{ssb}$
Promień sąsiedztwa	$0{,}1(X_{max} - X_{min})$
CS – algorytm kukułki	
Liczba gniazd N_{nest}	N_{nest} = 50
Prawdopodobieństwo wykrycia jaja podrzuconego przez kukułkę p_a	p_a = {0,25; 0,30; 0,35; 0,40; 0,45; 0,46; 0,47; 0,48; 0,49; 0,50}
FA – algorytm świetlika	
Liczba świetlików N_{ff}	N_{ff}=50
Parametr losowości α_{ff}	α_{ff} = {0,20; 0,50; 1,00}
Współczynnik „atrakcyjności" odniesienia β_0	β_0={0,10; 0,20; 1,00}
Współczynnik absorpcji γ	γ ={0,01; 0,10; 1,00 10,00}
PSO – algorytm ptasi	
Liczebność „roju" N_{pso}	N_{pso}=50
Współczynnik inercji ω	ω= 1,00
Współczynnik poznawczy *Cognition Coefficient* c_1	c_1= {1,00; 1,50; 1,60; 1,70; 1,80; 1,90; 2,00;}
Współczynnik społeczny *Social Coefficient* c_2	c_2 = {1,00; 1,50; 2,00; 2,10; 2,20; 2,30; 2,50; 3,00}
SA – symulowane wyżarzanie	
Parametry algorytmu	T_{min}=10^{-6}; T_{max}=10^2; r_t = 0,999; L_{epo}=N_{AP}
GA – algorytm genetyczny	
Oddziaływanie krzyżowania na populację (*crossover fraction*) c_p	$c_p = 0{,}90$
Liczba generacji n_{gen}	$n_{gen} = 50000$
Prawdopodobieństwo mutacji m_u	$m_u = 0{,}01$
GA z NM	
Domyślne ustawienia parametrów w pakiecie *Matlab Global Optimisation Toolbox*	

Przy doborze parametrów dla wszystkich dziewięciu algorytmów OPA (tab. 3.2) oparto się na wynikach prac: [9], [10], [12], [30], [494], [543].

Tabela 3.3. Współrzędne 3 punktów AP oraz wartości funkcji F_{C_4} zadania optymalizacji dla scenariusza SOO_{3AP}^{20ST}

Algorytm	Współrzędne trzech AP [m]		N_{ST}	Wartość funkcji F_{c_4} [(Mb/s) m^2]
	$X_{1_{AP}}$	$X_{2_{AP}}$		
GA	17,95	8,95	1	856
	17,64	7,56	13	
	21,45	9,22	6	
GA/NM	21,83	9,56	1	2123
	22,96	8,30	11	
	15,82	8,90	8	
BA (r_i =1,00 A_i =0,00)	21,11	0,01	9	2438
	22,44	0,92	10	
	21,63	0,44	1	
ABC (a_{ABC} =1,00)	16,45	9,60	14	2599
	14,96	8,24	1	
	14,87	8,08	5	
BeA	14,55	9,52	1	3699
	14,60	9,46	11	
	21,83	9,52	8	
FA (α_{ff} = 1,00; β_0=1,00; γ= 1,00)	21,83	9,56	8	3783
	14,55	9,62	1	
	14,61	9,58	11	
PSO (c_1 = 1,80; c_2 = 2,20)	14,55	9,62	1	3783
	21,83	9,62	8	
	14,70	9,57	11	
SA	21,83	9,62	8	3787
	14,55	9,62	1	
	14,62	9,59	11	
CS (p_a = 0,50)	15,71	9,62	**1**	4391
	16,28	9,62	**1**	
	16,33	9,57	**18**	

Z tabeli 3.3 wynika, że tylko algorytm kukułki CS doprowadził do innego niż pozostałe algorytmy przydziału stacji ST poszczególnym trzem punktom AP. Ponadto dla algorytmu CS osiągnięto najlepszą, bo największą wartość funkcji kryterialnej F_{c_4}.

Następnie postawiono hipotezę zerową o braku istotnych różnic między największymi (najlepszymi) wartościami funkcji kryterialnej F_{c_4}, uzyskanymi dla poszczególnych par porównywanych ze sobą algorytmów z tabeli 3.2. Tak sformułowaną hipotezę zweryfikowano za pomocą testu Wilcoxona (tab. 3.4).

Z analizy wartości testu T wynika, że hipoteza ta powinna zostać odrzucona dla wszystkich par algorytmów, dla których prawdopodobieństwa p<0,95. A to z kolei oznacza, że algorytm kukułki CS – dla poziomu ufności 95% – należy uznać za najlepszy ze wszystkich analizowanych w zadaniu optymalizacji dla scenariusza SOO_{3AP}^{20ST}.

Tabela 3.4. Wyniki testów T kolejności par Wilcoxona dla najlepszych rozwiązań uzyskanychw ramach scenariusza SOO_{3AP}^{20ST} za pomocą algorytmów OPA (tab. 3.2) i 10 ważnych prób

Algorytm nr 1 vs nr 2	Wartość testu T	Wartość prawdopodobieństwa p
CS vs {pozostałe OPA z tab. 8.3}	0,000	0,0051
CS vs PSO	1,000	0,0069
CS vs BeA	2,000	0,0093
CS vs FA	3,000	0,0125
CS vs SA	5,000	0,0218

Analizę porównawczą algorytmów OPA, wymienionych w tabeli 3.2, przeprowadzono także dla sieci WLAN IEEE 802.11 w środowisku radiokomunikacyjnym zaprezentowanym na rysunku 3.2, w którym w kilku pomieszczeniach oraz na korytarzu jednego piętra budynku znajdowały się 74 stacje ST, komunikujące się z 3 punktami AP (scenariusz SOO_{3AP}^{74ST}), pracujące w trzech niezakłócających się kanałach radiowych pasma ISM 2.4 GHz o numerach $ch_{nr} \in \{1, 6, 11\}$.

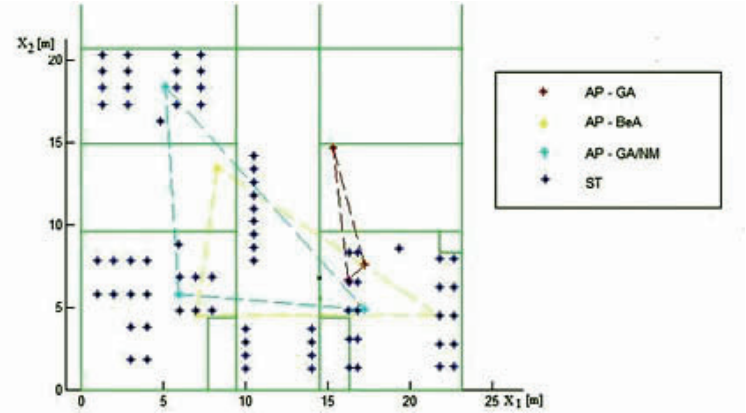
Dla scenariusza SOO_{3AP}^{74ST} porównano ze sobą – w ramach zadania poszukiwania minimum zasięgowej funkcji kryterialnej $F_{c_1} = \sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} d_{ji}$, zapisanej wzorem (2.2) – algorytmy optymalizacyjne OPA o parametrach przedstawionych w tabeli 3.2.

Tabela 3.5. Wyniki testów T kolejności par Wilcoxona dla najlepszych rozwiązań uzyskanych w ramach scenariusza SOO_{3AP}^{74ST} za pomocą algorytmów OPA (tab. 3.2) i 10 ważnych prób

Algorytm nr 1 vs nr 2	Wartość testu T	Wartość prawdopodobieństwa p
GA/NM vs GA	0,000	0,0051
GA/NM vs BeA	3,000	0,0125
GA/NM vs { pozostałe OPA z tab. 8.3}	≥15,000	≥ 0,2026
CS vs GA	0,000	0,0051
CS vs BeA	1,000	0,0069
CS vs { pozostałe OPA z tab. 8.3}	≥19,000	≥ 0,3863

Tabela 3.6. Współrzędne 3 punktów AP oraz wartości funkcji F_{c_1} zadania optymalizacji dla scenariusza SOO_{3AP}^{74ST}

Metoda optymalizacji	Współrzędne trzech AP [m]		N_{ST}	Wartość funkcji F_{c1} [m]
	$X1_{AP}$	$X2_{AP}$		
GA	15,34	14,66	21	628,92
	17,23	7,58	15	
	16,27	6,67	38	
BeA	7,02	4,55	24	411,81
	21,83	4,48	13	
	8,25	13,47	37	
ABC	4,43	18,62	18	334,16
	3,05	6,00	37	
	14,99	5,23	19	
BA	19,41	4,29	27	301,60
	5,76	17,61	18	
	6,42	5,22	29	
PSO	4,43	18,62	18	296,38
	5,38	6,69	27	
	16,63	4,61	29	
FA	7,11	6,29	32	293,69
	19,41	4,92	24	
	4,43	18,62	18	
SA	19,41	4,92	25	291,76
	5,35	18,26	18	
	6,88	5,77	31	
CS	5,51	18,14	18	290,06
	16,84	4,55	29	
	5,00	6,08	27	
GA/NM	5,90	5,78	26	287,52
	5,12	18,38	18	
	17,27	4,87	30	



Rysunek 3.2. Rzut poziomy wybranych sal i fragmentu korytarza jednego piętra budynku z 74 stacjami ST oraz trzema przykładowo rozlokowanymi punktami AP dla scenariusza SOO_{3AP}^{74ST}

Dla algorytmu pszczelego BeA, ze względu na zbyt długi czas prowadzonych obliczeń, zdecydowano się ograniczyć maksymalną liczbę iteracji N_{iter}^{max} z 1000 do 300 [10].

Analogicznie jak dla scenariuszu SOO_{3AP}^{20ST}, tak i tym razem postawiono hipotezę zerową o braku istotnych różnic między najmniejszymi (najlepszymi) wartościami funkcji kryterialnej F_{C_1}, uzyskanymi dla poszczególnych par porównywanych ze sobą algorytmów z tabeli 3.2.

Analiza wyników testu Wilcoxona (tab. 3.5), dla dwóch par algorytmów: GA/NM vs GA oraz GA/NM vs BeA prowadzi do wniosku, że postawiona hipoteza zerowa powinna zostać odrzucona przy założonym poziomie $p < 0{,}05$.

Na rysunku 3.2 przedstawiono uzyskane za pomocą algorytmu GA/NM najlepsze rozwiązanie zadania optymalizacji jednokryterialnej dla funkcji F_{C_1}, zapisanej wzorem (2.2) i scenariusza SOO_{3AP}^{74ST}.

Dla wszystkich siedmiu algorytmów (poza GA oraz BeA) rozlokowanie punktów AP między poszczególnymi pomieszczeniami (rys. 3.2) było podobne (tab. 3.6). A dokładniej to wszystkie punkty AP znalazły się w tych samych pomieszczeniach, co w najlepszym – uzyskanym za pomocą algorytmu hybrydowego GA/NM – rozwiązaniu (rys. 3.2). Analizując wyniki zaprezentowane można zauważyć w tabeli 3.6, że różnica wartości funkcji kryterialnej F_{C_1} otrzymanych przy użyciu algorytmu GA/NM i CS wynosi $\Delta F_c \approx 2{,}54$ m, czyli mniej niż 1% w odniesieniu do rozwiązania najlepszego, a błąd położenia dla wszystkich trzech punktów AP to $\Delta X \approx 1{,}91$ m.

W tabelach 3.3 i 3.6 przedstawiono wartości odpowiednio funkcji F_{C_4} i C_1 oraz współrzędne trzech punktów AP wyznaczone za pomocą wybranych algorytmów optymalizacyjnych OPA dla scenariuszy SOO_{3AP}^{20ST} i SOO_{3AP}^{74ST}Poza tym, zamieszczono w nich wartości czasów T_{opt}, po upływie których – zgodnie z przyjętymi kryteriami stopu – algorytmy OPA (tab. 3.2) znajdowały wartości optymalne rozwiązywanych zadań.

Przeprowadzone analizy porównawcze, w tym między innymi porównanie współrzędnych punktów AP – otrzymanych za pomocą wybranych OPA, wykazały (tab. 3.3), że funkcja F_{C_1} jest niezbyt czuła na wybór algorytmu optymalizacyjnego. Co nie oznacza, że pozostałe funkcje kryterialne, omówione w podrozdziale 2.4, nie wymagają bardziej starannego doboru i implementacji algorytmów optymalizacyjnych, wspierających automatyczne planowanie sieci WLAN z infrastrukturą [543].

3.2. Porównanie wybranych algorytmów optymalizacji wielokryterialnej

W niniejszym podrozdziale do wyznaczenia – dla scenariusza MOO_{3AP}^{58ST} – odpowiednio maksimum funkcji zasięgowej F_{C_1} (2.2) oraz maksimum funkcji kryterialnej F_{C_8} (2.15) uwzględniającej przepustowość sieci WLAN, wykorzystano:

- algorytm NSGAIII (*Non-dominated Sorting Genetic Algorithm*) [443], [444];
- algorytm optymalizacji rojem cząstek MOPSO (*Multi Objective Particle Swarm Optimisation)* [446], [447];
- algorytm kukułki MOCS (*Multi Objective Cuckoo Search)* [448].

W ramach scenariusza $MOO_{3AP}^{\mathbf{58ST}}$ poszukiwano – za pomocą wybranych algorytmów optymalizacyjnych (tab. 3.2 i 3.7) – najlepszego przydziału pięćdziesięciu ośmiu stacji ST ($N_{ST} = 58$) do trzech punktów AP ($N_{AP} = 3$).

Porównania, wymienionych w tabeli 3.7 algorytmów optymalizacji wielokryterialnej MOO, dokonano przy użyciu wskaźnika zagregowanego (syntetycznego) GQ_i , obliczonego z wykorzystaniem Metody Unitaryzacji Zerowanej (MUZ) [544].

Zmienną agregowaną (syntetyczną) GQ_i (*Global Quality*) obliczono jako:

$$GQ_i = \frac{1}{N_{F_c}} \sum_{j=1}^{N_{F_c}} z_{ij} \quad \left(i = 1,2, \dots, N_p\right), F_{c_j} \in St, \tag{3.1}$$

gdzie: N_{F_c} – liczba zastosowanych zmiennych diagnostycznych (np. funkcji kryterialnych), N_p – liczba wszystkich rozważonych przypadków (N_p^{known} lub N_p^{true}), z – unormowana zmienna diagnostyczna obliczana dla stymulanty (St) ze wzoru:

$$z_{ij} = \frac{F_{c_{ij}} - \min\limits_i F_{c_{ij}}}{\max\limits_i F_{c_{ij}} - \min\limits_i F_{c_{ij}}} \begin{pmatrix} i = 1,2, \dots, N_p \\ j = 1,2, \dots, N_{F_c} \end{pmatrix}, \tag{3.2}$$

$$F_{c_j} \in St,$$

w przypadku destymulanty (*De*) ze wzoru:

$$z_{ij} = \frac{\max\limits_i F_{c_{ij}} - F_{c_{ij}}}{\max\limits_i F_{c_{ij}} - \min\limits_i F_{c_{ij}}} \begin{pmatrix} i = 1,2, \dots, N_p \\ j = 1,2, \dots, N_{F_c} \end{pmatrix}, \tag{3.3}$$

$$F_{c_j} \in De,$$

zaś w przypadku nominanty (No) ze wzoru:

$$z_{ij} = \begin{cases} \dfrac{F_{c_{ij}} - \min\limits_i F_{c_{ij}}}{c_{1no} - \min\limits_i F_{c_{ij}}} \begin{pmatrix} i = 1,2, \dots, N_p \\ j = 1,2, \dots, N_{F_c} \end{pmatrix}, \\ 1, c_1 \le F_{c_j} \le c_2, \\ \dfrac{F_{c_{ij}} - \max\limits_i F_{c_{ij}}}{c_{2no} - \max\limits_i F_{c_{ij}}} \begin{pmatrix} i = 1,2, \dots, N_p \\ j = 1,2, \dots, N_{F_c} \end{pmatrix}, \end{cases} \tag{3.4}$$

$$F_{c_j} \in No.$$

Stymulanta (*St*) to zmienna diagnostyczna, której wzrost wartości jest kojarzony z poprawą, a spadek z pogorszeniem się oceny analizowanego zjawiska. Destymulanta (*De*) to zmienna diagnostyczna, której wzrost jest kojarzony ze spadkiem, a spadek ze wzrostem oceny danego zjawiska. Nominata (*No*) to taka zmienna diagnostyczna, dla której istnieje najkorzystniejsza (z punktu widzenia zjawiska złożonego) wartość lub przedział wartości.

Zaprezentowany we wzorach (3.2)–(3.3) sposób normowania zmiennych diagnostycznych stymulanty *St* oraz destymulanty *De* jest jednym z wielu możliwości [26]. Metody te mogą być oparte na:

- formule przekształcenia ilorazowego z miarami zróżnicowania cech (odchyleniu standardowym zmiennej, rozstępie zmiennej), innymi

stałymi parametrami cech (średniej arytmetycznej, maksymalnej bądź minimalnej wartości zmiennej, długości wektora realizacji zmiennej, sumie realizacji zmiennej),
- metodach rangowych.

Podgrupę obiektów (rozwiązań) najlepszych określono jako [544]:

$$GQ_i^{best} \in [\max GQ_i - \frac{1}{3}(\max GQ_i - min\ GQ_i), max\ GQ_i] \tag{3.5}$$

Zmiennymi diagnostycznymi były:
- *GD*(*Generational Distance*) [545], która wyraża odległość (bliskość) rozwiązań z frontu Pareto ($\mathcal{P}$) do znalezionych rozwiązań:

$$GD \triangleq \frac{1}{N_p^{known}} \sqrt{\sum_{j=1}^{N_p^{known}} dp_j^2}, \tag{3.6}$$

gdzie:

$N_p{}^{known}$ – liczba znanych rozwiązań,

dp_j – odległość rozwiązania (punktu) ze zbioru N^{known} do najbliższego punktu z frontu Pareto ($\mathcal{P}$),

- *RGD* (*Reverse Generational Distance*) [545], [546], [547], która wyraża odległość (bliskość) frontu Pareto ($\mathcal{P}$):

$$RGD \triangleq \frac{1}{N_p^{true}} \sqrt{\sum_{j=1}^{N_p^{true}} dp_j^2}, \tag{3.7}$$

gdzie:

$N_p{}^{true}$ – liczba punktów frontu Pareto ($\mathcal{P}$),

dp_j – odległość rozwiązania (punktu) z frontu Pareto ($\mathcal{P}$) do najbliższego rozwiązania (ze zbioru N^{known}),

miara rozrzutu dla znalezionych rozwiązań *MS* (*Most Spread*) obliczona jako:

$$MS = \sqrt{\sum_{j=1}^{K} \left(\frac{\max\limits_{i=1\ldots N_p^{known}} F_{c_{ij}} - \min\limits_{i=1\ldots N_p^{known}} F_{c_{ij}}}{\max\limits_{k=1\ldots N_p^{true}} F_{c_{kj}} - \min\limits_{i=1\ldots N_p^{true}} F_{c_{kj}}} \right)^2}, \tag{3.8}$$

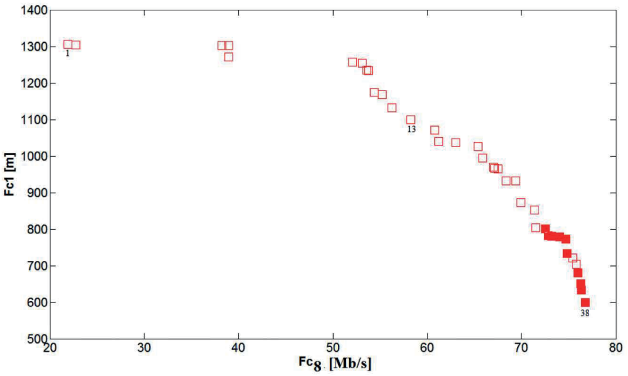
- liczba znanych rozwiązań leżących na froncie Pareto $N_p{}^{\boldsymbol{kntr}}$.

Ponieważ w tego typu zadaniach nie jest znany front Pareto, dlatego utworzono go, biorąc pod uwagę wszystkie wyniki uzyskane dla trzech algorytmów i dla wszystkich zestawów parametrów. Otrzymano front Pareto ($\mathcal{P}$), składający się z 38 punktów (rys. 3.3, tab. 3.8).

Tabela 3.7. Dane opisujące zadanie optymalizacji dla scenariusza MOO^{58ST}_{3AP}

Kryterium stopu OPA– maksymalna liczba iteracji N_{iter}^{max}		$N_{iter}^{max} = 100$
NSGAIII	Rozmiar populacji N_{pop}	N_{pop}=50
	Prawdopodobieństwo mutacji m_u	$m_u = \{0{,}01;\ 0{,}02; 0{,}04; 0{,}05;\ 0{,}06; 0{,}08;\ 0{,}10\}$
	Oddziaływanie krzyżowania na populację (*crossover fraction*) c_p	$c_p = \{0{,}50; 0{,}60; 0{,}70; 0{,}80; 0{,}90\}$
	Rozmiar frakcji zmutowanych osobników m_p	$m_p = \{0{,}50; 0{,}60; 0{,}70\}$
MOPSO	Liczebność „roju" N_{pso}	N_{pso}=50
	Współczynnik poznawczy N_{pso}	$c_1 = \{1{,}00;\ 1{,}50;\ 1{,}80;\ 2{,}00;\ 2{,}50; 3{,}00\}$
	Współczynnik społeczny c_2	$c_2 = \{3{,}00;\ 2{,}50;\ 2{,}20;\ 2{,}00;\ 1{,}50; 1{,}00\}$
MOCS	Liczba gniazd N_{nest}	N_{nest}=50
	Prawdopodobieństwo wykrycia jaja podrzuconego przez kukułkę p_a	$p_a = \{0{,}25; 0{,}30, ; 0{,}35; 0{,}40; 0{,}45; 0{,}47;\ 0{,}50\}$

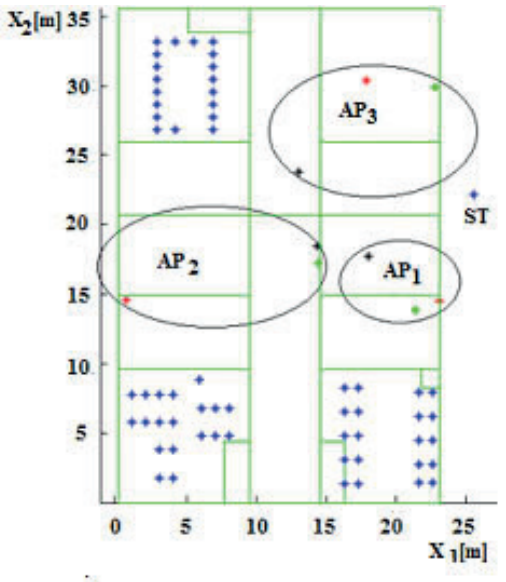
W tabeli 3.8, dla danego zestawu obliczeniowego, zaprezentowano dodatkowo czasy trwania obliczeń numerycznych T_{opt} na komputerze PLgrid/Hydra.



Rysunek 3.3. Front Pareto ($\mathcal{P}$) *z zaznaczonymi* ($\sum N_p^{kntr} = N_p^{true} = 38$) *rozwiązaniami* (wypełnione na czerwono to rozwiązania z zastosowaniem MOCS)

Tabela 3.8. Wyniki obliczeń numerycznych dla scenariusza $MOO^{\mathbf{58ST}}_{3AP}$

Algorytm	Parametry	GQ_i	GD	RGD	MS	N_p^{kntr}	Czas trwania obliczeń na komputerze PLgrid/Hydra T_{opt} [s]
NSGAIII	$m_u = 0{,}01$ $c_p = 0{,}5$ $m_p = 0{,}5$	0,46	2,54	15,79	1,12	0	19872
	$m_u = 0{,}01$ $c_p = 0{,}8$ $m_p = 0{,}5$	0,31	2,19	27,09	0,79	0	78963
	$m_u = 0{,}01$ $c_p = 0{,}9 m_p = 0{,}5$	0,47	2,41	14,06	1,09	0	87974
	$m_u = 0{,}01$ $c_p = 0{,}6$ $m_p = 0{,}6$	0,50	2,96	12,75	1,29	0	74223
	$m_u = 0{,}02$ $c_p = 0{,}5$ $m_p = 0{,}5$	0,42	3,23	16,67	0,99	0	64292
	$m_u = 0{,}02$ $c_p = 0{,}6$ $m_p = 0{,}6$	0,46	2,78	15,33	1,17	0	78321
	$m_u = 0{,}02$ $c_p = 0{,}7$ $m_p = 0{,}7$	0,54	2,87	12,92	1,54	0	85819
	$m_u = 0{,}04$ $c_p = 0{,}5$ $m_p = 0{,}5$	0,44	2,28	16,73	1,05	0	61508
	$m_u = 0{,}05$ $c_p = 0{,}5$ $m_p = 0{,}5$	0,48	2,54	13,70	1,18	0	19917
	$m_u = 0{,}05$ $c_p = 0{,}6$ $m_p = 0{,}5$	0,43	2,77	17,63	1,10	0	74161
	$m_u = 0{,}06$ $c_p = 0{,}5$ $m_p = 0{,}5$	0,38	2,47	20,95	0,93	0	65323
	$m_u = 0{,}08$ $c_p = 0{,}5$ $m_p = 0{,}5$	0,50	2,66	12,71	1,25	0	63285
	$m_u = 0{,}1$ $c_p = 0{,}5$ $m_p = 0{,}5$	0,43	2,47	17,17	1,05	0	65597
MOPSO	$\boldsymbol{c_1 = 1{,}0}$ $\boldsymbol{c_2 = 1{,}0}$	**0,74**	**1,23**	**8,88**	**0,85**	**11**	34419
	$c_1 = 1{,}0$ $c_2 = 3{,}0$	0,44	10,14	10,75	1,43	0	27571
	$\boldsymbol{c_1 = 1{,}5}$ $\boldsymbol{c_2 = 1{,}5}$	**0,66**	**18,92**	**2,40**	**2,12**	**7**	32812
	$\boldsymbol{c_1 = 1{,}5}$ $\boldsymbol{c_2 = 2{,}5}$	**0,63**	**8,64**	**3,83**	**1,38**	**5**	65028
	$c_1 = 1{,}8$ $c_2 = 2{,}2$	0,50	5,63	4,90	1,13	0	67323
	$c_1 = 2{,}0$ $c_2 = 2{,}0$	0,49	12,53	4,70	1,62	0	31540
	$c_1 = 2{,}5$ $c_2 = 1{,}5$	0,43	12,05	8,37	1,42	0	27380
	$\boldsymbol{c_1 = 3{,}0}$ $\boldsymbol{c_2 = 1{,}0}$	**0,72**	**3,38**	**3,27**	**1,46**	**5**	77200
MOCS	p_a =0,25	0,49	5,82	12,53	1,45	0	67201
	p_a =0,30	0,21	5,70	32,46	0,60	1	65952
	$\boldsymbol{p_a}$ **=0,35**	**0,57**	**5,95**	**10,91**	**1,46**	**3**	62361
	p_a =0,40	0,51	5,91	16,92	1,40	3	61300
	p_a =0,45	0,23	6,44	26,40	0,60	0	58758
	p_a =0,47	0,41	8,00	13,32	0,93	2	57620
	p_a =0,50	0,43	4,70	18,21	1,14	1	56802



Rysunek 3.4. Lokalizacja trzech punktów AP w sieci WLAN z 58 stacjami ST dla trzech najlepszych (GQ_i^{best}) znanych rozwiązań

W tabeli 3.8 przedstawiono, obliczone za pomocą MUZ [544] dla różnych wartości parametrów sterujących zastosowanych algorytmów, wartości wskaźnika GQ_i.W obliczeniach przyjęto, że GD i RGD są destymulantami, zaś MS i N_p^{kntr} stymulantami (tab. 3.8).

Z analizy rezultatów zamieszczonych w tabeli 3.8 wynika, że zastosowanie algorytmu NSGAIII, w żadnym przypadku nie prowadziło do uzyskania punktu leżącego na froncie Pareto ($N_p^{kntr} = 0$). Mimo małych wartości współczynnika GD ($N_p^{known} = 50$), żadne z rozwiązań nie należało do grupy „obiektów" najlepszych.

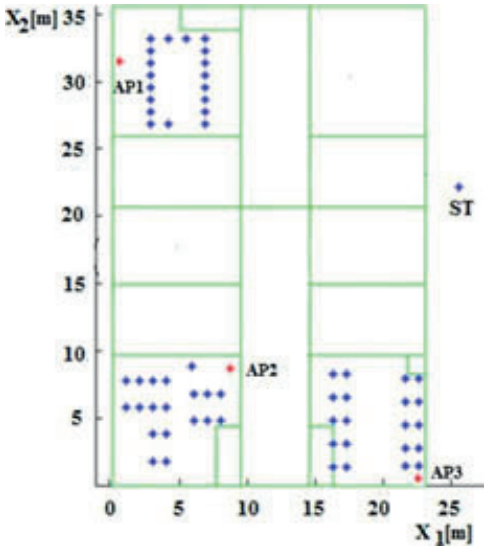
Spośród 38 rozwiązań, które znalazły się na froncie Pareto (P) aż 28 uzyskano dla MOPSO (rys. 3.3, tab. 3.8). Dla tego algorytmu uzyskano także największe wartości zmiennej zagregowanej GQ_i, a cztery zestawy parametrów znalazły się w grupie najlepszych przypadków (tab. 3.8).

Zastosowanie algorytmu MOCS doprowadziło do 10 rozwiązań (tab. 3.8) z dużą wartością funkcji przepustowości sieci F_{c8}, leżących na froncie Pareto (P) (rys. 3.3). Dla jednego zestawu parametrów (p_a =0,35) otrzymano wartość zmiennej syntetycznej (obliczonej na $GQ_i > 0{,}56$) w grupie rozwiązań najlepszych ($GQ_i^{\boldsymbol{best}}$) – wyróżnionych w tabeli 3.8 pogrubioną czcionką.

Analizując przedstawione w tabeli 3.8 wyniki obliczeń należy zauważyć, że dla algorytmu MOPSO uzyskano najmniejsze wartości *GD* i *RGD*. Zastosowanie tego algorytmu doprowadziło także do największych wartości współczynnika *MS* oraz największej liczby punktów na froncie Pareto (N_p^{kntr}).

Dodatkowo przeanalizowano wszystkie rozwiązania z N_p^{true} oraz $N_{p\ ST}^{tknown}$ i wybrano te, w których liczba przydzielonych do punktu AP (N^{APn}) stacji ST nie przekraczała 20 (jest nominantą). Spośród dziewięciu rozwiązań trzy – o największym GQ_i^{best}– przedstawiono na rysunku 3.4.

Ze względu na fakt, że usytuowanie stacji ST było nierównomierne (odpowiadało położeniu komputerów stacjonarnych PC w laboratoriach) rozmieszczenie punktów AP wystąpiło w pomieszczeniach bez stacji ST (rys. 3.5).



Rysunek 3.5. Przykładowa lokalizacja trzech punktów AP w sieci WLAN z 58 stacjami ST dla najlepszego (GQ_i^{best}) rozwiązania z jednorodnymi przeszkodami i tłumieniem $L_w = 10$ dB

Ponieważ uwarunkowania lokalne mają wpływ na rozmieszczenie punktów AP, dodatkowo sprawdzono działanie algorytmu w przypadku ujednolicenia rodzaju przeszkód i zwiększenia ich tłumienia L_w do 10 dB, w analizowanym środowisku radiokomunikacyjnym. Zastosowanie rozważanych algorytmów optymalizacyjnych – z najlepszymi parametrami określonymi dla scenariusza MOO_{3AP}^{58ST}, doprowadziło do umiejscowienia trzech punktów AP w pomieszczeniach, w których znajdowały się stacje ST (rys. 3.5) [548].

3.3. Podsumowanie

Należy zauważyć, że zarówno dla scenariusza $SOO_{3AP}^{\mathbf{20ST}}$ i optymalizacji jednokryterialnej, jak i dla scenariusza $MOO_{3AP}^{\mathbf{58ST}}$ i optymalizacji dwukryterialnej, najlepsze rozwiązania uzyskano przy zastosowaniu algorytmu kukułki (CS). Ponadto dla scenariusza $SOO_{3AP}^{\mathbf{74ST}}$ różnica ΔF_c między najmniejszymi (najlepszymi) wartościami funkcji kryterialnej F_{c1}, otrzymanymi przy użyciu algorytmu genetycznego GA/NM i algorytmu kukułki CS, nie przekraczała 1%.

Zastosowanie optymalizacji wielokryterialnej MOO daje możliwość uwzględnienia wielu kryteriów optymalizacji oraz otrzymania zbioru rozwiązań niezdominowanych. Trudnością w zastosowaniu MOO jest brak znajomości frontu Pareto (P) oraz duża liczba rozwiązań Paretooptymalnych, które trzeba przeanalizować, wybierając jedno z nich. Analizując otrzymane wyniki można zauważyć, że zastosowane algorytmy MOPSO oraz MOCS dla $MOO_{3AP}^{\mathbf{58ST}}$ uzupełniły się. Algorytm MOPSO generowało rozwiązania z większą wartością funkcji celu F_{c1}, a MOCS z większymi wartościami F_{c_8} [549].

Dlatego też, opierając się na wynikach prac [448], [550] oraz na rezultatach analizowanych w tym rozdziale zadaniach optymalizacyjnych, zdecydowano się wybrać algorytm kukułki CS, zarówno w wersji jednokryterialnej SOO, jak i wielokryterialnej MOO (MOCS łącznie z MOPSO) do wykorzystania go w pierwszej kolejności w scenariuszach badawczych zaprezentowanych w rozdziale czwartym niniejszej monografii, a dotyczących wybranych przykładów planowania sieci WLAN standardu IEEE 802.11 z infrastrukturą.

4. PRZYKŁADY ZADAŃ OPTYMALIZACJI W PLANOWANIU SIECI WLAN

W niniejszym rozdziale zaprezentowano przykładowe zadania optymalizacji jednokryterialnej SOO (podrozdz. 4.1) i wielokryterialnej MOO (podrozdz. 4.2), dotyczące problematyki planowania sieci WLAN z infrastrukturą standardu IEEE 802.11. Rozwiązania tych zadań znaleziono z wykorzystaniem modelu propagacyjnego MWM (podrozdz. 2.9.5, 2.10) i modelu matematycznego DCF_2 (6; 15; 7) przepustowości sieci (podrozdz. 2.12.5) za pomocą jedno i wielokryterialnego algorytmu kukułki MOCS (rozdz. 1), a także rojem cząstek (MOPSO).

4.1. Przykłady zadań optymalizacji jednokryterialnej SOO w planowaniu sieci WLAN

Dla pomieszczeń z rysunku 4.1, reprezentujących typowe wewnątrzbudynkowe środowisko radiokomunikacyjne, poszukiwano dla sieci WLAN z infrastrukturą standardu IEEE 802.11b/g (tab. 4.1) optymalnego rozmieszczenia N_{AP} punktów dostępu AP, pracujących w trzech kanałach radiowych $ch_{nr} = 1, ch_{nr} = 6, ch_{nr} = 11$ (rys. 2.4).

W tym celu sformułowano i następnie rozwiązano zadanie optymalizacji jednokryterialnej SOO, w którym wykorzystano, zapisaną wzorem (2.10), funkcję kryterialną F_{c_5}:

$$F_{c_5} = \Psi \cdot \left(\frac{\sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} M_{TR_{max_{ij}}}}{N_{ST} \cdot M_{TR_{max}}} \right) + (1 - \Psi) \cdot \left(\frac{\sum_{j=1}^{N_{AP}} N_{ST_j}}{N_{ST}} \right), \tag{4.1}$$

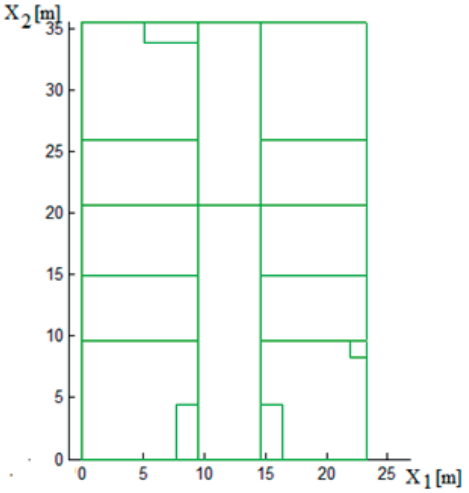
dla dwóch skrajnych przypadków:

$$F_{c_5}(\Psi = 0) = F_{c_5}^0 = \frac{\sum_{j=1}^{N_{AP}} N_{ST_j}}{N_{ST}} \text{ i } F_{c_5}(\Psi = 1) = F_{c_5}^1 = \frac{\sum_{j=1}^{N_{AP}} \sum_{i=1}^{N_{ST_j}} M_{TR_{max_{ij}}}}{N_{ST} \cdot M_{TR_{max}}} \tag{4.2}$$

Tabela 4.1. Dane techniczne punktów dostępu AP i stacji abonenckich ST w sieci WLAN zaprojektowanej dla pomieszczeń i korytarzy z rysunku 4.1

Sieć WLAN standardu IEEE 802.11b/g	
Czułości P_S^{thr} odbiornika stacji ST, gwarantujące na jego wyjściu bitową stopę błędów $BER \leq 10^{-5}$ i odpowiadające im szybkości transmisji M_{TR}	P_S^{thr} $= \{-90;\ -89;\ -88;-87;\ -86;\ -85;$ $-84;\ -82;\ -79;\ -75;\ -72;\ -68\}$ dBm
	$M_{TR} = \{1{,}0;\ 2{,}0;\ 5{,}5;\ 6{,}0;\ 9{,}0;\ 11{,}0;$ $12{,}0;\ 18{,}0;\ 24{,}0;\ 36{,}0;\ 48{,}0;\ 54{,}0\}$ Mb/s
Moc zastępcza promieniowana izotropowo $EIRP$ puntu dostępu AP	$EIRP$ = 20 dBm
Rozmiar ładunku danych $L_{DATA\_payload}$	$L_{DATA\_payload}$ =1472 B

Dla $\Psi = 0$ funkcja kryterialna F_{c_5} definiuje zadanie optymalizacji ukierunkowane wyłącznie na zasięg działania sieci, natomiast dla $\Psi = 1$ na szybkości transmisji osiągane w poszczególnych lokalizacjach stacji abonenckich ST.



Rysunek 4.1. Rzut poziomy wybranych pomieszczeń i korytarzy jednego piętra budynku

Miejsca – punkty testowe TP (*Test Point*), w których wyznaczano za pomocą modelu MWM poziomy mocy sygnału *RSSI*, znajdowały się w węzłach siatki o wymiarach 1 m×1 m, co dla rozważanego obszaru z rysunku 4.1 dało 864 lokalizacje.

W tabeli 4.2 zaprezentowano otrzymane za pomocą algorytmu kukułki CS, rozwiązania zadania optymalizacji, które gwarantowały w miejscu lo-

kalizacji punktu TP dla minimalnego poziomu mocy *RSSI* równego $P_S^{thr}(1) = -90$ dBm szybkość transmisji nie gorszą niż 1 Mb/s, a dla $P_S^{thr}(6) = -87$ dBm, co najmniej 6 Mb/s [483] (tab. 4.1).

Tabela 4.2. Wartości funkcji kryterialnej F_{C5}^0 oraz liczba i współrzędne punktów AP

N_{AP}	$P_S^{thr}(1) = -90$ dBm			$P_S^{thr}(6) = -87$ dBm		
	Współrzędne punktu AP [m]		F_{C5}^0	Współrzędne punktu AP [m]		F_{C5}^0
	$X_{1_{AP}}$	$X_{2_{AP}}$		$X_{1_{AP}}$	$X_{2_{AP}}$	
$AP_1(ch_{nr} = 1)$	13,89	15,30	0,9979	13,96	17,92	0,9931
$AP_1\ (ch_{nr} = 1)$ $AP_2\ (ch_{nr} = 6)$	15,51 7,39	8,24 21,80	1,0000	10,48 5,48	11,70 35,60	1,0000

Tabela 4.3. Wartości funkcji kryterialnej F_{C5}^0 i współrzędnych punktów AP

N_{AP}	$P_S^{thr}(36) = -75$ dBm			$P_S^{thr}(54) = -68$ dBm		
	Współrzędne punktu AP [m]		F_{C5}^0	Współrzędne punktu AP [m]		F_{C5}^0
	$X_{1_{AP}}$	$X_{2_{AP}}$		$X_{1_{AP}}$	$X_{2_{AP}}$	
$AP_1(ch_{nr} = 1)$	11,07	13,47	0,6736	11,90	9,66	0,4190
$AP_1\ (ch_{nr} = 1)$ $AP_2\ (ch_{nr} = 6)$	9,82 13,92	25,88 10,21	0,9757	11,86 11,74	9,67 26,11	0,7870
$AP_1\ (ch_{nr} = 1)$ $AP_2\ (ch_{nr} = 6)$ $AP_3\ (ch_{nr} = 11)$	13,20 0,62 16,46	26,84 14,30 9,50	0,9971	5,37 14,58 11,60	9,91 10,77 26,03	0,9225
$AP_1\ (ch_{nr} = 1)$ $AP_2\ (ch_{nr} = 6)$ $AP_3\ (ch_{nr} = 11)$ $AP_4\ (ch_{nr} = 1)$	16,96 8,66 15,40 4,09	0,97 12,59 23,43 35,32	**1,0000**	19,31 11,00 6,01 0,01	26,62 15,02 31,45 2,29	0,8889
$AP_1\ (ch_{nr} = 1)$ $AP_2\ (ch_{nr} = 6)$ $AP_3\ (ch_{nr} = 11)$ $AP_4\ (ch_{nr} = 1)$ $AP_5\ (ch_{nr} = 6)$	18,11 17,27 12,33 0,01 3,20	35,60 0,01 15,52 6,73 35,60	1,0000	0,01 16,32 11,45 17,02 8,12	2,90 3,56 15,64 15,32 27,75	0,9444
$AP_1\ (ch_{nr} = 1)$ $AP_2\ (ch_{nr} = 6)$ $AP_3\ (ch_{nr} = 11)$ $AP_4\ (ch_{nr} = 1)$ $AP_5\ (ch_{nr} = 6)$ $AP_6\ (ch_{nr} = 11)$	17,55 23,17 0,01 0,01 5,16 2,59	21,98 7,35 26,96 0,01 20,69 14,37	1,0000	19,18 15,85 23,17 1,19 0,01 11,41	8,92 29,46 32,93 26,76 0,01 15,30	0,8113

Z tabeli 4.2 wynika, że osiągnięcie we wszystkich 864 punktach testowych TP poziomów, czy to $P_S^{thr}(1)$ =−90 dBm, czy $P_S^{thr}(6)$ =−87 dBm i tym samym zapewnienie wszystkim położonym w 864 punktach stacjom ST odpowiednio szybkości 1 i 6 Mb/s wymagało uruchomienia co najmniej dwóch punktów AP. Bowiem, dopiero dla dwóch punktów AP w analizowanej sieci WLAN (rys. 4.1) funkcja kryterialna Fc_5^0 przyjmowała wartość równą 1. Podczas gdy dla $N_{AP} = 1$, czyli dla jednego punktu AP, wartość funkcji kryterialnej Fc_5^0 była mniejsza od 1.

Natomiast w tabeli 4.3 zaprezentowano wartości funkcji Fc_5^0 oraz współrzędne punktów AP uzyskane dla poziomów *RSSI* równych $P_S^{thr}(36)$= −75 dBm oraz $P_S^{thr}(54)$= −68 dBm, którymo dpowiadały szybkości transmisji: 36 i 54 Mb/s (tab. 4.1).

Tabela 4.4. Wartości funkcji kryterialnej Fc_5^1 i współrzędnych punktów AP

N_{AP}	Współrzędne punktu AP [m]		Wartość F_{C5}^1
	$X_{1_{AP}}$	$X_{2_{AP}}$	
$AP_1(ch_{nr} = 1)$	11,26	15,04	0,6976
AP_1 $(ch_{nr} = 1)$	11,88	9,72	0,9457
AP_2 $(ch_{nr} = 6)$	11,81	25,91	
AP_1 $(ch_{nr} = 1)$	6,29	9,86	0,9811
AP_2 $(ch_{nr} = 6)$	11,73	26,41	
AP_3 $(ch_{nr} = 11)$	14,85	11,05	
AP_1 $(ch_{nr} = 1)$	22,05	8,57	0,9645
AP_2 $(ch_{nr} = 6)$	18,51	25,97	
AP_3 $(ch_{nr} = 11)$	7,45	13,11	
AP_4 $(ch_{nr} = 1)$	0,91	33,49	
AP_1 $(ch_{nr} = 1)$	0,98	8,83	0,9862
AP_2 $(ch_{nr} = 6)$	7,09	28,38	
AP_3 $(ch_{nr} = 11)$	11,75	18,25	
AP_4 $(ch_{nr} = 1)$	15,86	28,15	
AP_5 $(ch_{nr} = 6)$	23,15	3,95	
AP_1 $(ch_{nr} = 1)$	11,95	22,15	0,9098
AP_2 $(ch_{nr} = 6)$	0,01	34,64	
AP_3 $(ch_{nr} = 11)$	0,01	16,03	
AP_4 $(ch_{nr} = 1)$	23,17	0,01	
AP_5 $(ch_{nr} = 6)$	11,61	9,18	
AP_6 $(ch_{nr} = 11)$	23,17	35,60	

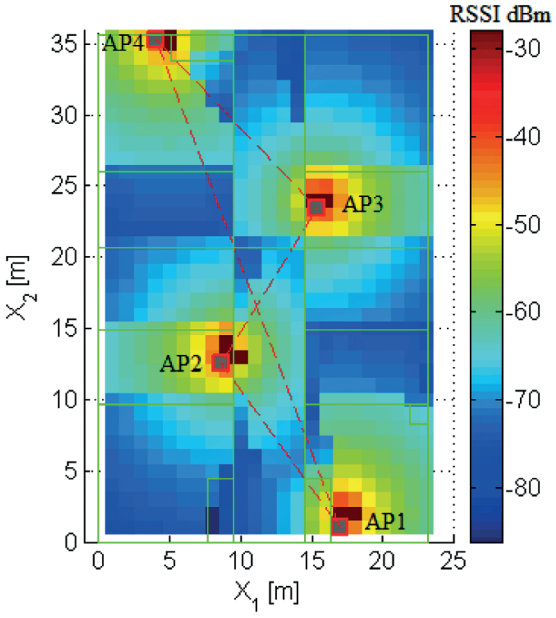
Dla przypadku $P_S^{thr}(36) = -75$ dBm, funkcja Fc_5^0 osiągnęła wartość 1 już dla 4 punktów dostępu AP (tab. 4.3), co oznaczało, że w każdym z 864 punktów TP możliwa byłaby transmisja z szybkością nie mniejszą niż 36 Mb/s. Natomiast dla czułości odbiornika stacji ST równej $P_S^{thr}(54) = -68$ dBm szybkość transmisji wynoszącą 54 Mb/s udało się osiągnąć,

w najlepszym przypadku przy zastosowaniu 5 punktów AP, dla niespełna 94,5% punktów TP, czyli w około 816 na 864 punkty TP.

Podobna sytuacja wystąpiła dla funkcji F^1, ukierunkowanej na uzyskanie maksymalnych szybkości transmisji, co ilustrują wyniki zamieszczone w tabeli 4.3, gdzie w najlepszym przypadku maksymalną przepustowość sieci udało się uzyskać przy 5 punktach dostępu AP w 852 punktach (98,6%) na 864 punkty testowe TP.

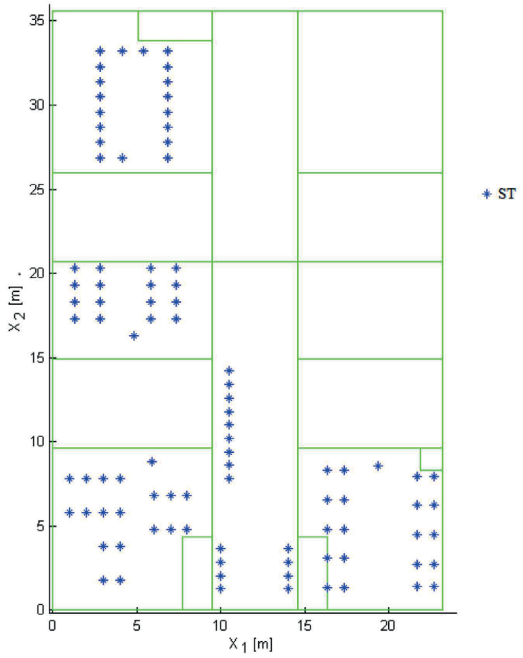
Na rysunku 4.2 przedstawiono wzajemne rozmieszczenie czterech punktów AP dla przypadku $P_S^{thr}(36) = -75$ dBm. Dwa z tych punktów AP_1 i AP_4 pracowały w tym samym kanale radiowym, zaś pozostałe dwa AP_2 i AP_3 używały wzajemnie niezakłócających się kanałów radiowych (tab. 4.3). Zastosowanie optymalizacji algorytmem kukułki CS doprowadziło do rozwiązania, w którym działające w tym samym kanale radiowym punkty AP_1 i AP_4 (rys. 4.2) zostały rozsunięte na maksymalnie dużą odległość.

Podobna sytuacja miała miejsce w przypadkach z większą liczbą punktów AP, w których punkty współkanałowe AP były rozmieszczane względem siebie przez algorytm kukułki CS w jak największych odległościach.



Rysunek 4.2. Rozmieszczenie 4 punktów AP w sieci dla pomieszczeń z rysunku 4.1, opartej na kryterium zasięgu dla Fc_5^0 i $P_S^{thr}(36) = -75$ dBm

W uzupełnieniu rozważań dotyczących zadań optymalizacji jednokryterialnej SOO w planowaniu sieci WLAN w pomieszczeniach z rysunku 4.1, w tabelach: 4.5, 4.7, 4.9, 4.11 przedstawiono wartości funkcji kryterialnych: Fc_1, Fc_2, Fc_3, Fc_4, Fc_7, Fc_8/N_{AP} (rozdz. 2), wyznaczone dla 20 stacji ST znajdujących się w pomieszczeniu z rysunku 3.1, 74 stacji ST w pomieszczeniach z rysunku 3.2 oraz 93 stacji ST rozmieszczonych w pięciu pomieszczeniach z rysunku 4.3. W przypadku wszystkich funkcji kryterialnych, za wyjątkiem funkcji Fc_1 i Fc_2, obliczono wartości maksymalne.



Rysunek 4.3. Rzut poziomy wybranych pomieszczeń i korytarzy jednego piętra budynku z 93 stacjami ST

Tabela 4.5. Wartości wybranych funkcji kryterialnych (rozdz. 2) uzyskane w zadaniach optymalizacji jednokryterialnej SOO dla N_{ST}=20 stacji ST (rys. 3.1)

Nr	Funkcja kryterialna	N_{AP}		
		1	2	3
1.	F_{c_1}[m]	70,91	59,99	**53,98**
2.	$\lvert F_{c_2} \rvert$ [dB]	**46,20**	46,31	46,38
3.	F_{c_3}[(Mb/s)·m^2]	359,05	7510,19	**8282,07**
4.	F_{c_4}[(Mb/s)·m^2]	166,00	3333,57	**3628,54**
5.	F_{c_7} [Mb/s]	**1080,00**		
6.	F_{c_8} /N_{AP} [Mb/s]	24,97	26,82	**27,55**

Dla przypadku N_{ST} =20 stacji ST działających w pomieszczeniu z rysunku 3.1, liczbę analizowanych punktów AP ograniczono od jednego do trzech. Można zauważyć, że liczba punktów AP, uzyskana w wyniku rozwiązania zadania optymalizacji dla poszczególnych funkcji kryterialnych wynosiła jeden lub trzy punkty AP (tab. 4.5). Użycie wszystkich trzech wzajemnie niezakłócających się kanałów radiowych doprowadziło do osiągnięcia przez funkcje Fc_1, Fc_3, Fc_4 i Fc_8/N_{AP} maksymalnych wartości (tab. 4.5).

Dla funkcji kryterialnych: Fc_1 oraz Fc_8/N_{AP} otrzymano ich optymalne (odpowiednio najmniejsze i największe) wartości dla trzech punktów AP, obsługujących podobną liczbę stacji ST (tab. 4.6). Dla trzech punktów AP osiągnięto także wartość maksymalną funkcji: Fc_3 i Fc_4, z tym, że w tym przypadku dała się zauważyć znaczna dysproporcja liczby stacji ST przydzielanych do obsługi poszczególnym punktom AP (tab. 4.6).

Tabela 4.6. Liczba stacji przydzielonych ST punktom AP dla wybranych funkcji kryterialnych (rozdz. 2) w zadaniach optymalizacji jednokryterialnej SOO dla N_{ST} =20 stacji ST (rys. 3.1)

Funkcja kryterialna	Liczba N_{AP} punktów dostępu AP	Liczba stacji ST przydzielonych poszczególnym punktom AP $N_{ST}^{AP_n}$	$\sigma_{\overline{N_{ST}}}$
F_{c_1}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$	8 5 7	1,25
F_{c_2}	$AP_1(ch_{nr} = 1)$	20	0,00
F_{c_3}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$	1 1 18	8,01
F_{c_4}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$	1 1 18	8,01
F_{c_7}	$AP_1(ch_{nr} = 1)$	20	0,00
F_{c_8}/N_{AP}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$	7 7 6	0,47

Dla 74 stacji ST, znajdujących się w trzech pomieszczeniach przedstawionych na rysunku 3.2, przeprowadzona z użyciem funkcji Fc_1 optymalizacja doprowadziła do znalezienia rozwiązania z trzema punktami AP (tab. 4.7). Podczas, gdy w przypadku funkcji kryterialnej Fc_8/N_{AP}, algorytm kukułki CS przydzielił poszczególnym punktom AP mocno zróżnicowaną liczbę stacji ST (tab. 4.8), w tym, na przykład, punktowi AP_3 aż 58, zaś pozostałym punktom AP tylko po 4 stacje ST.

Duże zróżnicowanie co do liczby stacji ST, przydzielonych poszczególnym punktom AP, miało miejsce dla funkcji kryterialnej Fc_4. A to dlatego, że funkcja ta realizuje koncepcję podziału zasięgu radiowego punktu dostępu AP na obszary, w których szybkości transmisji zależą od wartości $SINR$. I tak punktowi AP_3 zostało przydzielonych 72, podczas gdy pozostałym dwóm punktom AP zaledwie (tab. 4.8) po jednej stacji ST.

Tabela 4.7. Wartości wybranych funkcji kryterialnych (rozdz. 2) uzyskane w zadaniach optymalizacji SOO dla N_{ST} =74 stacje ST (rys. 3.2)

Nr	Funkcja kryterialna	N_{AP}					
		1	2	3	4	5	6
1.	F_{c_1}[m]	622,43	433,59	**299,44**	346,52	331,72	321,01
2.	$\lvert F_{c_2} \rvert$ [dB]	56,86	56,92	56,98	**56,85**	56,92	56,99
3.	F_{c_3}[(Mb/s)·m^2]	30301,51	104148,15	131932,40	137955,02	**160765,57**	138650,11
4.	F_{c_4}[(Mb/s)·m^2]	66,67	27138,79	**45513,89**	42870,25	31138,79	
5.	F_{c_7} [Mb/s]	**3990,00**	**3996,00**				
6.	F_{c_8}/N_{AP} [Mb/s]	20,65	24,41	25,67	26,31	**26,69**	26,43

W przypadku zadania optymalizacji o 74 stacjach ST, użycie funkcji kryterialnych F_{c2} doprowadziło do rozwiązania, dla którego optymalna liczba punktów AP była taka sama ($N_{AP} = 4$), jak liczba pomieszczeń ze stacjami ST (tab. 4.7). Z kolei funkcja kryterialna F_{c_8}/N_{AP} osiągnęła wartość maksymalną, równą 26,69, dla $N_{AP} = 5$ punktów AP. A ponieważ była ona o niecały 1% większa od wartość 26,43, uzyskanej dla funkcji Fc_8/N_{AP} o $N_{AP} = 6$ punktach AP (tab. 4.7), dlatego i to rozwiązanie – z powodu przyjętego na 1% poziomu (błędu) zbieżności optymalizacji algorytmem kukułki CS – należy poddać weryfikacji (rozdział 4.3).

Tabela 4.8. Liczba stacji ST przydzielonych punktom AP dla wybranych funkcji kryterialnych (rozdz. 2) w zadaniach optymalizacji jednokryterialnej dla N_{ST} =74 stacji ST (rys. 3.2)

Funkcja kryterialna	Liczba N_{AP} punktów dostępu AP	Liczba stacji STprzydzielonych poszczególnym punktom AP $N_{ST}^{AP_n}$	$\sigma_{\overline{N_{ST}}}$
F_{c_1}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$	19 18 37	8,73
F_{c_2}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$ AP_4 $(ch_{nr} = 1)$	2 34 17 21	11,41
F_{c_3}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$ AP_4 $(ch_{nr} = 1)$ AP_5 $(ch_{nr} = 6)$	10 2 20 4 38	13,18
F_{c_4}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$	1 1 72	33,47
F_{c_7}	$AP_1(ch_{nr} = 1)$	74	0,00
F_{c_8}/N_{AP}	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$ AP_4 $(ch_{nr} = 1)$ AP_5 $(ch_{nr} = 6)$	4 4 58 4 4	21,60
	AP_1 $(ch_{nr} = 1)$ AP_2 $(ch_{nr} = 6)$ AP_3 $(ch_{nr} = 11)$ AP_4 $(ch_{nr} = 1)$ AP_5 $(ch_{nr} = 6)$ AP_6 $(ch_{nr} = 11)$	19 5 17 11 19 3	6,50

Tabela 4.9. Wartości wybranych funkcji kryterialnych (rozdz. 2) uzyskane w zadaniach optymalizacji jednokryterialnej SOO dla N_{ST} =93 stacji ST (rys. 4.3)

Nr	Funkcja kryterialna	N_{AP}					
		1	2	3	4	5	6
1.	F_{c_1}[m]	1025,76	617,02	480,33	391,37	386,04	**370,54**
2.	$\lvert F_{c_2} \rvert$ [dB]	60,96	61,02	61,08	**60,95**	61,02	61,08
3.	F_{c_3}[(Mb/s)·m^2]	44982,37	88156,85	135039,82	**152102,64**	81801,12	146628,84
4.	F_{c_4}[(Mb/s)·m^2]	723,37	10804,69	27321,08	**36497,08**	35989,10	31598,24
5.	F_{c_7} [Mb/s]	4607,50	**5022,00**				
6.	F_{c_8}/N_{AP} [Mb/s]	14,15	22,88	24,83	25,62	**26,16**	26,00

Tabela 4.10. Liczba stacji ST przydzielonych punktom AP dla wybranych funkcji kryterialnych (rozdz. 2) w zadaniach optymalizacji jednokryterialnej dla N_{ST} =93 stacji ST (rys. 4.3)

Funkcja kryterialna	Liczba N_{AP} punktów dostępu AP	Liczba stacji ST przydzielonych poszczególnym punktom AP $N_{ST}^{AP_n}$	$\sigma_{\overline{N_{ST}}}$
F_{c_1}	AP_1 $(ch_{nr} = 1)$	10	5,99
	AP_2 $(ch_{nr} = 6)$	25	
	AP_3 $(ch_{nr} = 11)$	17	
	AP_4 $(ch_{nr} = 1)$	21	
	AP_5 $(ch_{nr} = 6)$	9	
	AP_6 $(ch_{nr} = 11)$	11	
F_{c_2}	AP_1 $(ch_{nr} = 1)$	21	13,55
	AP_2 $(ch_{nr} = 6)$	46	
	AP_3 $(ch_{nr} = 11)$	12	
	AP_4 $(ch_{nr} = 1)$	14	
F_{c_3}	AP_1 $(ch_{nr} = 1)$	21	3,49
	AP_2 $(ch_{nr} = 6)$	25	
	AP_3 $(ch_{nr} = 11)$	19	
	AP_4 $(ch_{nr} = 1)$	28	
	AP_1 $(ch_{nr} = 1)$	30	12,77
	AP_2 $(ch_{nr} = 6)$	33	
	AP_3 $(ch_{nr} = 11)$	3	
	AP_4 $(ch_{nr} = 1)$	19	
	AP_5 $(ch_{nr} = 6)$	1	
	AP_6 $(ch_{nr} = 11)$	7	
F_{c_4}	AP_1 $(ch_{nr} = 1)$	1	29,09
	AP_2 $(ch_{nr} = 6)$	72	
	AP_3 $(ch_{nr} = 11)$	19	
	AP_4 $(ch_{nr} = 1)$	1	
F_{c_7}	$AP_1(ch_{nr} = 1)$	63	16,50
	AP_2 $(ch_{nr} = 6)$	30	
F_{c_8}/N_{AP}	AP_1 $(ch_{nr} = 1)$	5	23,48
	AP_2 $(ch_{nr} = 6)$	4	
	AP_3 $(ch_{nr} = 11)$	65	
	AP_4 $(ch_{nr} = 1)$	5	
	AP_5 $(ch_{nr} = 6)$	14	
	AP_1 $(ch_{nr} = 1)$	34	10,14
	AP_2 $(ch_{nr} = 6)$	5	
	AP_3 $(ch_{nr} = 11)$	20	
	AP_4 $(ch_{nr} = 1)$	19	
	AP_5 $(ch_{nr} = 6)$	9	
	AP_6 $(ch_{nr} = 11)$	6	

Z analizy tabeli 4.8 wynika, że w przypadku Fc_8 /5, tzn. przy 5 punktach AP, algorytm kukułki CS przydzielił jednemu tylko punktowi AP_3 aż 58 stacji (80%), zaś pozostałym czterem zaledwie po 4 stacje ST. Bardziej równomierny przydział stacji miał miejsce w przypadku $Fc_8/6$, gdzie trzem punktom AP, pracującym w nieinterferujących ze sobą kanałach radiowych,

algorytm kukułki CS przydzielił od 17 do 20 stacji ST, zaś pozostałym trzem od 3 do 10 stacji.

W przypadku 93 stacji ST, rozlokowanych w pomieszczeniach i na korytarzu jednego piętra budynku (rys. 4.3), dla wszystkich analizowanych funkcji kryterialnych – za wyjątkiem funkcji Fc_7 – algorytm kukułki CS znajdował rozwiązania optymalne wymagające użycia nie mniej niż czterech punktów dostępu AP. Dla funkcji kryterialnej Fc_7 już przy dwóch punktach AP otrzymano wartość maksymalną tej funkcji, równą wielokrotności rozważanej liczby stacji i maksymalnej szybkości transmisji. Dla funkcji kryterialnej Fc_8/N_{AP}, opartej na podziale zasięgu radiowego punktów dostępu AP na obszary, w których szybkości transmisji zależą od wartości $SINR$ na wejściu stacji ST, algorytm kukułki CS znalazł – podobnie jak w scenariuszu z 74 stacjami – dwa rozwiązania zadania optymalizacji, tzn. dla 5 i 6 punktów dostępu AP (tab. 4.9), różniące się w granicach założonej zbieżności algorytmu, czyli o mniej niż 1%.

Dodatkowo wartości funkcji kryterialnych i innych wskaźników uzyskane w ramach zadania optymalizacji jednokryterialnej SOO (Fc_5^0) dla największego obszaru i $N_{ST} = 93$ stacji ST (rys. 4.3) przedstawiono w tabeli 4.11.

Tabela 4.11. Wartości funkcji kryterialnych uzyskane w ramach zadania SOO maksymalizacji funkcji Fc_5^0 dla N_{ST}=93 stacji ST (rys. 4.3) i najlepszych rozwiązań GQ_i^{best}

Optymalizacja SOO($F_{c_5}^0$)	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	GQ_i^{best}	PM [(Mb/s) m^2]
$P_S^{thr}(54)$= –68 dBm	686,68	70,60	5016,00	8,52	36;38;19	0,63	339,20
	587,26	98,89	5022,00	10,11	13;21;40;0;19	0,66	348,46
	628,94	98,40	5022,00	6,26	21;0;0;19;19;34	0,69	465,40
$P_S^{thr}(36)$= –75 dBm	737,91	71,31	5022,00	9,42	19;42;32	0,63	366,74
	611,99	71,60	5022,00	13,59	24;50;0;19	0,67	256,01
	630,21	98,40	5022,00	6,26	0;21;34;19;19	0,69	459,60
	691,16	93,28	4824,00	8,26	0;21;0;13;36;23	0,66	**704,73**

4.2. Przykłady zadań optymalizacji wielokrytrialnej MOO w planowaniu sieci WLAN

Planowanie sieci WLAN z infrastrukturą standardu IEEE 802.11b/g w pomieszczeniach (rys. 4.1) przeprowadzono, także formułując przykładowe zadania optymalizacji wielokryterialnej MOO [551].

W tym celu, analogicznie jak dla optymalizacji jednokryterialnej SOO, wykorzystano:

- model propagacyjny MWM [511], [514] o parametrach wyznaczonych empirycznie [548],
- model matematyczny przepustowości sieci [492], uwzględniający: początkowy rozmiar okna rywalizacji, maksymalną liczbę stanów procedury *backoff* oraz liczbę ponowień transmisji pakietów dla mechanizmu DCF protokołu CSMA/CA,
- transmisję UDP, opartą na pakietach o ładunku 1472 B i 1500 bajtowej jednostce MTU (*Maximum Transmission Unit*),
- wskaźnik jakości globalnej GQ metody MUZ [544] do budowy rankingu i analizy rozwiązań najlepszych GQ_i^{best} (3.5),
- wskaźnik wydajności PM (2.9) [491] do oceny i porównania otrzymanych rozwiązań.

Optymalizację wielokryterialną MOO przeprowadzono za pomocą wybranych w rozdziale trzecim algorytmów rojowych: kukułki MOCS [448], [552], [553] oraz optymalizacji rojem cząstek MOPSO [446].

Opierając się na zestawach funkcji kryterialnych: (Fc_1, Fc_8) [548], (Fc_1, Fc_7) oraz na scenariuszach: $(Fc_8, \sigma_{\overline{N_{ST}}})$, $(Fc_7, Fc_8, \sigma_{\overline{N_{ST}}})$, (Fc_1, Fc_7, Fc_8), poszukiwano – za pomocą wybranych algorytmów optymalizacji wielokryterialnej MOO (tab. 3.7) – najlepszego przydziału stacji ST poszczególnym punktom AP ($N_{AP} = \{1 \div 6\}$) – w liczbie od jednego do sześciu.

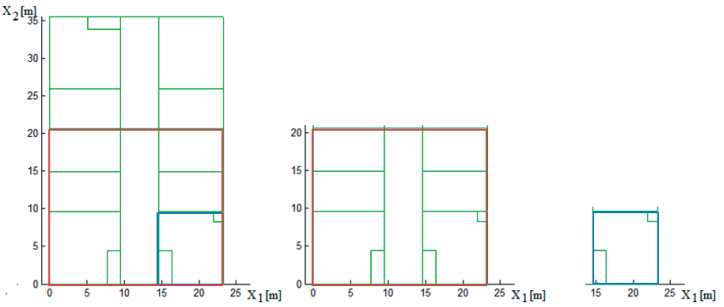
W poszukiwaniu ekstremów wymienionych zestawów funkcji kryterialnych wykorzystano algorytmy MOPSO [446] i MOCS [448] – o parametrach przedstawionych w tabeli 3.7 – a to ze względu na ich wzajemne uzupełnianie się. Algorytm MOPSO generował rozwiązania optymalne w sensie Pareto z lepszymi wartościami funkcji zasięgowej Fc_1, zaś MOCS z lepszymi wartościami funkcji Fc_8 [549].

Dodatkowo oprócz optymalizowanych wartości funkcji kryterialnych (stymulant), w konstruowaniu rankingu rozwiązań wzięto pod uwagę odchylenie standardowe ($\sigma_{N_{ST}}$) odnoszące się do średniej liczby stacji ST

przydzielanych punktom AP (destymulantom), jak również uwzględniono obciążenie punktów AP stacjami ST (nominanty), dla których przyjęto najmniejszą $c_{1no} = 1$ i największą $c_{2no} = 20$ liczbę stacji ST – wzór (3.4).

Do oceny i porównania otrzymanych rozwiązań w pierwszej kolejności wybrano wskaźnik wydajności PM (2.9). Wskaźnik PM [491] łączy w sobie zasięg radiowy sieci WLAN oraz daje możliwość uwzględnienia w niej przepustowości osiąganych przez poszczególne stacje ST wymieniające dane z punktami AP [491], [492].

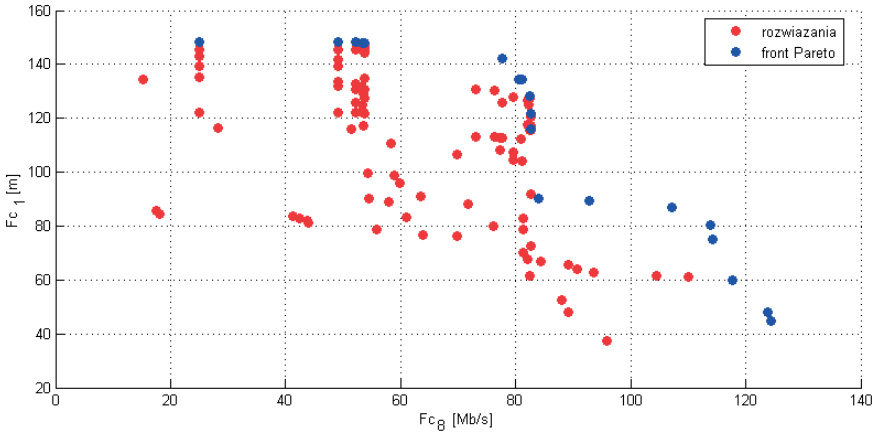
Dla trzech coraz mniejszych obszarów przedstawionych na rysunku 4.4 rozważono rozmieszczenie kolejno: 93, 74 i 20 stacji ST – analogicznie jak dla optymalizacji SOO.



Rysunek 4.4. Obszar projektowanej sieci kolejno dla 93, 74 i 20 stacji ST

Tabela 4.12. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO maksymalizacji zestawu funkcji (Fc_1, Fc_8) dla N_{ST} =20 stacji ST (rys. 3.1) i najlepszych rozwiązań GQ_i^{best} (10 z 22)

Nr	GQ_i^{best}	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	0,84	121,70	82,63	1080,00	0,94	6;6;8	1258,52
2.	0,84	115,39	82,64	1080,00	0,47	7;6;7	1126,11
3.	0,82	128,45	82,44	1080,00	2,49	10;4;6	596,29
4.	0,80	148,16	24,97	1080,00	0,00	20	166,00
5.	0,79	134,42	81,05	1080,00	4,11	2;12;6	661,05
6.	0,79	142,10	77,75	1080,00	4,92	1;13;6	641,70
7.	0,77	134,50	80,94	1080,00	5,25	2;14;4	794,49
8.	0,77	142,14	77,75	1080,00	5,44	1;14;5	691,86
9.	0,75	147,97	53,65	1080,00	5,00	15;5	652,23
10.	0,75	134,55	80,60	1080,00	5,91	2;15;3	946,03

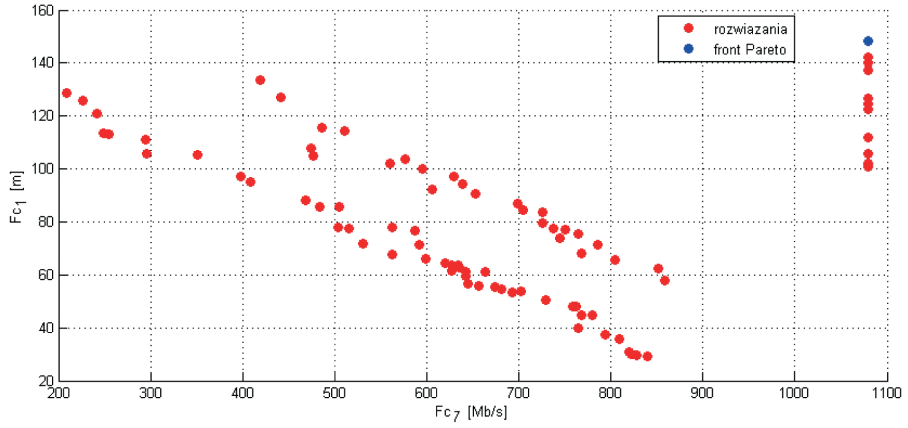


Rysunek 4.5. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla MOO (Fc_1, Fc_8) i N_{ST} =20 stacji ST (rys. 3.1)

Tabela 4.13. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO maksymalizacji zestawu funkcji (Fc_1,Fc_7) dla N_{ST} =20 stacji ST (rys. 3.1)

GQ_i^{best}	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
0,80	148,16	24,97	1080,00	0	20	166,00

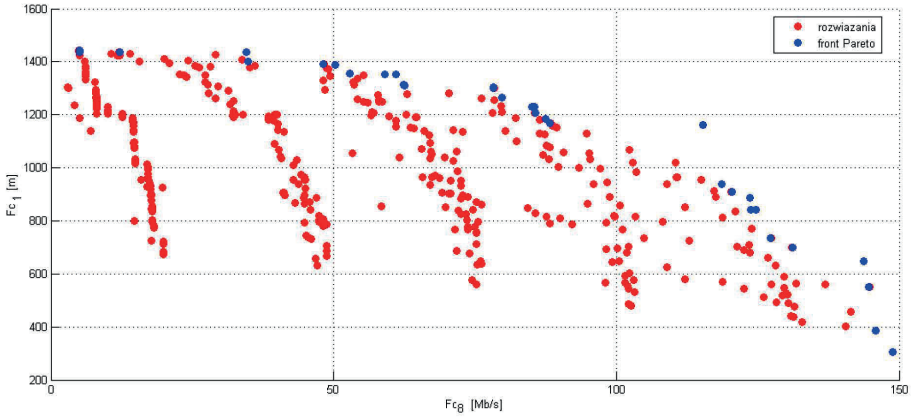
W przedstawionym scenariuszu MOO(Fc_1, Fc_8) otrzymano 22 rozwiązania z 10 najlepszymi GQ_i^{best} (tab. 4.12) oraz dla scenariusza MOO(Fc_1, Fc_7) – jedno rozwiązanie leżące na froncie Pareto i znajdujące się w zbiorze rozwiązań najlepszych (tab. 4.13).



Rysunek 4.6. Rozwiązania i front Pareto (punkt koloru niebieskiego) dla MOO (Fc_1, Fc_7) N_{ST}=20 stacji ST (rys. 3.1)

Tabela 4.14. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO maksymalizacji zestawu funkcji (Fc_1, Fc_7) dla N_{ST}=74 stacji ST (rys. 3.2) i najlepszych rozwiązań GQ_i^{best} (15 z 38)

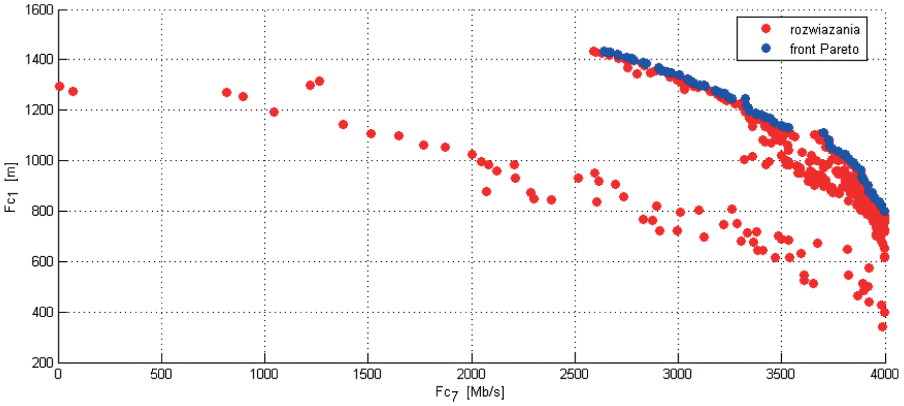
Nr	GQ_i^{best}	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	**PM** [(Mb/s) m^2]
1.	0,76	646,70	143,68	3570,00	6,72	13;5;19;2;20;15	1539,63
2.	0,74	305,85	148,94	3786,00	7,25	13;3;14;3;19;22	475,83
3.	0,74	385,57	145,85	3864,00	11,10	4;4;1;13;33;19	3243,11
4.	0,73	551,57	144,72	3678,00	10,55	30;5;1;2;19;17	1008,22
5.	0,70	841,01	124,64	3564,00	14,78	5;3;43;16;7	1136,72
6.	0,70	842,99	123,80	3546,00	16,76	8;10;48;4;4	1561,11
7.	0,70	734,24	127,32	3960,00	24,13	1;4;63;4;2	1108,66
8.	0,68	888,40	123,58	3540,00	18,18	5;9;51;5;4	1654,38
9.	0,66	910,26	120,36	3570,00	21,10	4;5;57;4;4	2120,44
10.	0,66	1263,76	79,77	3180,00	14,22	1;32;33;8	1703,19
11.	0,65	939,61	118,64	3528,00	23,12	3;2;61;3;5	1938,70
12.	0,64	1184,91	87,47	3384,00	20,74	2;54;11;7	2706,41
13.	0,63	698,19	131,09	2920,00	10,73	15;13;3;6;3;34	1668,02
14.	0,62	1168,41	88,32	3342,00	21,73	3;8;56;7	2434,55
15.	0,62	1162,22	115,33	3282,00	24,12	4;3;63;3;1	3843,08



Rysunek 4.7. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla MOO (Fc_1, Fc_7) i N_{ST} =74 stacji ST (rys. 3.2, 4.4)

Tabela 4.15. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO maksymalizacji zestawu funkcji (Fc_1, Fc_7) dla N_{ST} =74 stacji ST (rys. 3.2) i najlepszych rozwiązań GQ_i^{best} (7 z 69)

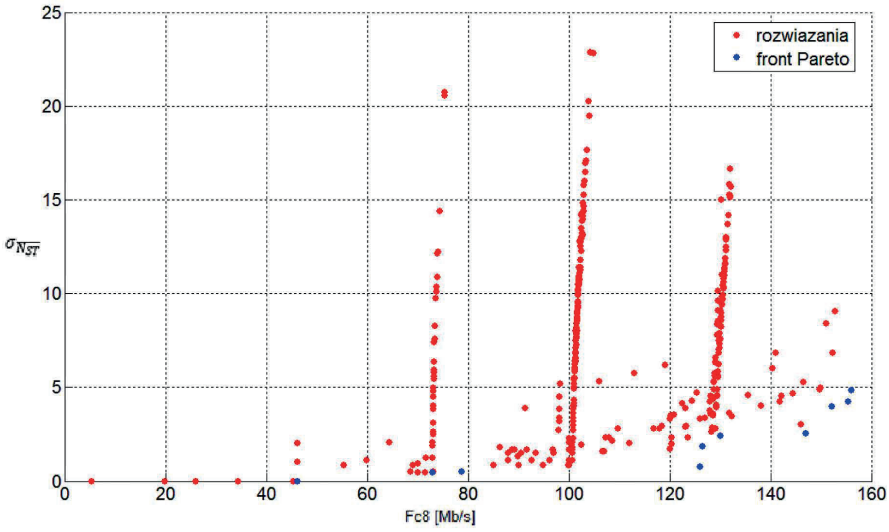
Nr	GQ_i^{best}	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	0,66	930,39	42,83	3894,00	4,00	41;33	426,58
2.	0,65	1244,46	84,35	3324,00	19,41	1;15;51;7	2325,84
3.	0,64	1024,75	69,97	3810,00	20,24	7;53;14	1969,10
4.	0,64	922,44	72,00	3900,00	20,14	8;53;13	1800,51
5.	0,63	801,39	73,99	3996,00	21,20	53;2;19	2842,75
6.	0,61	1131,56	34,28	3534,00	9,00	28;46	528,97
7.	0,60	1324,88	25,52	3046,00	2,00	35;39	424,78



Rysunek 4.8. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla MOO (Fc_1, Fc_7) i N_{ST} =74 stacji ST (rys. 3.2)

Tabela 4.16. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO optymalizacji zestawu dwóch funkcji (Fc_8, $\sigma_{\overline{N_{ST}}}$) dla N_{ST}= 74 stacji ST (rys. 3.2)

Nr	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	501,23	46,15	3996	0	37;37	214,24
2.	592,55	73,04	3996	0,47	24;25;25	450,20
3.	364,19	78,78	3234	0,50	18;19;18;19	429,35
4.	387,55	125,98	3948	0,75	15;15;16;14;14	867,92
5.	332,48	126,46	3972	1,83	13;14;13;17;17	711,72
6.	294,23	130,15	3528	2,43	9;11;14;10;15;15	489,20
7.	285,90	146,94	3798	2,56	14;9;15;9;15;12	414,63
8.	308,95	152,17	3906	3,99	15;8;18;15;7;11	1013,83
9.	312,04	155,40	3966	4,23	3;15;13;15;14;14	551,23
10.	278,98	156,10	3960	4,82	5;18;19;11;11;10	420,39

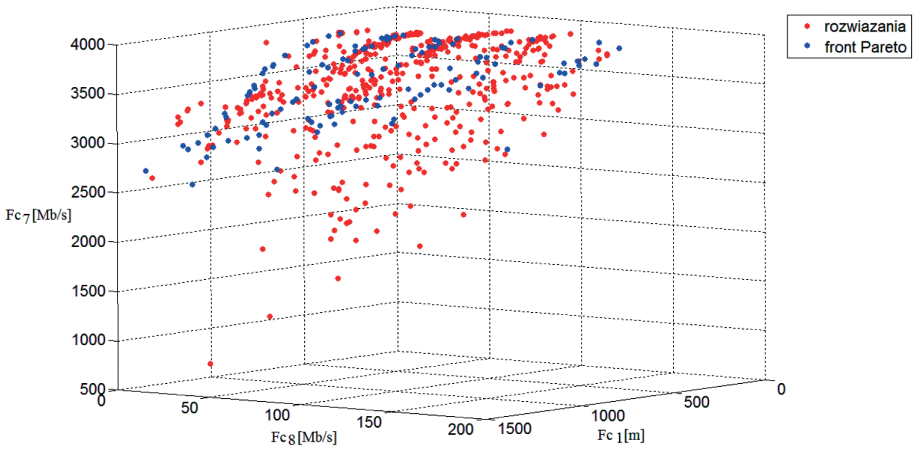


Rysunek 4.9. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla MOO (Fc_8, $\sigma_{\overline{N_{ST}}}$) i N_{ST}=74 stacji ST (rys. 3.2)

W scenariuszu MOO (F_{c_1}, F_{c_8}) z 74 stacjami ST (rys. 3.2) otrzymano 38 rozwiązań leżących na froncie Pareto (rys. 4.7) i należących do 15 najlepszych rozwiązań (tab. 4.14). Z kolei dla MOO (F_{c_1}, F_{c_7}) otrzymano 69 rozwiązań (rys. 4.8) optymalnych w sensie Pareto z siedmioma rozwiązaniami znajdującymi się w zbiorze rozwiązań najlepszych GQ_i^{best} (tab. 4.15). Dodatkowo dla zadania optymalizacji MOO (F_{c_8}, $\sigma_{\bar{N}ST}$) uzyskano dziesięć rozwiązań, które znalazły się na froncie Pareto (tab. 4.16, rys. 4.9). Ale tylko w jednym z tych rozwiązań (nr 3) w sieci WLAN znalazły się cztery punkty AP z najmniejszą, równą 3234 Mb/s, wartością funkcji kryterialnej F_{c_7}. Dodatkowo front Pareto ominął całą grupę rozwiązań znajdujących się między rozwiązaniami (rys. 4.9) odpowiednio nr 3 i nr 4 z tabeli 4.16 (między wartościami F_{c_8} wynoszącymi 78,78 Mb/s a 125,98 Mb/s).

Dlatego też zdecydowano się rozważyć dwa nowe zadania optymalizacji MOO (F_{c_1}, F_{c_7}, F_{c_8}) oraz MOO (F_{c_7}, F_{c_8}, $\sigma_{\bar{N}ST}$), oparte, nie jak dotychczas na dwóch, a na trzech funkcjach kryterialnych. W przypadku zadania MOO (F_{c_1}, F_{c_7}, F_{c_8}) front Pareto objął aż 116 rozwiązań (rys. 4.10). Z czego w 8 rozwiązaniach, dla których funkcja F_{c_7} osiągnęła wartość maksymalną (tab. 4.17) w sieci WLAN pracowały cztery punkty AP. Przyglądając się tym ośmiu rozwiązaniom, można zauważyć – wyrażającą się dużą wartością parametru $\sigma_{\bar{N}ST}$ (tab. 4.17) – znaczną dysproporcję, co do liczby stacji abonenckich ST, przydzielanych tym czterem punktom AP.

Z tego też względu rozważono kolejne, dodatkowe zadanie optymalizacji MOO (F_{c_7}, F_{c_8}, $\sigma_{\bar{N}ST}$).



Rysunek 4.10. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla MOO (F_{c_1}, F_{c_7}, F_{c_8}) i N_{ST} =74 stacji ST (rys. 3.2)

Tabela 4.17. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO maksymalizacji zestawu trzech funkcji MOO (F_{c_1}, F_{c_7}, F_{c_8}) i N_{ST}= 74 stacji ST (rys. 3.2) i 4 punktów AP

Nr	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	530,38	98,69	3996,00	19,25	1;21;49;3	312,83
2.	601,62	102,22	3996,00	24,18	2;60;2;10	2393,42
3.	678,04	102,40	3996,00	21,12	3;54;15;2	914,37
4.	638,05	103,06	3996,00	14,97	6;21;42;5	700,06
5.	598,80	103,09	3996,00	23,58	10;3;59;2	1207,11
6.	642,80	103,46	3996,00	17,63	6;48;16;4	1463,37
7.	698,37	103,90	3996,00	19,44	9;9;52;4	1237,27
8.	551,05	104,04	3996,00	19,40	5;52;9;8	1086,08

Tabela 4.18. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO optymalizacji zestawu trzech funkcji MOO (F_{c_7}, F_{c_8}, $\sigma_{\overline{N_{ST}}}$ dla N_{ST}= 74 stacji ST (rys. 3.2)

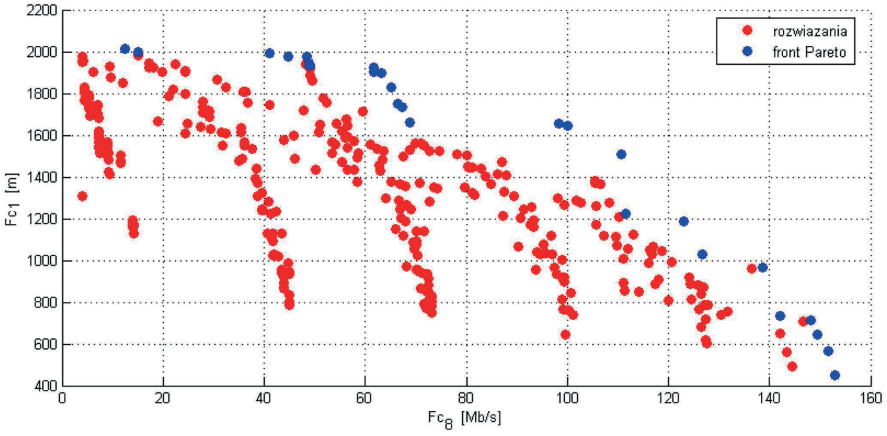
Nr	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	501,23	46,15	3996	0	37;37	214,24
2.	592,55	73,04	3996	0,47	24;25;25	450,20
3.	364,19	78,78	3234	0,50	18;19;18;19	429,35
4.	391,51	100,77	3996	1,12	19;20;18;17	504,16
5.	**458,71**	**100,79**	**3996**	**1,66**	**21;17;19;17**	**864,58**
6.	452,42	100,80	3996	2,06	17;16;20;21	628,31
7.	502,94	100,85	3996	2,69	19;14;20;21	716,62
8.	387,55	125,98	3948	0,75	15;15;16;14;14	867,92
9.	332,48	126,46	3972	1,83	13;14;13;17;17	711,72
10.	402,45	129,19	3996	2,93	10;14;16;15;19	855,00
11.	294,23	130,15	3528	2,43	9,11,14,10,15,15	489,20
12.	285,90	146,94	3798	2,56	14;9;15;9;15;12	414,63
13.	308,95	152,17	3906	3,99	15;8;18;15;7;11	1013,83
14.	312,04	155,40	3966	4,23	3;15;13;15;14;14	551,23
15.	278,98	156,10	3960	4,82	5;18;19;11;11;10	420,39

Zadanie MOO(F_{c_8}, F_{c_7}, $\sigma_{\overline{N_{ST}}}$) doprowadziło do 15 rozwiązań, leżących na froncie Pareto (tab. 4.4.18), wśród których, oprócz 10 wyznaczonych wcześniej w zadaniu MOO (F_{c_8}, $\sigma_{\overline{N_{ST}}}$) (tab. 4.4.16), znalazło się pięć nowych rozwiązań. Spośród tych 5 nowych rozwiązań na uwagę zasługuje

rozwiązanie numer 5 (tab. 4.18). Bowiem w rozwiązaniu tym wszystkie 4 punkty AP rozlokowane zostały w pomieszczeniach, w których znajdowały się stacje ST (rys. 4.2) [551].

Tabela 4.19. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO maksymalizacji zestawu funkcji (F_{c_1}, F_{c_8}) dla N_{ST} =93 stacji ST (rys. 4.3) i najlepszych rozwiązań GQ_i^{best} (8 z 28)

Nr	GQ_i^{best}	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	0,74	711,24	148,10	4884,00	10,06	26,2,30,17,6,12	1146,20
2.	0,72	565,34	151,71	4962,00	10,94	15,17,4,20,2,35	**2013,03**
3.	0,72	1186,58	122,98	4818,00	14,09	5,17,31,38,2	1690,56
4.	0,72	451,56	152,85	4986,00	9,95	3,4,16,32,17,21	1048,57
5.	0,70	646,27	149,56	4704,00	11,83	31,33,5,5,10,9	1033,40
6.	0,66	967,23	138,70	4368,00	16,76	2,10,50,21,1,9	2730,94
7.	0,66	736,14	142,04	4374,00	14,27	6,4,21,3,44,15	5169,51
8.	0,66	1509,16	106,81	4118,00	17,48	2,17,52,8,14	7733,02

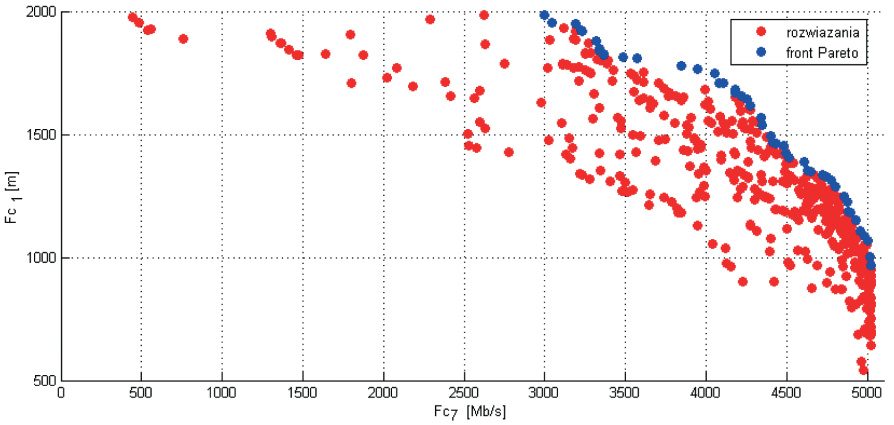


Rysunek 4.11. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla zestawu funkcji MOO (F_{c_1}, F_{c_8}) i N_{ST}=93 stacji ST (rys. 4.3)

W przypadku 93 stacji ST, pracujących w sieci WLAN (rys. 4.3), najlepsze rozwiązania GQ_i^{best} znalezione dla zadania optymalizacji MOO (F_{c_1}, F_{c_8}) zaprezentowano w tabeli 4.19, zaś dla zadania MOO (F_{c_1}, F_{c_7}) w tabeli 4.20.

Tabela 4.20. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO maksymalizacji zestawu funkcji (F_{c_1}, F_{c_8}) dla N_{ST}=93 stacji ST (rys. 4.3) i najlepszych rozwiązań GQ_i^{best} (13 z 47)

Nr	GQ_i^{best}	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\overline{N_{ST}}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	0,69	1068,20	98,30	5004,00	13,99	18,8,46,21	1116,69
2.	0,68	1084,10	97,54	4986,00	13,99	8,18,21,46	**1144,38**
3.	0,67	967,04	71,14	5022,00	7,48	23,29,41	520,56
4.	0,66	1106,20	69,72	4956,00	10,71	29,45,19	743,95
5.	0,63	1313,50	66,16	4776,00	16,27	20,54,19	875,33
6.	0,63	1288,50	66,44	4800,00	16,27	20,54,19	834,85
7.	0,63	1224,80	68,18	4878,00	16,27	20,54,19	800,88
8.	0,63	1186,57	69,05	4884,00	15,58	19,53,21	717,95
9.	0,62	1249,50	66,55	4854,00	16,27	20,54,19	844,44
10.	0,62	1328,70	63,58	4746,00	16,27	20,54,19	867,91
11.	0,62	1184,20	68,19	4896,00	16,27	20,54,19	778,92
12.	0,62	1149,70	69,32	4932,00	16,27	20,54,19	703,58
13.	0,62	1389,89	19,26	4610,00	1,50	48,45	248,86



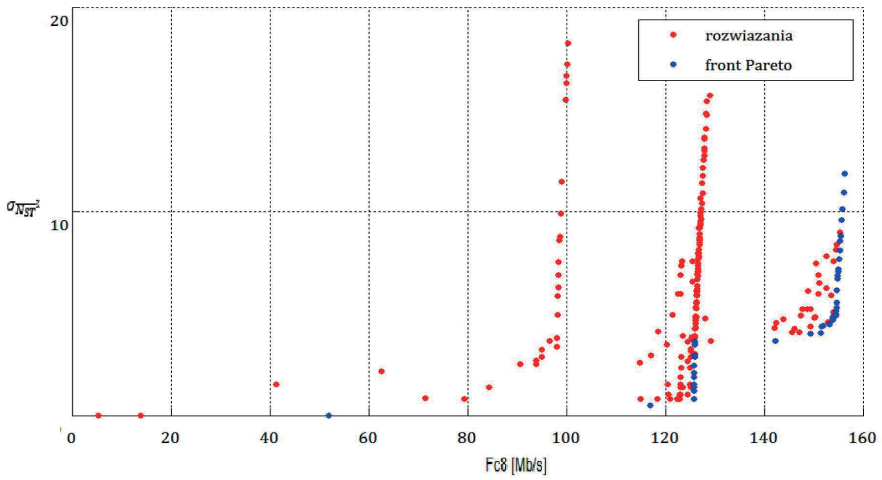
Rysunek 4.12. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla MOO (F_{c_1}, F_{c_7}) i N_{ST}=93 stacji ST (rys. 4.3)

W przypadku N_{ST} = 93 stacji ST (rys. 4.3) dla MOO (F_{c_1}, F_{c_8}) otrzymano 28 rozwiązań Pareto-optymalnych, a dla MOO (F_{c_1}, F_{c_7}) aż 47. Rozwiązań najlepszych GQ_i^{best} było odpowiednio osiem (tab. 4.19) i trzynaście (tab. 4.20).

Ze względu na duże wartości $\sigma_{\bar{N}_{ST}}$ (tab. 4.20) i małe F_{c_8} (tab. 4.21) zdecydowano się znaleźć rozwiązania dla zadania optymalizacji MOO (F_{c_8}, $\sigma_{\bar{N}_{ST}}$).

Tabela 4.21. Wartości wybranych funkcji kryterialnych (front Pareto) uzyskane w ramach zadania MOO optymalizacji zestawu funkcji (F_{c_8}, $\sigma_{\bar{N}_{ST}}$) dla N_{ST}= 93 stacji ST (rys. 4.3) i najlepszych rozwiązań GQ_i^{best} (8 z 43)

Nr	GQ_i^{best}	F_{c_1} [m]	$F_{c_8}=\sum_{j=1}^{N_{AP}}\sum_{k=1}^{N_{ST_j}} S_k$ [Mb/s]	F_{c_7} [Mb/s]	$\sigma_{\bar{N}_{ST}}$	$N_{ST}^{AP_n}$	PM [(Mb/s) m^2]
1.	0,74	674,06	125,90	5022	1,50	21,19,17,17,19	1138,71
2.	0,67	463,09	116,96	4884	0,49	19,18,19,18,19	781,98
3.	0,64	408,93	122,93	4944	0,80	19,19,19,17,19	446,28
4.	0,64	414,82	122,98	4944	1,02	18,19,19,17,20	445,61
5.	0,58	507,77	149,37	4830	5,19	21,10,21,19,14,8	840,23
6.	0,57	460,36	154,81	5022	6,10	13,19,14,17,25,5	612,94
7.	0,54	402,02	128,08	4607	4,72	22,13,19,19,10,10	463,90
8.	0,51	406,12	155,42	5022	8,96	8,19,13,17,32,4	610,68



Rysunek 4.13. Rozwiązania i front Pareto (punkty koloru niebieskiego) dla MOO (F_{c_8}, $\sigma_{\bar{N}_{ST}}$) i N_{ST}=93 stacji ST (rys. 4.3)

W przypadku zadania MOO (F_{c_8}, $\sigma_{\bar{N}_{ST}}$) rozwiązanie numer 1 z tabeli 4.21, odpowiadało rozlokowaniu punktów AP w pomieszczeniach, w których znajdowały się stacje ST.

4.3. Weryfikacja wyników zadań optymalizacji jednokryterialnej SOO oraz wielokryterialnej MOO za pomocą modelu empirycznego

Chcąc ocenić przydatność uzyskanych – rozwiązań zadań optymalizacji jednokryterialnej SOO oraz wielokryterialnej MOO dokonano ich empirycznej weryfikacji [554]. Do tego celu wykorzystano stanowisko pomiarowe umożliwiające prowadzenie, między innymi, pomiarów przepustowości poszczególnych stacji ST oraz punktów AP w środowisku wewnątrzbudynkowym – dokładny opis modelu empirycznego, wykonanego w formie laboratoryjnego stanowiska pomiarowego oraz metodyki przeprowadzenia pomiarów przedstawiono w pracach: [551], [553], [554], [555], [556], [557], [558]. W bieżącym podrozdziale przedstawiono wyniki końcowe rozważań.

Do ceny efektywności wybranych – optymalnych w sensie przyjętych dla zadań SOO i MOO kryteriów optymalizacji – konfiguracji sieci WLAN standardu IEEE 802.11 z infrastrukturą wykorzystano wskaźnik PM (2.9), którego wartość wyrażono, jako $\bar{PM}^{\bar{E}} \pm \sigma_{\bar{PM}^{\bar{E}}}$, gdzie $\bar{PM}^{\bar{E}}$ był średnią, a $\sigma_{\bar{PM}^{\bar{E}}}$ estymatorem niepewności tej średniej, a także wykorzystano wartości indeksów $\bar{JFI}^{\bar{E}} \pm \sigma_{\bar{JFI}^{\bar{E}}}$ i $\bar{CFI}^{\bar{E}} \pm \sigma_{\bar{CFI}^{\bar{E}}}$ sprawiedliwego przydziału zasobów JFI i CFI [468].

Jednak, w pierwszej kolejności, analizę wydajności konkretnych przypadków konfiguracji sieci WLAN oparto na wyznaczonych za pomocą modelu empirycznego (tab. 4.22) wartościach przepustowości stacji ST i punktów AP, liczonych jako $\bar{S}^{\bar{E}} \pm \sigma_{\bar{S}^{\bar{E}}}$. Wartości $\bar{S}^{\bar{E}}$ i $\sigma_{\bar{S}^{\bar{E}}}$ wyznaczono, w każdym z analizowanych przypadków, na podstawie dziesięciu wyników pomiarów wartości chwilowych przepustowości [554].

Tabela 4.22. Dane modelu empirycznego

Nazwa parametru	Wartość parametru
Położenie / liczba stacji N_{ST}	Ustalone wg. scenariuszy dla $N_{ST} \in \{20;74;93\}$
Liczba punktów AP N_{AP}	$N_{AP} \in \{1;2;3;4;5;6\}$
Generator pakietów	TG_1 *Iperf*
Liczba pomiarów w serii N_{meas}	$N_{meas} = 10$
Czas pojedynczego pomiaru T_{meas}	$T_{meas} = 60$ s
Urządzenie zastosowane w (AP/ST)	(AP^8, ST^3, ST^5) (tab. 5.2)
Moc zastępcza promieniowana izotropowo punktu AP $EIRP$	$EIRP$ = 20 dBm
Rozmiar pakietu $L_{DATA\_payload}$	$L_{DATA\_payload}$=1472 B

Na podstawie wartości średnich przepustowości $\bar{S}^{E}$, a także wskaźnika wydajności $\bar{P}\bar{M}^{\bar{E}}$ i indeksu sprawiedliwości Jaina $\bar{J}\bar{F}\bar{I}^{\bar{E}}$ – traktowanych, jako stymulanty – oraz na podstawie liczby stacji ST $N_{ST} = c_2 = 20$ (3.4) i wymaganej liczby punktów AP N_{AP}, spełniających założenia projektowe (2.1) – traktowanych, jako nominanty – zbudowano następnie, w oparciu o wartości wskaźnika jakości globalnej GQ_i^{best} (3.5), ranking rozwiązań.

Dla uzyskanych w ramach zadań optymalizacji jednokryterialnej SOO, stosujących kryterium zasięgu $F_{c_5^0}$ oraz funkcję kryterialną $F_{c_5^1}$, rozwiązań zastosowano model empiryczny z odpowiednią liczbą stacji ST.

Dla pierwszego scenariusza z N_{ST} =20 stacjami ST (rys. 3.1) oszacowano [64] wymaganą liczbę N_{AP} (2.1) punktów AP jako:

$$N_{AP}^{20ST} = \frac{\sum_{i=1}^{N_{ST}} M_{TR_i} \cdot \nu_i}{M_{TR}^{max} \cdot TUL} = \frac{\sum_{i=1}^{N_{ST}} M_{TR_i} \cdot \nu_i}{S_{TUL}} = \left\lceil \frac{20 \cdot 6 \cdot 0{,}2}{54 \cdot 0{,}5} \right\rceil = 1, \tag{4.3}$$

Analogicznie dla scenariuszy z 74 i 93 stacjami ST, wymaganą liczbę N_{AP} punktów AP obliczono, odpowiednio, jako:

$$N_{AP}^{74ST} = \left\lceil \frac{74 \cdot 6 \cdot 0{,}2}{54 \cdot 0{,}5} \right\rceil = 4, \tag{4.4}$$

$$N_{AP}^{93ST} = \left\lceil \frac{93 \cdot 6 \cdot 0{,}2}{54 \cdot 0{,}5} \right\rceil = 5. \tag{4.5}$$

Liczba oszacowanych punktów AP (4.3)÷(4.5) odpowiadała liczbie pomieszczeń, w których rozmieszczono stacje ST. W przypadku zastosowania Przyjęto ostrożne założenia, mające umożliwić transmisję UDP, np. „telefonii internetowej" VoIP (*Voice over Internet Protocol*), plików TFTP (*Trivial File Transfer Protocol*), gier sieciowych czy skompresowanej transmisji wideo w standardzie MPEG4.

brak tekstu

Opierając się na wynikach optymalizacji jednokryterialnej SOO (tab. 4.5), w scenariuszu z N_{ST} =20 stacjami ST (rys. 3.1, rys. 4.4), rozważono od jednego do trzech punktów AP umieszczonych tylko w jednym pomieszczeniu. Dla „zasięgowych" funkcji kryterialnych $F_{c_7^0}, F_{c_8^1}$, oraz dla F_{c_7} i F_{c_8}, ich maksima przypadały dla jednego punktu AP zlokalizowanego w dowolnym miejscu rozważanego pomieszczenia. Wówczas dla wszystkich stacji ST znajdujących się w tym pomieszczeniu udało się osiągnąć szybkości transmisji $M_{TR} = M_{TR}^{max}$=54 Mb/s.

Z analizy tabeli 4.23 wynika, że sześć przypadków GQ_i^{best} miało miejsce dla zadań optymalizacji SOO lub MOO oraz funkcji kryterialnych F_{c_4} lub F_{c_8}, w których uwzględniono przepustowości poszczególnych stacji ST. Dla N_{ST} =20 stacji ST i analizowanego środowiska radiokomunikacyjnego

(rys. 3.1) z rezultatów uzyskanych w ramach zadań optymalizacji SOO oraz MOO wynika, że wśród rozwiązań „najlepszych" GQ_i^{best} przeważały te z trzema punktami AP.

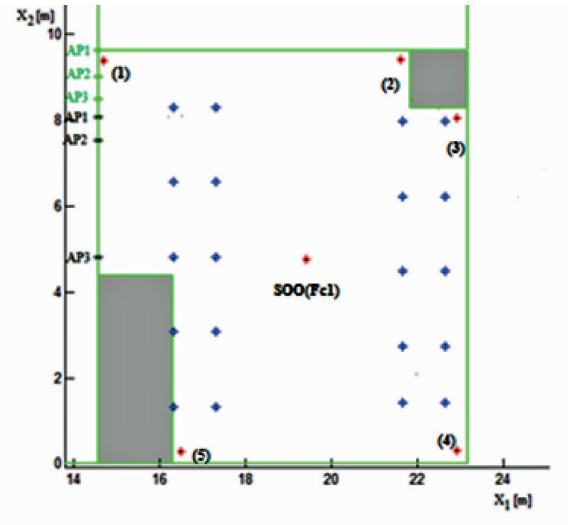
Tabela 4.23. Najlepsze rozwiązaniaGQ_i^{best} dla modelu empirycznego i N_{ST}=20 stacji ST (rys. 3.1)

Nr	Zadanie optymalizacji	GQ_i^{best}	$\overline{S^E}$ [Mb/s]	$\overline{PM^E}$ [(Mb/s)m^2]	$\overline{JFI^E}$	$\overline{CFI^E}$ [Mb/s]	Liczba stacji $N_{ST}^{AP_n}$	$\sigma_{\overline{N_{ST}}}$
1.	MOO(F_{c_1}, F_{c_8})	0,55	71	963	0,97	69	7,6,7	0,47
2.	MOO(F_{c_1}, F_{c_8})	0,53	67	1036	0,94	63	6,6,8	0,94
3.	SOO(F_{c_8})	0,53	73	243	0,97	69	7,7,6	0,47
4.	SOO(F_{c_3})	0,51	71	3698	0,22	16	1,1,18	8,01
5.	SOO(F_{c_4})	0,50	71	3646	0,22	16	1,1,18	8,01
6.	MOO(F_{c_1}, F_{c_7}) MOO(F_{c_1}, F_{c_8})	0,50	24	163	0,94	23	20	0,00
7.	SOO(F_{c_8})	0,50	24	42	0,94	23	20	0,00
8.	SOO($F_{c_5}^0$)	0,50	24	163	0,94	23	20	0,00
9.	SOO($F_{c_5}^1$)/SOO(F_{c_7})	0,50	68	305	0,94	63	7,5,8	1,25
10.	SOO(F_{c_1})	0,50	24	28	0,94	23	20	0,00
11.	SOO(F_{c_1})	0,50	68	273	0,93	63	8,5,7	1,25

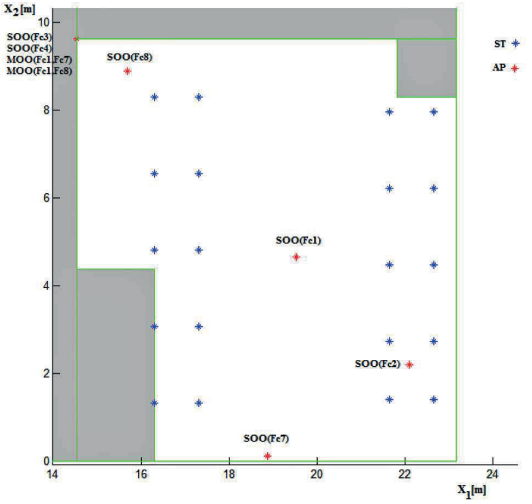
Spośród jedenastu najlepszych rozwiązań, zaprezentowanych w tabeli 4.23, cztery z nich, już dla jednego punktu AP, spełniały określone wzorem (4.3) wymagania stawiane sieci WLAN pracującej w środowisku radiokomunikacyjnym z rysunku 3.1.

Największe wartości $\bar{P}\bar{M}^E$, przy relatywnie małej wartości indeksu $\bar{C}\bar{F}\bar{I}^E$, osiągnięto dla zadań optymalizacji SOO (F_{c_3}) i SOO (F_{c_4}) oraz dla zadania MOO (F_{c_1}, F_{c_8}) (tab. 4.).

Na rysunkach 4.14 i 4.15 zilustrowano najlepsze rozwiązania z tabeli 4.. ???
Na uwagę zasługuje rozwiązanie uzyskane dla funkcji kryterialne F_{c_1}, dla którego przy jednym punkcie AP algorytm CS wybrał dla tego punktu środek pomieszczenia, podczas gdy rozwiązania otrzymane w ramach zadań: SOO (F_{c_3}), SOO (F_{c_4}), MOO (F_{c_1}, F_{c_7}) i MOO (F_{c_1}, F_{c_8}) z udziałem także jednego punktu AP wskazywały dla niego jeden z narożników rozważanego obszaru (rys. 4.15).



Rysunek 4.14. Rozmieszczenie punktów AP dla trzech przypadków (nr 1, nr 2 i nr 5 z tabeli 4.23) dla N_{ST}=20 stacji ST (rys. 3.1)



Rysunek 4.15. Rozmieszczenie punktów AP dla N_{ST}=20 stacji ST (rys. 3.1)

Jeżeli założyć – opierając się na wzorze (4.3) – że wymagana, minimalna liczba punktów AP w sieci z rysunku 3.1 jest równa jeden, to rozwiązanie nr 6 (tab. 4.23) można przyjąć za rozwiązanie jak najbardziej odpowiednie (rys. 4.15).

Wyniki uzyskane na drodze pomiarowej dla sieci WLAN o N_{ST}=74 stacjach ST pracujących w środowisku radiokomunikacyjnym z rysunku 3.2 poddano analizie MUZ.

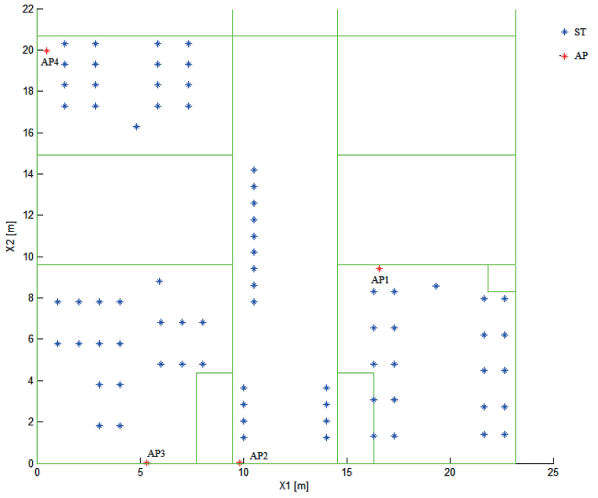
Tabela 4.24. Dziesięć najlepszych rozwiązań GQ_i^{best} dla modelu empirycznego i N_{ST}=74 stacji ST (rys. 3.2)

Nr	Zadanie optymalizacji	GQ_i^{best}	$\overline{S^E}$ [Mb/s]	$\overline{PM^E}$ [(Mb/s)m^2]	$\overline{JFI^E}$	$\overline{CFI^E}$ [Mb/s]	Liczba stacji $N_{ST}^{AP_n}$	$\sigma_{\overline{N_{ST}}}$
1.	MOO(F_{c_1}, F_{c_8})	0,55	71	963	0,97	69	7,6,7	0,47
2.	MOO(F_{c_1}, F_{c_8})	0,53	67	1036	0,94	63	6,6,8	0,94
3.	SOO(F_{c_8})	0,53	73	243	0,97	69	7,7,6	0,47
4.	SOO(F_{c_3})	0,51	71	3698	0,22	16	1,1,18	8,01
5.	SOO(F_{c_4})	0,50	71	3646	0,22	16	1,1,18	8,01
6.	MOO(F_{c_1}, F_{c_7}) MOO(F_{c_1}, F_{c_8})	0,50	24	163	0,94	23	20	0,00
7.	SOO(F_{c_8})	0,50	24	42	0,94	23	20	0,00
8.	SOO($F_{c_5}^0$)	0,50	24	163	0,94	23	20	0,00
9.	SOO($F_{c_5}^1$)/SOO(F_{c_7})	0,50	68	305	0,94	63	7,5,8	1,25
10.	SOO(F_{c_1})	0,50	24	28	0,94	23	20	0,00
11.	SOO(F_{c_1})	0,50	68	273	0,93	63	8,5,7	1,25

Analizując najlepsze rozwiązania GQ_i^{best}(tab. 4.24) można zauważyć, że największą wartość przepustowości $\bar{S}^E$ osiągnięto w zadaniach optymalizacji wielokryterialnej o numerach 8 i 9. Wśród rozwiązań najlepszych (tab. 4.24) znalazło się osiem rozwiązań zadania optymalizacji wielokryterialnej MOO (F_{c_7}, F_{c_8}, $\sigma_{\bar{N}_{ST}}$), w tym trzy rozwiązania MOO (F_{c_8}, $\sigma_{\bar{N}_{ST}}$). Przypadek z największą wartością F_{c_3} wskaźnika PM dotyczył rozwiązania zadania SOO (F_{c_3}). Na uwagę zasługuje także rozwiązanie nr 4 (tab. 4.24), które otrzymano w ramach zadania optymalizacji SOO ($F_{c_5}^0$).

Spośród dziesięciu najlepszych rozwiązań (tab. 4.24) siedem spełniło przyjęte założenia, gdyż wymagana liczba N_{AP} punktów AP nie przekroczyła czterech (4.4).

Na rysunku 4.16 zilustrowano najlepsze rozwiązanie zadania optymalizacji, czyli – zamieszczonego w tabeli 4.24 – zadania nr 1.



Rysunek 4.16. Rozmieszczenie czterech punktów AP (nr 1 tab. 4.24) dla N_{ST}=74 stacji ST

Tabela 4.25. Najlepsze rozwiązania GQ_i^{best} dla modelu empirycznego i N_{ST}=93 stacji ST (rys. 4.3)

Nr	Zadanie optymalizacji	GQ_i^{best}	$\overline{S^E}$ [Mb/s]	$\overline{PM^E}$ [(Mb/s)m^2]	$\overline{JFI^E}$	$\overline{CFI^E}$ [Mb/s]	Liczba stacji $N_{ST}^{AP_n}$	$\sigma_{\overline{N_{ST}}}$
1.	MOO($F_{c_8}, \sigma_{\overline{N_{ST}}}$)	0,75	126	871	0,96	121	19,18,19,18,19	0,49
2.	MOO($F_{c_8}, \sigma_{\overline{N_{ST}}}$)	0,75	126	462	0,96	120	19,19,19,17,19	0,80
3.	MOO($F_{c_8}, \sigma_{\overline{N_{ST}}}$)	0,75	124	462	0,96	119	18,19,19,17,20	1,02
4.	MOO($F_{c_8}, \sigma_{\overline{N_{ST}}}$)	0,74	119	1086	0,95	114	21,19,17,17,19	1,50
5.	SOO(F_{c_7})	0,69	112	450	0,83	93	10,19,28,17,19	5,75
6.	MOO($F_{c_8}, \sigma_{\overline{N_{ST}}}$)	0,66	133	520	0,59	78	8,19,13,17,32	8,96
7.	SOO($F_{c_5}^1$)	0,62	98	421	0,93	91	19,19,34,21	6,26
8.	SOO(F_{c_3})	0,61	102	2182	0,57	54	5,26,31,15,16	9,09

Wyniki otrzymane na drodze pomiarowej dla sieci WLAN o N_{ST} =93 stacjach ST pracujących w środowisku radiokomunikacyjnym z rysunku 4.3 poddano analizie MUZ.

Spośród ośmiu przedstawionych najlepszych rozwiązań (tab. 4.25) siedem spełniło przyjęte założenia, gdyż wymagana liczba N_{AP} punktów AP wyniosła pięć (4.5).

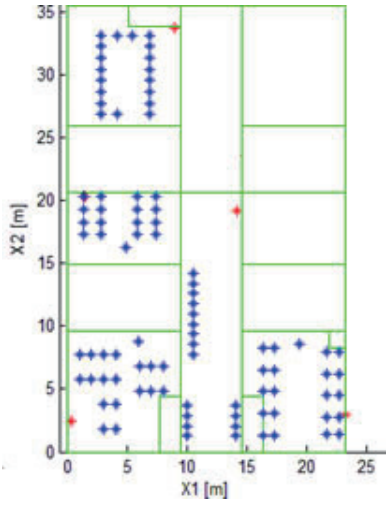
Natomiast wśród rozwiązań zadania optymalizacji SOO) ($F_{c_5}^{0}$) znalazło się rozwiązanie otrzymane dla $P_{\S}^{thr}$ (36) (rys. 4.2, tab. 4.11) już przy czterech punktach AP.

Podobnie jak we wcześniej analizowanych przypadkach sieci WLAN, także dla N_{ST} =93 stacji ST pracujących w środowisku wewnątrzbudynkowym z rysunku 4.3, największe przepustowości $\bar{S}^{E}$, osiągnięto rozwiązując zadanie optymalizacji jednokryterialnej SOO (F_{c_8}). Jakkolwiek, rozwiązanie to nie znalazło się w zbiorze rozwiązań najlepszych. Podobnie stało się z rozwiązaniem, dla którego, że pomimo bardzo dużych wartości wskaźnika $\bar{P}\bar{M}^{E}$ dla zadania optymalizacji jednokryterialnej: SOO (F_{c_3}) oraz SOO (F_{c_4}) nie znalazły się one w zbiorze najlepszych rozwiązań (tab. 4.25) [558]. ???

Spośród najlepszych rozwiązań na uwagę zasługują rozwiązania zadania MOO($F_{c_8}, \sigma_{\bar{N}^{ST}}$), dla których osiągnięto duże wartości wskaźnika $\bar{P}\bar{M}^{E}$ oraz małe $\sigma_{\bar{N}}$.

Zastosowanie zadania optymalizacji MOO($F_{c_8}, \sigma_{\bar{N}^{ST}}$), doprowadziło do rezygnacji z wprowadzania obszarów (stref) zabronionych w obszarze planowanej sieci [553].

Na rysunku 4.17 zilustrowano rozwiązanie nr 4 z tabeli 4.25, które przy dużej wartosci GQ_i^{best}osiągnęło także dużą wartość $\bar{P}\bar{M}^{E}$, a punkty AP zostały rozlokowane tylko w tych pomieszczeniach, w których znajdowały się stacje ST.



Rysunek 4.17. Przykładowe rozmieszczenie 5 punktów AP (nr 4 tab. 4.25) w sieci WLAN dla N_{ST}=93 stacji ST (rys. 4.3) jako rozwiązanie zadania optymalizacji MOO ($F_{c_8}, \sigma_{\bar{N}^{ST}}$)

4.4. Podsumowanie

Optymalizacja wielokryterialna MOO, oparta na wielu różnych – niekiedy nawet przeciwstawnych funkcjach kryterialnych – pozwala na lepsze, bardziej wydajne wykorzystanie możliwości urządzeń i oprogramowania w nowo planowanych sieciach WLAN. W niniejszym rozdziale oceniono i porównano ze sobą rozwiązania uzyskane w ramach zadań optymalizacji wielokryterialnej MOO z rozwiązaniami dla wybranych zadań optymalizacji jednokryterialnj SOO. Na podstawie otrzymanych rezultatów badań utworzono front Pareto, a do oceny wyników optymalizacji wielokryterialnej MOO zastosowano metodę MUZ, która pozwoliła – na podstawie rankingu wskaźnika GQ_i^{best} uwzględniającego wartości $\bar{S}^{\bar{E}}$, $\bar{P}\bar{M}^{\bar{E}}$, $\bar{J}\bar{F}\bar{I}^{\bar{E}}$, liczbę stacji ST przydzielonych poszczególnym punktom AP oraz wymaganą liczbę punktów AP N_{AP}, spełniających założenia projektowe (2.1) – wskazać rozwiązania najlepsze. A docelowo umożliwiła oszacowanie dla zadanej liczby stacji ST, pracujących w wybranym wewnatrzbudynkowym środowisku radiokomunikacyjnym optymalnej liczby punktów AP oraz ich wzajemne rozmieszczenie i konfigurację w nowo planowanej sieci WLAN standardu IEEE 802.11 z infrastrukturą.

Oceniając rozwiązania zadań optymalizacji uzyskane z zastosowaniem różnych zestawów funkcji, można stwierdzić, że we wszystkich analizowanych w tym rozdziale przypadkach dla funkcji kryterialnej F_{c_8} uwzględniającej przepustowość sieci WLAN rozwiązania i front Pareto rozwiązań uzależniony był od liczby punktów AP (rys. 4.5, 4.7 i 4.11). W zadaniach wykorzystujących optymalizację MOO (F_{c_1}, F_{c_7}) rozwiązania układały się bardziej równomiernie (rys. 4.6, 4.8 i 4.12) i osiągnęły swoje maksimum dla $F_{c_7} = N_{ST} \cdot M$.

Analizując rozwiązania najlepsze (GQ_i^{best}) można zauważyć, że uzyskano je jako rozwiązanie zadań: MOO (F_{c_1}, F_{c_5}), MOO (F_{c_7}, F_{c_8}, $\sigma_{\bar{\bar{N}}^{\bar{\bar{ST}}}}$) oraz MOO($F_{c_5}$, $\sigma_{\bar{\bar{N}}^{\bar{\bar{ST}}}}$).

Należy także stwierdzić, że uwzględnienie w funkcji kryterialnej F_{c_8} przepustowości sieci, pozwoliło na bardziej efektywne – w zakresie przyjętych wskaźników oceny – rozmieszczenie punktów dostępu AP w wybranym wewnątrzbudynkowym środowisku radiokomunikacyjnym.

5. PODSUMOWANIE

W celu znalezienia – najlepiej oddającego rzeczywiste warunki propagacyjne środowiska wewnątrzbudynkowego Akademii Tarnowskiej – modelu matematycznego o współczynnikach empirycznych przeprowadzono cały szereg eksperymentów empirycznych opartych na pomiarze poziomu mocy *RSSI* (*fingerprinting*) sygnałów na wejściu odbiornika stacji ST.

Ponieważ współczynnik korelacji liniowej r Pearsona osiągnął największą wartość, równą około 0,97, dla modelu propagacyjnego MWM [551], [558], dlatego też to ten właśnie model uznano za najlepiej odzwierciedlający właściwości propagacyjne analizowanego środowiska wewnątrzbudynkowego. Parametry modelu MWM zostały wykorzystane w autorskim programie komputerowym, umożliwiającym obliczenie wartości wybranych parametrów analizowanej sieci WLAN, w tym do wyznaczenia przepustowości stacji ST.

W monografii poddano analizie porównawczej wybrane modele matematyczne sieci WLAN IEEE 802.11 z infrastrukturą oraz dokonano ich weryfikacji na drodze pomiarowej za pomocą modelu empirycznego. Porównano modele matematyczne $DCF_1(m; CW_{min})$ [88], [89] i $DCF_2(m; CW_{min}; L_r)$ [91] – oparte na łańcuchach Markowa – z modelami matematycznymi z prac [492], [541] i [557] oraz empirycznym [557].

Z porównania przepustowości, wyznaczonych za pomocą modeli matematycznych, wynikało, że modele te w porównaniu z wynikami uzyskanymi na drodze pomiarowej [558] przeszacowywały wartości przepustowości osiągane przez stacje ST.

Porównując rezultaty otrzymane na podstawie modeli matematycznych $DCF_1(6; 16)$ i $DCF_2(6; 16; 7)$ [558] z wynikami pomiarów, stwierdzono, że współczynnik korelacji liniowej r Pearsona osiągał stosunkowo duże wartości, a część wyników była nawet istotna statystycznie z p<0,05 [557], [558]. W przypadku modelu $DCF_2(m; CW_{min}; L_r)$ uzyskano, dla wszystkich punktów AP, zdecydowanie lepsze wartości i to zarówno współczynnika korelacji liniowej r Pearsona wyników obliczeń z pomiarami, jak i współczynnika M [559], [557]. Zastosowane w tych punktach AP oprogramowanie umożliwiało uproszczoną modyfikację parametrów funkcji

DCF. Wszystkie stacje ST działały z tymi samymi parametrami funkcji DCF, skonfigurowanymi – bez rozróżnienia klas ruchu – po stronie punktu AP.

Warto zauważyć, że współczynniki korelacji liniowej r Pearsona dla rozwiązań uzyskanych za pomocą modeli matematycznych były dla modelu $DCF_2(m; CW_{min}; L_r)$ lepsze od rozwiązań uzyskanych dla modelu $DCF_1(m; CW_{min})$, co może świadczyć o celowości zastosowanych w modelu matematycznym $DCF_1(m; CW_{min})$ modyfikacji, dotyczących liczby ponowień transmisji pakietów [558].

W rozdziale trzecim porównano ze sobą wybrane funkcje kryterialne, które mogą być wykorzystane w automatycznym planowaniu sieci WLAN IEEE 802.11 z infrastrukturą. Optymalne wartości tych funkcji wyznaczono za pomocą różnych algorytmów optymalizacji OPA, a zwłaszcza wspierających poszukiwanie ekstremów globalnych wykorzystanych funkcji kryterialnych.

W rozdziale trzecim przedstawiono wyniki ilustrujące wpływ czasu prowadzenia obliczeń – poszukiwania rozwiązania zadania optymalizacyjnego – oraz parametrów algorytmów optymalizacyjnych [92], [464] na dokładność uzyskiwanych rozwiązań.

Analizy porównawcze pokazały, że oparcie planowania sieci WLAN standardu IEEE 802.11 z infrastrukturą pracującą w paśmie ISM 2.4 GHz na funkcjach jednokryterialnych SOO może w niektórych przypadkach prowadzić do zawyżenia wartości wskaźnika wydajności *PM* [9].

W podrozdziale 3.1 porównano ze sobą w sposób arbitralny wybrane algorytmy optymalizacji OPA wspierające poszukiwanie ekstremów różnych funkcji kryterialnych automatycznego planowania sieci WLAN IEEE 802.11 z infrastrukturą [543]. Analiza polegała na porównaniu między sobą skuteczność wybranych algorytmów rozwiązywania zadań optymalizacji jednokryterialnych SOO dla różnych – odgórnie wybranych – funkcji kryterialnych [10], [543].

Dla scenariusza SOO_{3AP}^{20ST} z N_{ST} =20 stacjami ST (rys. 3.1) w przeprowadzonym zadaniu optymalizacji, warto zaważyć zgodność wyników modelu obliczeniowego (matematycznego) z modelem empirycznym [10]. Otrzymane rozwiązanie optymalne (algorytm kukułki CS) znacząco różni się od innych rozwiązań (tab. 3.3).

W tabeli 3.3 oraz w tabeli 3.6 zestawiono wartości oraz czasy rozwiązania zadania optymalizacji dla wybranych funkcji kryterialnych otrzymane za pomocą różnych OPA. Warto zauważyć, że uzyskano rozwiązania, w których punkty AP zostały różnie rozlokowane, jak i różnie obciążone stacjami ST (tab. 3.3).

Mimo, że moduły lub całe systemy zarządzające punktami AP lub siecią WLAN mogą w znaczy sposób modyfikować ustawienia sieci, to rozlokowanie punktów AP, przydział kanałów pracy i ich obciążenie stacjami ST, już na etapie projektowym może wpłynąć na efektywność działania sieci bezprzewodowej.

W tabeli 3.3 oraz w tabeli 3.6 zestawiono wartości oraz czasy optymalizacji wybranych funkcji kryterialnych otrzymane za pomocą wybranych OPA.

W zadaniach optymalizacji jednokryterialnej SOO [9] z zastosowaniem algorytmu SA dla N_{ST} =20 stacji ST (rys. 3.1), najwyższe wartości PM^E uzyskano dla funkcji kryterialnej F_{c_3} oraz F_{c_4} [558] (tab. 4.4.23). Funkcja kryterialna F_{c_4} łączy w sobie zasięg radiowy oraz przepustowość sieci WLAN, a wartości przepustowość uzyskane dla funkcji kryterialnej F_{c_4} są zależne od funkcji dostępu do kanału radiowego DCF [543].

Można zauważyć [9], że wskaźnik *PM* jest wrażliwy na niewielkie zmiany położenia punktów AP – implikujące zmiany liczby ST przydzielonych tym punktom.

Dla różnych rozważanych scenariuszy z różnymi funkcjami kryterialnymi istnieje optymalna liczba punktów AP, którą powinno rozważyć przy planowaniu sieci łączności bezprzewodowej. Często, obliczona liczba punktów AP, pokrywała się z liczbą niezależnych kanałów radiowych w paśmie ISM 2.4 GHz [492], [494], [92].

Przeprowadzone analizy porównawcze, w tym między innymi współrzędnych punktów AP, otrzymanych za pomocą wybranych OPA, pokazały, że najmniej czułą na wybór algorytmu optymalizacyjnego jest funkcja F_{c_1} [543]. Pozostałe funkcje kryterialne wymagały zdecydowanie bardziej starannego doboru i implementacji algorytmów optymalizacyjnych wspierających automatyczne planowanie sieci WLAN z infrastrukturą.

Porównane zostały także dwa algorytmy optymalizacji globalnej: SA oraz GA [92]. Użycie wagi $\Psi \neq 0$ dla funkcji kryterialnej F_{c_5}, prowadziło do ogólnej poprawy zasięgu, jak i utrzymaniu pojemności sieci na odpowiednim poziomie [92]. Parametry analizowanych algorytmów, dla których otrzymano optymalne wartości wykorzystanej funkcji kryterialnej z użyciem algorytmów GA oraz SA. ??? Należy zauważyć, że zastosowanie algorytmu SA dawało rozwiązania z większą wartością funkcji kryterialnej niż GA [92].

Większą skuteczność algorytmu SA w stosunku do GA można także zauważyć analizując wartości funkcji kryterialnej F_{c_1} oraz F_{c_4} (tab. 3.3 i tab. 3.6) w rozwiązaniu zadań optymalizacji SOO_{3AP}^{20ST} oraz SOO_{3AP}^{74ST}.

Porównując ze sobą czasy T_{opt}, po których wybrane algorytmy OPA kończyły proces poszukiwania wartości optymalnych, należy stwierdzić, że obliczenia prowadzone z wykorzystaniem algorytmu symulowango wyżarzania SA oraz algorytmu pszczelego BeA trwały zdecydowanie najdłużej (tab. 3.3 i tab. 3.6).

Z porównania otrzymanych rozwiązań z użyciem modelu matematycznego z wykorzystaniem algorytmu SA dla funkcji kryterialnych: F_{c_5}, F_{c_6} wynika, że użycie funkcji celu (F_{c_6}) uwzględniające mechanizm dostępu do kanału radiowego pozwoliło na bardziej efektywną optymalizację sieci WLAN, zwłaszcza w aspekcie pojemnościowym [492] niż zastosowanie funkcji kryterialnej F_{c_5}. Warto zauważyć jednak, że duża liczba parametrów algorytmu SA, decyduje o szybkości i zbieżności otrzymanego rozwiązania [92].

Rozwiązanie zadania optymalizacji dwukryterialnej przy N_{ST} =20 stacji ST (rys. 3.1), dla zestawów funkcji: MOO (F_{c_1}, F_{c_7}), MOO (F_{c_1}, F_{c_8}), MOO (F_{c_2}, F_{c_7}), MOO ($F_{c_{42}}$, F_{c_8}) [9], prowadziło do poprawy wyników $\bar{S}^{\bar{E}} \pm \sigma_{S^{\bar{E}}}$ w porównaniu do optymalizacji jednokryterialnej SOO.

W większości przypadków wzrosły także $\bar{JFI}^{\bar{E}} \pm \sigma_{\bar{JFI}^{\bar{E}}}$ oraz $\bar{CFI}^{\bar{E}} \pm \sigma_{\bar{CFI}^{\bar{E}}}$ [9]. ???

Dla przypadku MOO (F_{c_2}, F_{c_8}) uzyskano także największą wartość indeksu sprawiedliwości $\bar{JFI}^{\bar{E}} \pm \sigma_{\bar{JFI}^{\bar{E}}}$ oraz $\bar{FI}^{\bar{E}} \pm \sigma_{\bar{CFI}^{\bar{E}}}$ [9].

Analizując uzyskane wyniki należy zauważyć, że, uwzględnienie przepustowości sieci w funkcji kryterialnej prowadziło do wzrostu wartości wskaźnika *PM* [9].

Arbitralne rozmieszczenie punktów AP w analizowanych sieciach WLAN [9], wprawdzie nie prowadziło do rozwiązania najlepszego, to jednak niekiedy dawało lepsze rezultaty od tych osiąganych w przypadku oparcia rozmieszczenia punktów AP w sieci na takiej czy innej funkcji kryterialnej.

Analizując rozwiązania najlepsze (GQ_i^{best}) dla poszczególnych scenariuszy, można zauważyć, że uzyskano je jako rozwiązanie zadania optymalizacji: MOO (F_{c_1}, $Fc8_1$), MOO (F_{c_7}, F_{c_8}, $\sigma_{\bar{N}_{\bar{ST}}}$) oraz MOO ($F_{c_8}$, $\sigma_{\bar{N}_{\bar{ST}}}$). Natomiast, zastosowanie w rozważanych zadaniach optymalizacji MOO (F_{c_1}, F_{c_7}) czy MOO(F_{c_1}, F_{c_7}, F_{c_8}) nie doprowadziło do wymaganych rozwiązań.

Ponieważ najczęściej rozwiązanie zadania optymalizacji SOO (F_{c_8}) prowadziło do rozwiązania o największej wartości $\bar{S}^{\bar{E}}$, można wnioskować, że taki typ optymalizacji lub zastosowanie optymalizacji wielokryterialnej uwzględniającej przepustowość pozwoliło na optymalne rozmieszczenie

i skonfigurowanie punktów dostępu w rozważanych scenariuszach testowych dla wewnątrzbudynkowego środowiska radiokomunikacyjnego.

W celu analizy i porównania rozwiązań znajdujących się na froncie Pareto, w pracy wykorzystano metodę MUZ dla wybranych funkcji kryterialnych i parametrów sieci WLAN.

Zaproponowano użycie MUZ dla trzech funkcji kryterialnych: F_{c_1}, F_{c_7}, F_{c_8} (stymulant), a w budowie rankingu rozwiązań wzięto pod uwagę odchylenie standardowe ($\sigma_{\bar{N}\bar{S}\bar{T}}$) odnoszące się do średniej liczby stacji ST przydzielanych punktom AP (destymulantom), jak również uwzględniono obciążenie punktów AP stacjami ST (nominanty).

Finalnie rozwiązania uzyskane dla różnych funkcji kryterialnych i typów optymalizacji przenalizowano metodą MUZ, uwzględniając: $\bar{S}^{\bar{E}}$, $\bar{P}\bar{M}^{\bar{E}}$, $\bar{J}\bar{F}\bar{T}^{\bar{E}}$, wymaganą liczbę punktów AP oraz założoną liczbę stacji ST.

Ze względu na właściwości protokołu CSMA/CA (np. anomalii wydajności [565]) staranne zaplanowanie: konfiguracji, liczby i położenia punktów AP może wpływać na osiąganą efektywność sieci. Etap planistyczny jest również ważny z powodów ekonomicznych, gdyż wyznaczane są miejsca, do których doprowadzone musi zostać okablowanie strukturalne i zasilanie sieciowe.

Pierwszym etapem powinno być uzyskanie informacji dotyczących uwarunkowań lokalnych, przeszkodami ich rodzajami oraz innymi parametrami wpływającymi na tłumienie oraz zasięg sygnału. Jeżeli nałożone na to zostaną założenia dotyczące pojemności czy równoważenie obciążenia obsługiwanego przez punkty AP, można rozszerzyć zasięgową – „klasyczną" metodę planowania o inne kryteria.

Warto zauważyć, że uwzględnienie m.in. przepustowości sieci w funkcji kryterialnej (F_{c_8}), na której oparto planowanie infrastruktury sieci standardu IEEE 802.11, pozwoliło na bardziej efektywne (w zakresie przyjętych wskaźników) zaprojektowanie i rozmieszczenie punktów dostępu AP w wybranym wewnątrzbudynkowym środowisku radiokomunikacyjnym.

Wszystkie wymienione wyżej problemy, zagadnienia i zadania sformułowano w monografii w postaci tezy, mówiącej o tym, że: u*względnienie w funkcji kryterialnej, na której oparto planowanie infrastruktury sieci standardu IEEE 802.11, właściwości protokołów wielodostępu do kanału radiowego, w tym w szczególności protokołu CSMA/CA, pozwalają na optymalne – zarówno z punktu widzenia poszczególnych stacji abonenckich ST, jak i całej sieci WLAN – rozmieszczenie i skonfigurowanie punktów dostępu AP w wybranym wewnątrzbudynkowym środowisku radiokomunikacyjnym.*

6. SPIS LITERATURY

[1] K. Tang, K. Man i S. Kwong, „Wireless communication network design in IC factory," *IEEE Transactions on Industrial Electronics,* tom 48, nr 2, 2001.

[2] A. Zalewski i R. Cegieła, Matlab – obliczenia numeryczne i ich zastosowania, 2002.

[3] W. Findesein, J. Szymanowski i A. Wierzbicki, „Teoria i metody obliczeniowe optymalizacji", BNI, 1980.

[4] D. E. Goldberg i K. Grygiel, Algorytmy genetyczne i ich zastosowania, WNT, 2009.

[5] T. Weise, „Global Optimisation Algorithms – Theory and Application," http://www.it-weise.d, 2008.

[6] J. Kusiak, A. Danielowska-Tułecka i P. Oprocha, Optymalizacja – wybrane metody z przykładami zastosowań, PWN, 2009.

[7] S. Osowski, Sieci neuronowe do przetwarzania informacji, OWPW, 2000.

[8] J. Arabas, „Wykłady z algorytmów ewolucyjnych", WNT, 2004.

[9] A. Pieprzycki i W. Ludwin, „Weryfikacja wybranych metod automatycznego planowania sieci WLAN," w *PT 8-9/2015*, 2015.

[10] A. Pieprzycki i W. Ludwin, „Analiza wybranych rojowych algorytmów optymalizacji w zagadnieniach planowania sieci WLAN," w *KKRRiT*, Poznań, 2017.

[11] J. Kwiecień, Algorytmy stadne w rozwiązaniu wybranych zagadnień optymalizacji dyskretnej i kombinatorycznej, AGH, Kraków, 2015.

[12] X.-S. Yang, Nature-inspired optimisation algorithms, Elsevier, 2014.

[13] E.H. Aarts i J. Korst, Simulated Annealing and Boltzman Machines, Wiley, 1989.

[14] M. K. Heris, „Real-Coded Simulated Annealing (SA) in MATLAB project YPEA106," Yarpiz.com;, 2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/53149-real-coded- simulated-annealing--sa-.

[15] K. Deb i H. Jain, „An Evolutionary Many-Objective Optimisation Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation,* tom 18, nr 4, pp. 577–601, 2014.

[16] D. Horla, Metody obliczeniowe optymalizacji w zadanaich, Poznań: Wydawnictwo Politechniki Poznańskiej, 2016.

[17] „pyOpt – Reference".

[18] „GLPK (GNU Linear Programming Kit)".

[19] P. Krzyżanowski, Obliczenia inżynierskie i naukowe, Warszawa: Wydawnictwa Naukowe PWN, 2012.

[20] S. S. O. Librares, „Software".

[21] „COIN-OR: Computational Infrastructure for Operations Research; Open-source Software for the Operations Research Community".

[22] B. Zeng, L. Gao i X. Li, „Whale Swarm Algorithm for Function Optimisation," w *ICIC International Conference on Intelligent Computing 2017, Lecture Notes in Computer Science*, 2017.

[23] R. Fourer, „Linear programming," 06 2013. [Online]. Available: https://www.zib.de/groetschel/teaching/SS2013/LP-Survey2013.pdf. [Data uzyskania dostępu: 07 11 2023].

[24] K. Schittkowski, „Softwareentwicklung Schittkowski GmbH".

[25] A. Stachurski, „Wprowadzenie do optymalizacji," Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2009.

[26] Z. Jędrzejczyk, K. Kukuła, J. Skrzypek i A. Walkosz, Badania operacyjne w przykładach i zadaniach, Warszawa: PWN, 2014.

[27] „Global Optimisation Toolbox Documentation," MathWorks.

[28] „Yarpiz – Academic Source Codes and Tutorials".

[29] „File Exchange – MATLAB Central".

[30] A. Ostanin, Metody optymalizacji z Matlab, Nakom, 2012.

[31] „Optimisation Toolbox Documentation," [Online]. Available: https://www.mathworks.com/help/optim/. [Data uzyskania dostępu: 08 12 2023].

[32] M. H. Center, „linprog Solve linear programming problems," 2023.

[33] „Mixed-integer linear programming (MILP) – MATLAB intlinprog," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/optim/ug/intlinprog.html?s_tid=d oc_srchtitle. [Data uzyskania dostępu: 08 12 2023].

[34] „File Exchange - MATLAB Central".

[35] „Quadratic programming – MATLAB quadprog," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/optim/ug/quadprog.html?searchHi ghlight=quadprog&s_tid=doc_srchtitle. [Data uzyskania dostępu: 08 12 2023].

[36] „File Exchange – MATLAB Central".

[37] „File Exchange – MATLAB Central".

[38] „Find minimum of constrained nonlinear multivariable function – MATLAB fmincon," Mathworks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/optim/ug/fmincon.html?s_tid=doc _srchtitle. [Data uzyskania dostępu: 08 12 2023].

[39] „Solve minimax constraint problem – MATLAB fminimax," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/optim/ug/fminimax.html?s_tid=d oc_srchtitle. [Data uzyskania dostępu: 30 01 2023].

[40] „Find minimum of semi-infinitely constrained multivariable nonlinear function – MATLAB fseminf," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/optim/ug/fseminf.html?s_tid=doc_srchtitle. [Data uzyskania dostępu: 30 01 2024].

[41] „Root of nonlinear function – MATLAB fzero," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/matlab/ref/fzero.html?searchHighl ight=fzero&s_tid=doc_srchtitle. [Data uzyskania dostępu: 08 12 2023].

[42] „Solve system of nonlinear equations – MATLAB fsolve," MathWorks, [Online]. Available: https://www.mathworks.com/help/optim/ug/fsolve.html. [Data uzyskania dostępu: 08.12.2023].

[43] „lsqcurvefit Solve nonlinear curve-fitting (data-fitting) problems in least-squares sense," File Exchange – MATLAB Central.

[44] „Solve constrained linear least-squares problems – MATLAB lsqlin," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/optim/ug/lsqlin.html?s_tid=doc_s rchtitle. [Data uzyskania dostępu: 30.01.2024].

[45] „Solve nonlinear least-squares (nonlinear data-fitting) problems – MATLAB lsqnonlin," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/optim/ug/lsqnonlin.html?s_tid=do c_srchtitle. [Data uzyskania dostępu: 30 01 2024].

[46] „Solve nonnegative linear least-squares problem – MATLAB lsqnonneg," MathWorks, [Online]. Available: https://www.mathworks.com/help/releases/R2022a/matlab/ref/lsqnonneg.html?s_tid= doc_srchtitle. [Data uzyskania dostępu: 30.01.2024].

[47] „Solve multiobjective goal attainment problems – MATLAB fgoalattain," MathWorks, [Online]. Available: https://www.mathworks.com/help/optim/ug/fgoalattain.html?searchHighlight=fgoalat tain&s_tid=srchtitle_support_results_1_fgoalattain. [Data uzyskania dostępu: 08.12.2023].

[48] G. B. Dantzig, A. Orden i P. Wolfe, „Generalized Simplex Method for Minimizing a Linear Form Under Linear Inequality Restraints," *Pacific Journal Math,* tom 5, pp. 183–195, 1955.

[49] B. Filipowicz, Badania operacyjne. wybrane metody i algorytmy, cz. I, wyd. IV,, Tarnów: Wydawnictwa PWSZ w Tarnowie, 2018.

[50] M. Daneshian, „Linear programming Simplex algorithm," 2024.

[51] P. Filipowicz, Z. Łucki, I. Stach i J. Wąchol, Badania operacyjne, Kraków: AGH Wydawnictwa Naukowo-Dydaktyczne, 2008.

[52] „Solve linear programming problems – MATLAB linprog," [Online]. Available: https://www.mathworks.com/help/optim/ug/linprog.html. [Data uzyskania dostępu: 12.11.2023].

[53] F. Leja, Rachunek różniczkowy i całkowy, Warszawa: PWN, 1976.

[54] R. S. Ghorpade i B. V. Limaye, A Course in Multivariable Calculus and Analysis, New York Dordrecht Heidelberg London: Springer, 2000.

[55] L. R. Childress, Mathematics for Managerial Decision, New Jersey: Prentice-Hall, 1974.

[56] T. P. Zieliński, P. Korohoda i R. Rumian, Cyfrowe przetwarzanie sygnałów w telekomunikacji, Warszawa: PWN, 2014.

[57] J. Izydorczyk, Matlab i podstawy telekomunikacji, Gliwice: Helion, 2017.

[58] „Three-Dimensional Indoor Positioning with 802.11az Fingerprinting and Deep Learning," Mathworks, 2024. [Online]. Available: https://www.mathworks.com/help/wlan/ug/three-dimensional-indoor-positioning-with-802-11az-fingerprinting-and-deep-learning.html. [Data uzyskania dostępu: 18.10.2024].

[59] R. Klempka, B. Świątek i A. Garbacz-Klempka, Programowanie, algorytmy numeryczne i modelowanie w Matlabie, Kraków: Wydawnictwa AGH, 2017.

[60] P. Beling i F. Wasilewski, „Metoda Hooke'a-Jeevsa", [Online]. Available: http://www.optymalizacja.w8.pl/HookeaJeevsa.html. [Data uzyskania dostępu: 24.04.2024].

[61] K. Darko, „HookJeev_method_and_NelderMead_method," MATLAB Central File Exchange, 2024.

[62] J. C. Lagarias, J. A. Reeds, M. H. Wright i P. E. Wright, „Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal of Optimisation,* tom 9, nr 1, p. 112-147, 1998.

[63] Optimizing Nonlinear Functions – MATLAB & Simulink.

[64] P. Gajewski i S. Wszelak, „Technologie bezprzewodowe sieci teleinformatycznych", WKŁ, 2008.

[65] „GitHub – ucnl/UCNLNav: Multilanguage (C#/Matlab/Rust) library for solving navigation (2D/3D)".

[66] „Solve Nonlinear Problem with Many Variables - MATLAB & Simulin," https://www.mathworks.com/.

[67] R. Filipek, „Wprowadzenie do MATLAB," AGH, Kraków.

[68] D.H. Wolpert i W.G. Macready, „No free lunch theorems for Optimisation," *IEEE Transactions on Evolutionary Computation,* tom 1, nr 1, pp. 67–82, 1997.

[69] P. Pijarski, Optymalizacja heurystyczna w ocenie warunków pracy i planowaniu rozwoju systemu elektroenergetycznego, Lublin: Wydawnictwo Politechniki Lubelskiej, 2019.

[70] T. Witkowski i P. Antczak, Algorytmy rojowe w harmonogrowaniu procesów produkcyjnych, Warszawa: Polskie Wydawnictwo Ekonomiczne PWE, 2022.

[71] R. Rani, S. Jain i H. Garg, „A review of nature-inspired algorithms on single-objective Optimisation problems from 2019 to 2023," *Artif Intell Rev,* tom 57, nr 126, 2024.

[72] P. Agrawal, H.F. Abutarboush, T. Ganesh i W.M. Mohamed, „Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009-2019)," *IEEE Access,* tom 9, pp. 26766-26791, 2021.

[73] I. Fister Jr., X.-S. Yang, I. Fister, J. Brest i D. Fister, „A Brief Review of Nature-Inspired Algorithms for Optimisation," *ELEKTROTEHNISKI VESTNIK,* tom 80, nr 3, pp. 1–7, 2013.

[74] A.E. Ezugwu, O.J. Adeleke, A.A. Akinyelu i S. Viriri, „A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems," *Neural Comput & Applic,* tom 32, p. 6207–6251, 2020.

[75] W. McCulloch i W. Pitts, „A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics,* nr 5, pp. 115–133, 1943.

[76] P.J. Braspenning, F. Thuijsman i A.J. M.M. Weijters, Artificial Neural Networks An Introduction to ANN Theory and Practice, Springer, 1995.

[77] L.A. Zadeh, „Fuzzy Sets," *Information Control,* nr 8, pp. 338–353, 1965.

[78] W. Pedrycz, Fuzzy control and fuzzy systems (2nd, extended ed.), Research Studies Press Ltd. GBR, 1993.

[79] L. Davis, Genetic algorithms and simulated annealing, Morgan Kaufman Publisher, 1987.

[80] Z. Michalewicz, Algorytmy genetyczne + struktury danych = programy ewolucyjne, WNT, 1999.

[81] K.H. Mostapha, „Differential Evolution (DE) in MATLAB," Yarpiz, 2015.

[82] S. Das i P.N. Suganthan,, „Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation,* tom 15, nr 1, pp. 4–31, 2011.

[83] R.J. Mahfoud, Y. Sun, N.F. Alkayem i H.H. Alhelou, „A Novel Combined Evolutionary Algorithm for Optimal Planning of Distributed Generators in Radial Distribution Systems," *Applied Sciences,* tom 16, nr 9, 2019.

[84] M.H. Nadimi-Shahraki i H. Zamani, „DMDE: Diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global Optimisation," *Expert Systems with Applications,* tom 198, 2022.

[85] S. Rahnamayan, H.R. Tizhoosh i M.A. Salam, „Opposition-Based Differential Evolution," *EEE Transactions on Evolutionary Computation,* tom 12, nr 1, pp. 64–79, 2008.

[86] H. Dong, J. He, H. Huang i W. Hou, „Evolutionary programming using a mixed mutation strategy," *Information Sciences,* tom 177, nr 1, pp. 312–327, 2007.

[87] H.G. Beyer i H.P. Schwefel, „Evolution strategies – A comprehensive introduction," *Natural Computing,* tom 1, pp. 3–52, 2002.

[88] N. Hansen i A. Ostermeier, „ Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation,* tom 2, nr 9, pp. 159–195, 2001.

[89] J.H. Holland, „Adaptation in Natural and Artificial Systems", Ann Arbor, Mich. USA: Univ. of Michigan Press, 1975.

[90] L. Fogel, A. J. Owens i M.J. Walsh, Artificial Intelligence through Simulated Evolution, New York: Wiley, 1966.

[91] Z. Michalewicz i D. B. Fogel, JAK TO ROZWIĄZAĆ CZYLI NOWOCZESNA HEURYSTYKA, WNT, 2006.

[92] A. Pieprzycki i P. Świętojański, „Heuristic automatic Optimisation algorithms in WLAN networks planning," w *KSTiT 2009 / Przegląd Telekomunikacyjny 8-9/2009*, 2009.

[93] Q.Y. Duan, V. K. Gupta i S. Sorooshian, „Shuffled complex evolution approach for effective and efficient global minimization," *J Optim Theory Appl,* nr 76, pp. 501–521, 1993.

[94] E. Özcan, J.H. Drake i A. Asta, „A self-adaptive multimeme memetic algorithm co- evolving utility scores to control genetic operators and their parameter settings," *Applied Soft Computing,* tom 49, pp. 81–93, 2016.

[95] J. Gálvez, E. Cuevas, S. Hinojosa, O. Avalos i M. Pérez-Cisneros, „A reactive model based on neighborhood consensus for continuous Optimisation," *Expert Systems with Applications,* tom 121, pp. 115–141, 2019.

[96] N.L. Cramer, „A Representation for the Adaptive Generation of Simple Sequential Programs," w *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, 1985.

[97] D. Dickmanns, J. Schmidhuber i A. Winklhofer, „Der genetische Algorithmus: Eine Implementierung in Prolog," Institut fur Informatik, Technische Universit at Manchen, 1987.

[98] J.R. Koza, Genetic Programming II, Automatic discovery of reusable Programs, The MIT Press, 1994.

[99] H. Faris, A.M. Al-Zoubi, A.A. Heidari, I. Aljarah, M. Mafarja, M. Hassonah i H. Fujita, „An Intelligent System for Spam Detection and Identification of the most Relevant Features based on Evolutionary Random Weight Networks," *Information Fusion,* nr 48, pp. 67–83, 2018.

[100] I. Rahman i J. Mohamad-Saleh, „Hybrid bio-Inspired computational intelligence techniques for solving power system Optimisation problems: A comprehensive survey," *Applied Soft Computing,* tom 69, pp. 72–130, 2018.

[101] D. Simon, „Biogeography-Based Optimisation," *IEEE Transactions on Evolutionary Computation,* tom 12, nr 6, pp. 702–713, 2009.

[102] S. Baluja, „Population-based incremental learning. a method for integrating genetic search based function Optimisation and competitive learning," Carnegie Mellon University, Pittsburgh, Pennsylvania, 1994.

[103] O.H.M. Ross, O. Castillo, P. Melin i A. Rodríguez-Díaz, „Human evolutionary model: A new approach to Optimisation," *Information Sciences,* tom 10, nr 177, pp. 2075–2098, 2007.

[104] C. Liu, M. Han i X. Wang, „A novel evolutionary membrane algorithm for global numerical Optimisation," w *Third International Conference on Intelligent Control and Information Processing (ICICIP)*, 2012.

[105] M. Pelikan, „Probabilistic Model-Building Genetic Algorithms," w *Hierarchical Bayesian Optimisation Algorithm. Studies in Fuzziness and Soft Computing*, tom 170, Berlin, Heidelberg, Springer, 2005.

[106] E. Çelik, „Improved stochastic fractal search algorithm and modified cost function for automatic generation control of interconnected electric power systems," *Engineering Applications of Artificial Intelligence,* tom 88, p. nr 103407, 2020.

[107] W. Du, M. Zhang, W. Ying, M. Perc, K. Tang, X. Cao i D. Wu, „The networked evolutionary algorithm: A network science perspective," *Applied Mathematics and Computation,* tom 338, pp. 33–43, 2018.

[108] M.A. Al-Betar, „β-Hill climbing: an exploratory local search," *Neural Computing and Applications,* tom 28 (Suppl. 1), p. 153–168, 2016.

[109] H.R. Lourenço, O.C. Martin i T. Stützle, „Iterated Local Search," w *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, tom 57, Boston, MA, Springer, 2003, pp. 320–353.

[110] N. Mladenović i P. Hansen, „Variable neighborhood search," *Computers & Operations Research,* tom 24, nr 11, pp. 1097–1100, 1997.

[111] N. Mladenović i P. Hansen, „Variable neighborhood search," *Computers & Operations Research,* tom 24, nr 11, pp. 1097–1100, 1997.

[112] T.A. Feo i M.G.C. Resende, „Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimisation,* tom 6, p. 109–133, 1995.

[113] C. Voudouris, E. P. Tsang i A. Alsheddy, „Guided Local Search," tom 146, Boston, MA, Springer, 2010.

[114] C. Audet i J. E. Dennis Jr., „Analysis of Generalized Pattern Searches," *SIAM Journal on Optimisation,* tom 13, nr 3, p. 889–903, 2003.

[115] S. H. Brooks, „A Discussion of Random Methods for Seeking Maxima," *Operations Research,* tom 6, nr 2, pp. 244–251, 1958.

[116] B. Doğan i T. Ölmez, „A new metaheuristic for numerical function Optimisation: Vortex Search algorithm," *Information Sciences,* tom 293, pp. 125–145, 2015.

[117] D. Atzmon, J. Li, A. Felner, E. Nachmani, S. Shperberg, N. Sturtevant i S. Koenig, „Multi-Directional Heuristic Search," w *Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.

[118] R. Marti, Á. Corberán i J. Peiró, „Scatter Search," w *Handbook of Heuristics*, Springer, Cham., 2015, pp. 1–24.

[119] F. Rezaei, H.R. Safavi, M. Abd Elaziz i S. Mirjalili , „GMO: geometric mean optimizer for solving engineering problems," *Soft Computing,* tom 27, p. 10571–10606, 2023.

[120] L. Abualigah, A. Diabat, S. Mirjalili, M.A. Elaziz i A.H. Gandomi, „The Arithmetic Optimisation Algorithm," *Computer Methods in Applied Mechanics and Engineering,* tom 376, nr 2, p. numer: 113609, 2021.

[121] H. Chickermane i H.C. Gea, „STRUCTURAL OPTIMISATION USING A NEW LOCAL APPROXIMATION METHOD," *International Journal for Numerical Methods in Engineering,* tom 39, nr 5, p. 829–846, 1996.

[122] I. Ahmadianfar, A.A. Heidari, S. Noshadian, H. Chen i A.H. Gandomi, „INFO: An efficient Optimisation algorithm based on weighted mean of vectors," *Expert Systems with Applications,* tom 195, 2022.

[123] D. Yang, G. Li i G. Cheng, „On the efficiency of chaos Optimisation algorithms for global Optimisation," *Chaos, Solitons & Fractals,* tom 34, nr 4, pp. 1366–1375, 2007.

[124] C. Grosan i A. Abraham, „A Novel Global Optimisation Technique for High Dimensional Functions," *International Journal of Intelligent Systems, Wiley,* tom 24, nr 4, pp. 421–440, 2009.

[125] I. Ahmadianfar, A.A. Heidari, A.H. Gandomi, X. Chu i H. Chen, „RUN beyond the metaphor: An efficient Optimisation algorithm based on Runge Kutta method," *Expert Systems with Applications,* tom 181, p. nr. 115079, 2021.

[126] E.H. Houssein, M. R. Saad, F.A. Hashim, H. Shaban i M. Hassaballah, „Lévy flight distribution: A new metaheuristic algorithm for solving engineering Optimisation problems," *Engineering Applications of Artificial Intelligence,* tom 94, p. art. 103731, 2020.

[127] S. Mirjalili, „SCA: A Sine Cosine Algorithm for solving Optimisation problems," *Knowledge-Based Systems,* tom 96, 2016.

[128] M.H. Suid,, M.A. Ahmad, M.R.T.R. Ahmad, M.R. Ghazali, A. Irawan i M.Z. Tumari, „An Improved Sine Cosine Algorithm for Solving Optimisation Problems," w *IEEE Conference on Systems, Process and Control (ICSPC)*, Melaka, Malaysia, 2018.

[129] T. Dahiya, N. Vashishth, D. Garg, A. K. Shrivastava i P.K. Kapur, „Novel Heuristic Algorithm & its Application for Reliability Optimisation," *International Journal of Mathematical, Engineering and Management Sciences,* tom 8, nr 4, pp. 755–768, 2023.

[130] J.M. Abdullah i T. Ahmed, „Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process," *IEEE Access,* tom 7, pp. 43473-43486, 2019.

[131] J.M. Abdullah, „Fitness Dependent Optimizer FDO," Mathworks, [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/71129-fitness- dependent-optimizer-fdo. [Data uzyskania dostępu: 03.01.2025].

[132] H.K. Hamarashid, B.A. Hassan i T.A. Rashid, „Modified-Improved Fitness Dependent Optimizer for Complex and Engineering Problems," arxiv, 2022. [Online]. Available: https://arxiv.org/abs/2407.14271.

[133] J.F. Salih, H.M. Mohammed i Z K. Abdul, „Modified Fitness Dependent Optimizer for Solving Numerical Optimisation Functions," *IEEE Access,* tom 10, nr 1, 2022.

[134] O.K. Erol i I. Eksin, „A new Optimisation method: Big Bang–Big Crunch," *Advances in Engineering Software,* tom 37, nr 2, pp. 106–111, 2006.

[135] A. Kaveh i S. Talatahari, „Size Optimisation of space trusses using Big Bang-Big Crunch algorithm," *Computers & Structures,* tom 87, nr 17–18, pp. 1129–1140, 2009.

[136] E. Rashedi, H. Nezamabadi-pour i S. Saryazdi, „GSA: a Gravitational Search Algorithm," *Information Sciences,* tom 179, nr 13, pp. 2232–2248, 2009.

[137] A. Hatamlou, „Black hole: A new heuristic Optimisation approach for data clustering," *Information Sciences,* tom 222, pp. 175–184, 2013.

[138] H. Shah-Hosseini, „Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation," *International Journal of Computational Science and Engineering,* tom 6, nr 1–2, pp. 132–140, 2011.

[139] H. Shah-Hosseini, „Problem solving by intelligent water drops," w *IEEE Congress on Evolutionary Computation*, 2007.

[140] S. Mirjalili, S.M. Mirjalili i A. Hatamlou, „Multi-Verse Optimizer: a nature-inspired algorithm for global Optimisation," *Neural Computing and Applications,* tom 27, nr 2, 2015.

[141] A. Sadollah, A. Bahreininejad, H. Eskandar i M. Hamdi, „Mine blast algorithm: A new population based algorithm for solving constrained engineering Optimisation problems," *Applied Soft Computing,* tom 13, nr 5, pp. 2592–2612, 2013.

[142] A. Ahrari i A.A. Atai, „Grenade Explosion Method—A novel tool for Optimisation of multimodal functions," *Applied Soft Computing,* tom 10, nr 4, pp. 1132–1140, 2010.

[143] A. Kaveh i S. Talatahari, „A novel heuristic Optimisation method: charged system search," *Acta Mechanica,* tom 213, p. 267–289, 2010.

[144] S. Kamel, E.H. Houssein, M.H. Hassan, M. Shouran i F.A. Hashim, „An Efficient Electric Charged Particles Optimisation Algorithm for Numerical

Optimisation and Optimal Estimation of Photovoltaic Models," *Mathematics,* tom 10, nr 6, 2022.

[145] D. Oliva, E. Cuevas, G. Pajares, Z. Zaldivar i V. Osuna, „A Multilevel Thresholding algorithm using electromagnetism Optimisation," *Neurocomputing,* tom 139, pp. 357–381, 2014.

[146] A. Kaveh i M. Khayatazad, „A new meta-heuristic method: Ray Optimisation," *Computers & Structures,* tom 112–113, p. 283–294, 2012.

[147] M. Abdechiri, M.R. Meybodi i H. Bahrami, „Gases Brownian Motion Optimisation: an Algorithm for Optimisation (GBMO)," *Applied Soft Computing,* tom 13, nr 5, p. 2932–2946, 2013.

[148] J. Cheng i W. Zhao, „Chaotic enhanced colliding bodies Optimisation algorithm for structural reliability analysis," *Advances in Structural Engineering,* tom 23, nr 3, pp. 438–453, 2020.

[149] H. Shareef, A.A. Ibrahim i A.H. Mutlag, „Lightning search algorithm," *Applied Soft Computing,* tom 36, pp. 315–333, 2015.

[150] A. Kaveh i T. Bakhshpoori, „Water Evaporation Optimisation: A novel physically inspired Optimisation algorithm," *Computers & Structures,* tom 167, pp. 69–85, 2016.

[151] A. Kaveh i A. Dadras, „A novel meta-heuristic Optimisation algorithm: Thermal exchange Optimisation," *Advances in Engineering Software,* tom 110, pp. 69–84, 2017.

[152] R.A. Formato, „Central force Optimisation: A new metaheuristic with applications in applied electromagnetics," *Progress In Electromagnetics Research,* tom 77, nr 1, pp. 425–491, 2007.

[153] H. Abedinpourshotorban, S. M. Shamsuddin, Z. Beheshti i D. N. A. Jawawi, „Electromagnetic field Optimisation: A physics-inspired metaheuristic Optimisation algorithm," *Swarm and Evolutionary Computation,* tom 26, pp. 8–22, 2016.

[154] Y. Chung, I. Char, W. Neiswanger, K. Kandasamy, A. O. Nelson, M. D. Boyer, E. Kolemen i J. Schneider, „Offline Contextual Bayesian Optimisation for Nuclear Fusion," arXiv:2001.01793, 2020.

[155] A.G. Hussien, A. Pop, S. Kumar, F.A. Hashim i G. Hu, „A Novel Artificial Electric Field Algorithm for Solving Global Optimisation and Real-World Engineering Problems," *Biomimetics,* tom 9, nr 3, p. nr. 186, 2024.

[156] S.I. Birbil i S.C. Fang, „An Electromagnetism-like Mechanism for Global Optimisation," *Journal of Global Optimisation,* tom 25, p. 263–282, 2003.

[157] E. Cuevas, A. Echavarría i M.A. Ramirez-Ortegon, „An Optimisation algorithm inspired by the States of Matter that improves the balance between exploration and exploitation," *Applied Intelligence,* tom 40, p. 256–272, 2014.

[158] K. Tamura i K. Yasuda, „Primary Study of Spiral Dynamics Inspired," *IEEJ Transactions on Electrical and Electronic Engineering,* tom 6, pp. 98-100, 2011.

[159] K. Tamura i K. Yasuda, „Spiral Dynamics Inspired Optimisation," *Journal of Advanced Computational Intelligence and Intelligent Informatics,* tom 15, nr 8, pp. 1116–1122, 2011.

[160] A.H. Kashan, „A new metaheuristic for Optimisation: Optics inspired Optimisation (OIO)," *Computers & Operations Research,* tom 55, pp. 99–125, 2015.

[161] W. Zhao, L. Wang i Z. Zhang, „Atom search Optimisation and its application to solve a hydrogeologic parameter estimation problem," *Knowledge-Based Systems,* tom 163, pp. 283–304, 2019.

[162] W. F. Sacco, D. C. Knupp, E. F. Pacheco da Luz, J. C. Becceneri, H. F. de Campos Velho i A. da Silva Neto, „Particle Collision Algorithm," w *Computational Intelligence Applied to Inverse Problems in Radiative Transfer*, Springer, Cham., 2023.

[163] P. Aungkulanon, P. Luangpaiboon i R. Montemanni, „An elevator kinematics optimisation method for aggregate production planning based on fuzzy MOLP model," *International Journal of Mechanical Engineering and Robotics Research,* tom 7, nr 4, 2018.

[164] Y.-T. Hsiao, C.-L. Chuang, J.-A. Jiang i C.-C. Chien, „A novel Optimisation algorithm: Space gravitational Optimisation," w *IEEE International Conference on Systems, Man and Cybernetics*, Waikoloa, Hawaii, 2005.

[165] H. Eskandar, A. Sadollah, A. Bahreininejad i M. Hamdi, „Water cycle algorithm – A novel metaheuristic Optimisation method for solving constrained engineering Optimisation problems," *Computers & Structures,* Tomy %1 z %2110-111, pp. 151–166, 2012.

[166] A.A. Heidari, R.A. Abbaspour i A.R. Jordehi, „An efficient chaotic water cycle algorithm for Optimisation tasks," *Neural Computing and Applications,* tom 28, nr 1, pp. 57–85, 2015.

[167] W. Zhang i J. Zhang, „Improved Water Cycle Algorithm with Chaotic Levy Distribution Applied to Engineering Problem," w *International Conference on Electronical, Mechanical and Materials Engineering (ICE2ME 2019)*, 2019.

[168] T.R. Biyanto, Matradji, M.N. Syamsi, H.Y. Fibrianto, N. Afdanny, A.H. Rahman, K.S. Gunawan, J.A.D. Pratama, A. Malwindasari, A.I. Abdillah, T.N. Bethiana i Y.A. Putra, „Optimisation of Energy Efficiency and Conservation in Green Building Design Using Duelist, Killer-Whale and Rain-Water Algorithms," *IOP Conference Series: Materials Science and Engineering,* tom 267, nr 1, pp. 1–8, 2017.

[169] A. Wedyan, J.L. Whalley i A. Narayanan, „Hydrological Cycle Algorithm for Continuous Optimisation Problems," *Journal of Optimisation,* nr 10, pp. 1–25, 2017.

[170] P. Rabanal, I. Rodríguez i F. Rubio, „Using River Formation Dynamics to Design Heuristic Algorithms," w *Proceedings Unconventional Computation, 6th International Conference, UC 2007*, Kingston, Canada, 2007.

[171] A. Mahdi, „Atomic Orbital Search: A Novel Metaheuristic Algorithm," *Applied Mathematical Modelling,* tom 93, pp. 657–683, 2021.

[172] P. Thakral i Y. Kumar, „An Improved Water Flow Optimizer for Data Clustering," *SN Computer Science,* tom 5, nr 6, p. nr. 715, 2024.

[173] Q. Jiang, L. Wang i X. Hei, „Parameter identification of chaotic systems using artificial raindrop algorithm," *Journal of Computational Science,* tom 8, pp. 20–31, 2015.

[174] A. R. Moazzeni i E. Khamehchi, „Rain Optimisation algorithm (ROA): A new metaheuristic method for drilling Optimisation solutions," *Journal of Petroleum Science and Engineering,* tom 195, 2020.

[175] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen i O. Shochet, „Novel Type of Phase Transition in a System of Self-Driven Particles," *Physical Review Letters,* tom 75, nr 6, p. 1226–1229, 1995.

[176] J.M. Bishop, „Stochastic Searching Networks," w *Artificial Neural Networks, 1989., First IEE International Conference on (Conf Publ. No. 313)*, 1989.

[177] M. Masoomeh i H. Naderpour, „Transit Search Optimisation Algorithm," Matlab File Exchange, 2022.

[178] R. Irizarry, „LARES: An Artificial Chemical Process Approach for Optimisation," *Evolutionary Computation,* tom 12, nr 4, pp. 435–459, 2004.

[179] B. Alatas, „A novel chemistry based metaheuristic Optimisation method for mining of classification rules," *Expert Systems with Applications,* tom 39, pp. 11080–11088, 2012.

[180] A. Lam i V.O.K. Li, „Chemical-Reaction-Inspired Metaheuristic for Optimisation," *IEEE Transactions on Evolutionary Computation,* tom 14, nr 3, pp. 381–399, 2010.

[181] N. Siddique i H. Adeli, „Nature Inspired Chemical Reaction Optimisation Algorithms," *Cognitive Computation,* tom 9, nr 4, p. 411–422, 2017.

[182] M. H. Salmani i K. Eshghi, „A Metaheuristic Algorithm Based on Chemotherapy Science: CSA," *Journal of Optimisation,* p. nr. 3082024, 2017.

[183] X. S. Yang, Nature-Inspired Metaheuristic Algorithms, 2nd Edition,, Luniver Press, 2010.

[184] F.A. Hashim, E.H. Houssein, M. S. Mabrouk, W. Al-Atabany i S. Mirjalili, „Henry gas solubility Optimisation: A novel physics-based algorithm," *Future Generation Computer Systems,* tom 101, pp. 646–667, 2019.

[185] F. Glover, „Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers & Operations Research,* tom 13, nr 5, pp. 533–549, 1986.

[186] S.-u.-R. Massan, A I. Wagan i M.M. Shaikh, „A new metaheuristic Optimisation algorithm inspired by human dynasties with an application to the wind turbine micrositing problem," *Applied Soft Computing,* tom 90, p. nr 106176, 2020.

[187] Y. Shi, „Brain Storm Optimisation Algorithm," w *ICSI 2011 International conference in swarm intelligence - Lecture Notes in Computer Science*, New York, USA, 2011.

[188] R.V. Rao, V. J. Savsani i D. P. Vakharia, „Teaching–learning-based Optimisation: A novel method for constrained mechanical design Optimisation problems," *Computer – Aided Design,* tom 43, nr 3, pp. 303–315, 2011.

[189] R.V. Rao i V. Patel, „An elitist teaching-learning-based Optimisation algorithm for solving complex constrained Optimisation problems," *International Journal of Industrial Engineering Computations,* tom 3, nr 4, pp. 535–560, 2012.

[190] R.V. Rao i V. Patel, „Multi-objective Optimisation of heat exchangers using a modified teaching-learning-based Optimisation algorithm," *Applied Mathematical Modelling,* tom 37, nr 3, pp. 1147–1162, 2013.

[191] M.J. Mahmoodabadi i R. Ostadzadeh, „CTLBO: Converged teaching–learning– based Optimisation," *Cogent Engineering,* tom 6, nr 1, 2019.

[192] R.V. Rao, „Jaya: A simple and new Optimisation algorithm for solving constrained and unconstrained Optimisation problems," *International Journal of Industrial Engineering Computations,* tom 7, nr 1, pp. 19–34, 2016.

[193] D. Karaboga i B. Akay, „Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimisation," w *IPROMS 2009 Innovative Production Machines and Systems Virtual Conference*, Cardiff,UK, 2009.

[194] P. Sabarinath, M. R. Thansekhar i R. Saravanan, „Multiobjective Optimisation Method Based on Adaptive Parameter Harmony Search Algorithm," *Journal of Applied Mathematics,* tom 2015, nr 2, 2015.

[195] Z.W. Geem, J. H. Kim i G. V. Loganathan, „A new heuristic Optimisation algorithm: harmony search," *Simulation,* tom 76, nr 2, pp. 60–68, 2001.

[196] M.K. Heris, „Harmony Search in MATLAB," Yarpiz, 2015.

[197] K. Pandiarajan i C. K. Babulal, „Fuzzy harmony search algorithm based optimal power flow for power system security enhancement," *International Journal of Electrical Power & Energy Systems,* tom 78, pp. 72–79, 2016.

[198] Y. Tan i Y. Zhu, „Fireworks Algorithm for Optimisation," w *Advances in Swarm Intelligence, First International Conference, ICSI 2010*, Beijing, China, 2010.

[199] H. Shayanfar i F.S. Gharehchopogh, „Farmland fertility: A new metaheuristic algorithm for solving continuous Optimisation problems," *Applied Soft Computing,* tom 71, pp. 728–746, 2018.

[200] A.M. Fathollahi-Fard, M. Hajiaghaei-Keshteli i R. Tavakkoli-Moghaddam, „The Social Engineering Optimizer (SEO)," *Engineering Applications of Artificial Intelligence,* tom 72, pp. 267–293, 2018.

[201] A. Tharwat i T. Gabel, „Parameters Optimisation of support vector machines for imbalanced data using social ski driver algorithm," *Neural Computing and Applications,* tom 32, nr 11, pp. 6925–6938, 2019.

[202] A. Tharwat, „Social Ski-Driver (SSD) Optimisation algorithm," MathWorks, 2019.

[203] M. Kumar, A.J. Kulkarni i S. Satapathy, „Socio evolution & learning Optimisation algorithm: A socio-inspired Optimisation methodology," *Future Generation Computer Systems,* tom 81, p. 52–272, 2017.

[204] M. Li, H. Zhao, X. Weng i T. Han, „Cognitive behavior Optimisation algorithm for solving Optimisation problems," *Applied Soft Computing,* tom 39, pp. 199-222, 2016.

[205] J. Zhang, M. Xiao, L. Gao i Q. Pan, „Queuing search algorithm: A novel metaheuristic algorithm for solving engineering Optimisation problems," *Applied Mathematical Modelling,* tom 63, pp. 464–490, 2018.

[206] C. Dai, Y. Zhu i W. Chen, „Seeker Optimisation Algorithm," w *Computational Intelligence and Security. CIS 2006. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2007.

[207] E. Atashpaz-Gargari i C. Lucas, „Imperialist Competitive Algorithm: An Algorithm for Optimisation Inspired by Imperialistic Competition," w *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*, Singapore, 2007.

[208] S. Hosseini i A. Al Khaled, „A survey on the Imperialist Competitive Algorithm metaheuristic: Implementation in engineering domain and directions for future research," *Applied Soft Computing,* tom 24, pp. 1078–1094, 2014.

[209] E. Bernal, O. Castillo, J. Soria i F. Valdez, „Imperialist Competitive Algorithm with Dynamic Parameter Adaptation Using Fuzzy Logic Applied to the Optimisation of Mathematical Functions," *Algorithms,* tom 10, nr 1, 2017.

[210] T.R. Biyanto, H.Y. Fibrianto, G. Nugroho, A.M. Hatta, E. Listijorini, T. Budiati i H. Huda, „Duelist Algorithm: An Algorithm Inspired by How Duelist Improve Their Capabilities in a Duel," w *Advances in Swarm Intelligence 7th International Conference, ICSI 2016*, Bali, Indonesia, 2016.

[211] T. Ray i K.M. Liew, „Society and civilization: An Optimisation algorithm based on the simulation of social behavior," *IEEE Transactions on Evolutionary Computation,* tom 7, nr 4, pp. 386–396, 2003.

[212] R. Reynolds, „An Introduction to Cultural Algorithms," w *3rd Annual Conference on Evolutionary Programming*, 1994.

[213] M.K. Heris, „Cultural Algorithm (CA) in MATLAB," Yarpiz, 2015.

[214] T. Chen, W. Guo i Z. Gao, „Artificial Searching Swarm Algorithm and Its Performance Analysis," *Applied Mathematics,* tom 3, nr 10A, 2012.

[215] X.-F. Xie, W.-J. Zhang i Z.-L. Yang, „Social cognitive Optimisation for nonlinear programming problems," w *International Conference on Machine Learning and Cybernetics (ICMLC)*, Beijing, China, 2002.

[216] C. Tang, Y. Zhou, Q. Luo i Z. Tang, „An enhanced pathfinder algorithm for engineering Optimisation problems," *Engineering with Computers,* tom 38, nr sup. 2, p. 1481–1503, 2022.

[217] A. H. Gandomi, „Interior search algorithm (ISA): A novel approach for global Optimisation," *ISA Transactions,* tom 53, nr 4, pp. 1168–1183, 2014.

[218] V. Punnathanam i P. Kotecha, „Yin-Yang-pair Optimisation: A novel lightweight Optimisation algorithm," *Engineering Applications of Artificial Intelligence,* tom 54, pp. 62–79, 2016.

[219] N.S. Jaddi i S. Abdullah, „Kidney-inspired algorithm with reduced functionality treatment for classification and time series prediction," *PLOS ONE,* tom 14, nr 1, pp. 1–17, 2019.

[220] P. Moscato, „On Evolution, Search, Optimisation, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," California Institute of Technology, Pasadena, 1989.

[221] X. Chen, Y. S. Ong, M. H. Lim i K. C. Tan., „A Multi-Facet Survey on Memetic Computation," *IEEE Transactions on Evolutionary Computation,* tom 5, nr 15, pp. 591–607, 2011.

[222] Y.S. Ong, M.-H. Lim i X. Chen, „Memetic Computation—Past, Present & Future [Research Frontier]," *IEEE Computational Intelligence Magazine,* tom 5, nr 2, pp. 24–31, 2010.

[223] J. D. Farmer, N.H. Packard i A. S. Perelson, „The immune system, adaptation, and machine learning," *Physica D: Nonlinear Phenomena,* tom 22, nr 1-3, pp. 187–204, 1986.

[224] H. Bersini i F.J. Varela, „Hints for adaptive problem solving gleaned from immune networks," w *Parallel Problem Solving from Nature*, Dortmund, FRG, 1990.

[225] D. Dasgupta, Artificial Immune Systems and Their Applications, Berlin: Springer-Verlag, Inc., 1998.

[226] M.A. Al-Betar, Z.A.A. Alyasseri, M. A. Awadallah i I. A. Doush , „Coronavirus herd immunity optimizer (CHIO)," *Neural Computing and Applications,* tom 33, p. 5011–5042, 2020.

[227] S. Asian i S. Demirci, „Immune Plasma Algorithm: A Novel Meta-Heuristic for Optimisation Problems," *IEEE Access,* nr 8, pp. 220227–220245, 2020.

[228] K.J. Gaston, J. Bennie, T.W. Davies i J. Hopkins, „The ecological impacts of nighttime light pollution: a mechanistic appraisal," *Biological Reviews,* tom 88, p. 912–927, 2013.

[229] L. Abualigah, „Group search optimizer: a nature-inspired meta-heuristic Optimisation algorithm with its results, variants, and applications," *Neural Computing and Applications,* tom 33, nr 7, 2020.

[230] R. Oftadeh, M.J. Mahjoob i M. Shariatpanahi, „A novel meta-heuristic Optimisation algorithm inspired by group hunting of animals: Hunting search," *Computers & Mathematics with Applications,* tom 60, nr 7, pp. 2087–2098, 2010.

[231] E. Fadakar i M. Ebrahimi, „A new metaheuristic football game inspired algorithm," w *1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, Bam, Iran, 2016.

[232] A.H. Kashan, „League Championship Algorithm (LCA): An algorithm for global Optimisation inspired by sport championships," *Applied Soft Computing,* tom 6, pp. 171–200, 2014.

[233] N. Moosavian i B. Kasaee Roodsari, „Soccer League Competition Algorithm: A Novel Meta-heuristic Algorithm For Optimal Design of Water Distribution Networks," *Swarm and Evolutionary Computation,* tom 17, nr 4, pp. 14–24, 2014.

[234] M. . Al-Betar, M.A. Awadallah, M.S. Braik, S. Makhadmeh i I. A. Doush, „Elk herd optimizer: a novel nature-inspired metaheuristic algorithm," *rtificial Intelligence Review,* tom 57, p. nr. 48, 2024.

[235] X.-S. Yang, „A New Metaheuristic Bat-Inspired Algorithm," w *Nature Inspired Cooperative Strategies for Optimisation (NICSO 2010), Studies in Computational Intelligence*, tom 284, Berlin, Heidelberg, Springer, 2010, pp. 65–74.

[236] X.-S. Yang, „Bat algorithm (demo)," 2024.

[237] A. O. Topal i O. Altun, „A novel meta-heuristic algorithm: Dynamic Virtual Bats Algorithm," *Information Sciences,* tom 354, pp. 222–235, 2016.

[238] A. H. Gandomi i X.-S. Yang, „Chaotic bat algorithm," *Journal of Computational Science,* tom 5, nr 2, pp. 224-232, 2014.

[239] X. Cai, X.-z. Gao i Y. Xue, „Improved bat algorithm with optimal forage strategy and random disturbance strategy," *International Journal of Bio-Inspired Computation (IJBIC),* tom 8, nr 4, 2016.

[240] S. Fong, S. Deb i X.-S. Yang, „A heuristic Optimisation method inspired by wolf preying behavior," *A heuristic Optimisation method inspired by wolf preying behavior,* tom 26, p. 1725–1738, 2015.

[241] S. Mirjalili, S.M. Mirjalili i A. Lewis, „Grey Wolf Optimizer," *Advances in Engineering Software,* tom 69, pp. 46–61, 2014.

[242] H. Rashaideh , A. Sawaie, M. Al-Betar, L. Abualigah, M.M. Al-Laham, R.M. Al- Khatib i M. Braik, „A Grey Wolf Optimizer for Text Document Clustering," *Journal of Intelligent Systems,* tom 29, nr 1, 2018.

[243] L. Li, L. Sun, J. Guo, J. Qi, B. Xu i S. Li, „Modified Discrete Grey Wolf Optimizer Algorithm for Multilevel Image Thresholding," *Computational Intelligence and Neuroscience,* 2017.

[244] M. Kohli i S. Arora, „Chaotic grey wolf Optimisation algorithm for constrained Optimisation problems," *Journal of Computational Design and Engineering,* tom 5, nr 4, pp. 458–472, 2018.

[245] S. Mirjalili i A. Lewis, „The Whale Optimisation Algorithm," *Advances in Engineering Software,* tom 95, pp. 51–67, 2016.

[246] F. S. Gharehchopogh i H. Gholizadeh, „A comprehensive survey: Whale Optimisation Algorithm and its applications," *Swarm and Evolutionary Computation,* tom 48, pp. 1–24, 2019.

[247] Y. Zhou, Y. Ling i Q. Luo, „Lévy Flight Trajectory-Based Whale Optimisation Algorithm for Global Optimisation," *IEEE Access,* tom 99, 2017.

[248] Y. Zhou, Y. Ling i Q. Luo, „Lévy flight trajectory-based whale Optimisation algorithm for engineering Optimisation," *Engineering Computations,* tom 35, nr 7, pp. 2406–2428, 2018.

[249] F. Hemasian-Etefagh i F. Safi-Esfahani, „Group-based whale Optimisation algorithm," *Soft Computing,* tom 24, nr 5, pp. 3647–3673, 2020.

[250] G. Kaur i S. Arora, „Chaotic whale Optimisation algorithm," *Journal of Computational Design and Engineering,* tom 5, nr 3, pp. 275–284, 2018.

[251] G. Dhiman i V. Kumar, „Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications," *Advances in Engineering Software,* tom 114, pp. 48–70, 2017.

[252] X. Chen, F. Cheng, L. Liu i L. Cheng, „An improved Wolf pack algorithm for Optimisation problems: Design and evaluation," *PLOS ONE,* tom 16, nr 8, 2021.

[253] M. Jain, V. Singh i A. Rani, „A novel nature-inspired algorithm for Optimisation: Squirrel search algorithm," *Swarm and Evolutionary Computation,* tom 44, pp. 148–175, 2019.

[254] A. Kaveh i N. Farhoudi, „A new Optimisation method: Dolphin echolocation," *Advances in Engineering Software,* tom 59, pp. 53–70, 2013.

[255] A. Seyyedabbasi i F. Kiani, „Sand Cat swarm Optimisation: a nature-inspired algorithm to solve global Optimisation problems," *Engineering with Computers,* tom 39, p. 2627–2651, 2023.

[256] H. Jia, J. Zhang, H. Rao i L. Abualigah, „Improved sandcat swarm Optimisation algorithm for solving global optimum problems," *Artif Intell Rev,* tom 58, nr 5, 2025.

[257] S.-C. Chu, P.-w. Tsai i J.-S. Pan, „Cat Swarm Optimisation," w *PRICAI Pacific Rim International Conference on Artificial Intelligence 2006: Trends in Artificial Intelligence Lecture Notes in Computer Science (LNAI)*, 2006.

[258] M. Bahrami, O. Bozorg-Haddad i X. Chu, „Cat Swarm Optimisation (CSO) Algorithm," *Advanced Optimisation by Nature-Inspired Algorithms. Studies in Computational Intelligence,* tom 720, pp. 9–18, 2018.

[259] K. Selvakumar, K. Vijayakumar i C. S. Boopathi, „CSO Based Solution for Load Kickback Effect in Deregulated Power Systems," *Applied Science,* tom 7, nr 11, 2017.

[260] R.R. Ihsan, S. M. Almufti, B. M. S. Ormani3, R. R. Asaad i R. B. Marqas, „A Survey on Cat Swarm Optimisation Algorithm," *Asian Journal of Research in Computer Science,* tom 10, nr 2, pp. 22–32, 2021.

[261] F.A. Hashim, E.H. Houssein, K. Hussain, M.S. Mabrouk i W. Al-Atabany, „Honey Badger Algorithm: New metaheuristic algorithm for solving Optimisation problems," *Mathematics and Computers in Simulation,* tom 192, pp. 84–110, 2022.

[262] C. Zhong, G. Li i L. Z. Meng, „Beluga whale Optimisation: A novel nature-inspired metaheuristic algorithm," *Knowledge-Based Systems,* tom 251, 2022.

[263] Z. Meng i J.-S. Pan, „Monkey King Evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption Optimisation," *Knowledge- Based Systems,* tom 97, pp. 144–157, 2016.

[264] Y. Jiang, Q. Wu, S. Zhu i L. Zhang, „Orca predation algorithm: A novel bio-inspired algorithm for global Optimisation problems," *Expert Systems with Applications,* tom 188, p. nr. 116026, 2022.

[265] T. R. Biyanto, Matradji, S. Irawan, H. Y. Febrianto, N. Afdanny, A. H. Rahman, K. S. Gunawan, J. A. D. Pratama i T. N. Bethiana, „Killer Whale Algorithm: An Algorithm Inspired by the Life of Killer Whale," w *4th Information Systems International Conference 2017, ISICO 2017, 6-8 November 2017/Procedia Computer Science*, Bali, Indonesia, 2017.

[266] C.E. Klein, V.C. Mariani i L. d. S. Coelho, „heetah Based Optimisation Algorithm: A Novel Swarm Intelligence Paradigm," w *ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence*, Bruges (Belgium), 2018.

[267] M.A. Akbari, M. Zare, R. Azizipanah-abarghooee, S. Mirjalili i M. Deriche, „The cheetah optimizer: a nature-inspired metaheuristic algorithm for large-scale Optimisation problems," *Scientific Report,* tom 12, p. nr. 10953, 2022.

[268] J.C. Bansal, H. Sharma, S. S. Jadon i M. Clerc, „Spider Monkey Optimisation algorithm for numerical Optimisation," *Memetic Computing,* tom 6, p. 31–47, 2014.

[269] J. Mumtaz, Z. Guan, L. Yue, L. Zhang i C. He, „Hybrid spider monkey optimisation algorithm for multi-level planning and scheduling problems of assembly lines," *Interntional Journal of production Research,* tom 58, nr 1, pp. 1–16, 2019.

[270] J. Pierezan i D.S. Coelho, „Coyote Optimisation Algorithm: A New Metaheuristic for Global Optimisation Problems," w *IEEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Brazil, 2018.

[271] G.-G. Wang, S. Deb i L. d. S. Coelho, „Elephant Herding Optimisation," w *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, Bali, Indonesia, 2016.

[272] R. Masadeh, B. A. Mahafzah i A. Sharieh, „Sea Lion Optimisation Algorithm," w *(IJACSA) International Journal of Advanced Computer Science and Applications*, 2019.

[273] R. Masadeh, A. Sharieh i B.A. Mahafzah, „Humpback Whale Optimisation Algorithm Based on Vocal Behavior for Task Scheduling in Cloud Computing," *International Journal of Advanced Science and Technology,* tom 13, nr 3, 2019.

[274] O. J. Agushaka, A. E.-S. Ezugwu i L. Abualigah, „Dwarf Mongoose Optimisation Algorithm," *Computer Methods in Applied Mechanics and Engineering,* tom 391, nr 10, 2022.

[275] J. Kennedy i R. Eberhart, „Particle Swarm Optimisation," w *IEEE International Conference on Neural Networks*, 1995.

[276] R. Eberhart, Y. Shi i J. Kennedy, Swarm Intelligence, San Francisco: Morgan Kaufman, 2001.

[277] Y. Sun, G. Qi, Z. Wang, B. J. van Wyk i Y. Hamam, „Chaotic particle swarm Optimisation," w *GEC ,09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, 2009.

[278] M. Zellagui, A. Lasmari, A. Alaboudy, S. Settoul i H. A. Hassan, „Enhancing energy efficiency for optimal multiple photovoltaic distributed generators integration using inertia weight control strategies in PSO algorithms," *POLITYKA ENERGETYCZNA – ENERGY POLICY JOURNAL,* tom 25, nr 1, pp. 59–88, 2022.

[279] Z. Ma, X. Yuan, S. Han, D. Sun i Y. Ma, „Improved Chaotic Particle Swarm Optimisation Algorithm with More Symmetric Distribution for Numerical Function Optimisation," *Symmetry,* tom 11, nr 7, 2019.

[280] W. Zhao, L. Wang i S. Mirjalili, „Artificial hummingbird algorithm: A new bio- inspired optimizer with its engineering applications," *Computer Methods in Applied Mechanics and Engineering,* tom 388, p. nr. 114194, 2022.

[281] J. Tu, H. Chen, M. Wang i A. H. Gandomi, „The Colony Predation Algorithm," *Journal of Bionic Engineering,* tom 18, p. 674–710, 2021.

[282] X. Yang, „Cuckoo Search (CS) Algorithm – File Exchange – MATLAB Central," [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm. [Data uzyskania dostępu: 17.09.2024].

[283] X.-S. Yang i S. Deb, „Cuckoo search via Lévy flights," w *World Congress on Nature & Biologically Inspired Computing (NaBIC 2009) IEEE Publications*, 2009.

[284] R. Rajabioun, „Cuckoo Optimisation Algorithm," *Applied Soft Computing,* tom 11, nr 8, pp. 5508–5518, 2011.

[285] S. Walton, O. Hassan, K. Morgan i M. R. Brown, „Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons & Fractals,* tom 44, nr 9, pp. 710–718, 2011.

[286] A.M. Kamoona, J.C. Patra i A. Stojcevski, „An Enhanced Cuckoo Search Algorithm for Solving Optimisation Problems," w *EEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Brazil, 2018.

[287] X.-S. Yang i S. Deb, „Eagle Strategy Using Lévy Walk and Firefly Algorithms for Stochastic Optimisation," *Nature Inspired Cooperative Strategies for Optimisation (NICSO 2010). Studies in Computational Intelligence,* 2010.

[288] A. Askarzadeh, „Bird mating optimizer: An Optimisation algorithm inspired by bird mating strategies," *Communications in Nonlinear Science and Numerical Simulation,* tom 19, nr 4, pp. 1213–1228, 2014.

[289] A. Arram, M. Ayob, G. Kendall i A. Sulaiman, „Bird Mating Optimizer for Combinatorial Optimisation Problems," *IEEE Access,* tom 8, pp. 96845–96858, 2020.

[290] Z. Chen, A. Francis, S. Li, B. Liao, D. Xiao, T. T. Ha, J. Li, L. Ding i X. Cao, „Egret Swarm Optimisation Algorithm: An Evolutionary Computation Approach for Model Free Optimisation," *Biomimetics,* tom 7, nr 4, p. nr. 144, 2022.

[291] D. Binu i B. S. Kariyappa, „RideNN: A new rider Optimisation algorithm based neural network for fault diagnosis of analog circuits," *EEE Transactions on Instrumentation & Measurement,* tom 68, nr 1, pp. 2–26, 2019.

[292] H. Zamani, M.H. Nadimi-Shahraki i A.H. Gandomi, „Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering Optimisation," *Computer Methods in Applied Mechanics and Engineering,* tom 392, p. nr. 114616, 2022.

[293] X.-B. Meng, X. Z. Gao, L. Lu, Y. Liu i H. Zhang, „A new bio-inspired optimisation algorithm: Bird Swarm Algorithm," *Journal of Experimental & Theoretical Artificial Intelligence,* tom 28, nr 4, pp. 673–687, 2016.

[294] A. Askarzadeh, „A novel metaheuristic method for solving constrained engineering Optimisation problems: Crow search algorithm," *Computers & Structures,* tom 169, pp. 1–12, 2016.

[295] K. Sridhar, C. Kavitha, W.-C. Lai i B.P. Kavin, „Detection of Liver Tumour Using Deep Learning BasedSegmentation with Coot Extreme Learning Model," *Biomedicines,* tom 11, nr 3, 2023.

[296] R.R. Mostafa, A.G. Hussien, M.A. Khan, S. Kadry i F.A. Hashim, „Enhanced coot Optimisation algorithm for dimensionalityreduction," w *2022 Fifth International Conference of Women in Data Science at Prince Sultan University (WiDSPSU)*, Riyadh, Saudi Arabia, 2022.

[297] I. Naruei i F. Keynia, „A new Optimisation method based on COOT bird natural life model," tom 183, 2021. ???

[298] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja i H. Chen, „Harris hawks Optimisation: Algorithm and applications," *Future Generation Computer Systems,* tom 97, pp. 849–872, 2019.

[299] B. Abdollahzadeh, F. S. Gharehchopogh i S. Mirjalili, „African vultures Optimisation algorithm: A new nature-inspired metaheuristic algorithm for global Optimisation problems," *Computers & Industrial Engineering,* tom 158, 2021.

[300] J. Xue i B. Shen, „A novel swarm intelligence Optimisation approach: sparrow search algorithm," *Systems Science & Control Engineering,* tom 8, nr 1, pp. 22–34, 2020.

[301] C. Zhang i S. Ding, „A stochastic configuration network based on chaotic sparrow search algorithm," *Knowledge-Based Systems,* tom 220, p. nr. 106924, 2020.

[302] H.A. Alsattar, A.A. Zaidan i B.B. Zaidan, „Novel meta-heuristic bald eagle search optimisation algorithm," *Artificial Intelligence Review,* tom 53, p. 2237–2264, 2019.

[303] M.A. Hasan, „Metaheuristic: Bald Eagle Search," Babylon University , 2020.

[304] A. Chhabra, A.G. Hussien i F.A. Hashim, „Improved bald eagle search algorithm for global Optimisation and feature selection," *Alexandria Engineering Journal,* tom 68, pp. 141–180, 2023.

[305] M. Huang, „Machine Learning Based Analysis and Prediction of Emotional Expressions in Dance Movements," *Applied Mathematics and Nonlinear Sciences,* tom 9, nr 1, pp. 1–22, 2024.

[306] R.K. Hamad i T.A. Rashid, „GOOSE algorithm: a powerful Optimisation tool for real-world engineering challenges and beyond," *Evolving Systems,* tom 15, p. 1249–1274, 2024.

[307] M. Zhang i G. Wen, „Duck swarm algorithm: theory, numerical Optimisation, and applications," *Cluster Computing,* tom 27, p. 6441–6469, 2024.

[308] O.W. Khalid, N.A.M. Isa i H.A.M. Sakim, „Emperor penguin optimizer: A comprehensive review based on state-of-the-art meta-heuristic algorithms," *Alexandria Engineering Journal,* tom 63, pp. 487–526, 2023.

[309] S.H.S. Moosavi i V.K. Bardsiri, „Satin bowerbird optimizer: A new Optimisation algorithm to optimize ANFIS for software development effort estimation," *Engineering Applications of Artificial Intelligence,* tom 60, pp. 1–15, 2017.

[310] V.K. Bardsiri, „Satin Bowerbird Optimizer (SBO 2017)," MATLAB Central File Exchange MathWorks.

[311] F.A. Hashim i A.G. Hussien, „Snake Optimizer: A novel meta-heuristic Optimisation algorithm," *Knowledge-Based Systems,* tom 242, p. nr. 108320, 2022.

[312] M.S. Braik, „Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems," *Expert Systems with Applications,* tom 74, 2021.

[313] S. Suyanto, „Komodo Mlipir Algorithm," MATLAB Central File Exchange, 2024.

[314] M.M. Eusuff i K.E. Lansey, „Optimisation of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm," *Journal of Water Resources Planning and Management,* tom 129, nr 3, pp. 210–225, 2003.

[315] M.K.Heris, „Shuffled Frog Leaping Algorithm in MATLAB," Yarpiz, 2015.

[316] X.L.Li, Z. J. Shao i J. X. Qian, „ Optimizing method based on autonomous animats: fish-swarm algorithm," *System Engineering Theory and Practice,* tom 22, nr 11, pp. 32–38, 2002.

[317] L. Chen, „SwarmFish- The Artificial Fish Swarm Algorithm," Matlab Cental File Exchange MathWorks, 2024.

[318] F. Pourpanah, R. Wang, C.P. Lim i X.-Z. Wang, „A Review of Artificial Fish Swarm Algorithms: Recent Advances and Applications," *Artificial Intelligence Review,* tom 56, nr 6, 2022.

[319] Z. Meng, J.S. Pan i A. Alelaiwi, „A new meta-heuristic ebb-tide-fish-inspired algorithm for traffic navigation," *Telecommunication Systems,* tom 62, p. 403–415, 2016.

[320] S. Mohammad-Azari, O. Bozorg-Haddad i X. Chu, „Shark Smell Optimisation (SSO) Algorithm," w *Advanced Optimisation by Nature-Inspired Algorithms. Studies in Computational Intelligence*, tom 720, O. Bozorg-Haddad, Red., Singapore, Springer, 2018.

[321] N. Farahani, K. Karami, S. Farzin i M. Ehteram, „A New Method for Flood Routing Utilizing Four-Parameter Nonlinear Muskingum and Shark Algorithm," *Water Resources Management,* tom 33, nr 1, 2019.

[322] S. Shadravan, H.R. Naji i V.K. Bardsiri, „The Sailfish Optimizer: A novel nature- inspired metaheuristic algorithm for solving constrained engineering Optimisation problems," *Engineering Applications of Artificial Intelligence,* tom 80, pp. 20–34, 2019.

[323] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris i S. M. Mirjalili, „Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in Engineering Software,* tom 114, pp. 163–191, 2017.

[324] H. Faris, M.M. Mafarja, A.A. Heidari, I. Aljarah, A.M. Al-Zoubi, S. Mirjalili i H. Fujita, „An efficient binary Salp Swarm Algorithm with crossover scheme for feature selection problems," *Knowledge-Based Systems,* tom 154, pp. 43–67, 2018.

[325] G.I. Sayed, G. Khoriba i M.H. Haggag, „A novel chaotic salp swarm algorithm for global Optimisation and feature selection," *Applied Intelligence,* tom 48, p. 3462–3481, 2018.

[326] D. Zaldívar, B. Morales, A. Rodríguez, A. Valdivia-G, E. Cuevas i M. Pérez-Cisneros, „A novel bio-inspired Optimisation model based on Yellow Saddle Goatfish behavior," *Biosystems,* tom 174, pp. 1–21, 2018.

[327] G. Shao-w, „Cuckoo Catfish Optimizer: A New Meta-Heuristic Optimisation," Matlab File Exchange, 2025.

[328] Z. Chen, „A modified cockroach swarm Optimisation," *Energy Procedia,* tom 11, pp. 4–9, 2011.

[329] Z. Chen i H. Tang, „Cockroach swarm Optimisation for vehicle routing problems," *Energy Procedia,* tom 13, pp. 30–35, 2011.

[330] L. Cheng, Z. B. Wang, Y. H. Song i A. H. Guo, „Cockroach Swarm Optimisation Algorithm for TSP," *Advanced Engineering Forum,* tom 1, pp. 226–229, 2011.

[331] T.C. Havens, C.J. Spain, N.G. Salmon i J. Keller, „Roach Infestation Optimisation," w *Swarm Intelligence Symposium, 2008. SIS 2008*, St. Louis, MO, USA, 2008.

[332] X. Yang, „The Standard Firefly Algorithm (FA)," 2024.

[333] L. Zhang, L. Liu, X.-S. Yang i Y. Dai, „A Novel Hybrid Firefly Algorithm for Global Optimisation," *PLOS One,* 2016.

[334] W.-T. Pan, „A new Fruit Fly Optimisation Algorithm: Taking the financial distress model as an example," *Knowledge-Based Systems,* tom 26, nr 2, pp. 69–74, 2012.

[335] S. Mirjalili, „Dragonfly algorithm: a new meta-heuristic Optimisation technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications,* tom 27, p. 1053–1073, 2016.

[336] S. Arora i S. Singh, „Butterfly Optimisation algorithm: a novel approach for global Optimisation," *Soft Computing,* tom 23, p. 715–734, 2019.

[337] S. Mirjalili, „Moth-flame Optimisation algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems,* tom 89, pp. 228–249, 2015.

[338] M. Shehab, L. Abualigah, H. Al Hamad, H. Alabool, M. Alshinwan i A.M. Khasawneh , „Moth-flame Optimisation algorithm: variants and applications," *Neural Computing and Applications,* tom 32, p. 9859–9884, 2020.

[339] M.H. Nadimi-Shahraki, H. Zamani, A. Fatahi i S. Mirjalili, „MFO-SFR: An Enhanced Moth-Flame Optimisation Algorithm Using an Effective Stagnation Finding and Replacing Strategy," *Mathematics,* tom 11, nr 4, 2023.

[340] K. Kaipa i D. Ghose, „Glowworm Swarm Optimisation: A New Method for Optimising Multi-Modal Functions," *International Journal of Computational Intelligence Studies,* tom 1, nr 1, 2009.

[341] D. Karaboga, „An idea based on honey bee swarm for numerical Optimisation," Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[342] M.K. Heris, „Artificial bee Colony in Matlab," 2015.

[343] G. Yan i C. Li, „An effective refinement artificial bee colony optimisation algorithm based on chaotic search and application for PID control tuning," *J Comput Inf Syst,* tom 7, nr 9, p. 3309–3316, 2011.

[344] M.A. Sotelo-Figueroa, R. Baltazar i M. Carpio, „Application of the Bee Swarm Optimisation BSO to the Knapsack Problem," w *Soft Computing for Recognition Based on Biometrics*, Berlin, Heidelberg, Springer Berlin Heidelberg, 2010, pp. 191–206.

[345] A. Askarzadeh i A. Rezazadeh, „Artificial bee swarm Optimisation algorithm for parameters identification of solar cell models," *Applied Energy,* tom 102, pp. 943–949, 2013.

[346] A. Migacz i R. Tadeusiewicz, „Model rodziny pszczelej," Akademia Medyczna, Kraków, 1979.

[347] A. Migacz i R. Tadeusiewicz, „The computer model of the bee colony," *System Science,* nr 3, pp. 83–95, 1983.

[348] D. Teodorović, „Bee colony Optimisation (BCO)," w *Innovations in Swarm Intelligence*, Berlin, Springer Berlin Heidelberg, 2009, pp. 39–60.

[349] O. Bozorg-Haddad, A. Afshar i M.A. Mariño, „Honey-Bees Mating Optimisation (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimisation," *Water Resources Management,* tom 20, nr 5, pp. 66–-680, 2006.

[350] T. Niknam, S.I. Taheri, J. Aghaei i S. Tabatabaei, „A modified honey bee mating Optimisation algorithm for multiobjective placement of renewable energy resources," *Applied Energy,* tom 88, nr 12, pp. 4817–4830, 2006.

[351] O. Acar, M. Kalyoncu i A. Hassan, „Proposal of a Harmonic Bees Algorithm for Design Optimisation of a Gripper Mechanism," w *Advances in Mechanism and Machine Science*, Springer, 2019, pp. 2829-2839.

[352] X.-S. Yang, „Engineering Optimisations via Nature-Inspired Virtual Bee Algorithms," w *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, Berlin Heidelberg, Springer Berlin Heidelberg, 2005, pp. 317-323.

[353] H. Sato i M. Hagiwara, „Bee System: Finding Solution by a Concentrated Search," w *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics Computational Cybernetics and Simulation*, Orlando, FL, USA, 1997.

[354] P. Lucic i D. Teodorović, „Bee system: Modeling combinatorial Optimisation transportation engineering problems by swarm intelligence," w *TRISTAN IV Triennial Symposium on Transportation Analysis*, Sao Miguel, Azores Islands, 2001.

[355] P. Navrat, A.B. Ezzeddine, L. Jastrzembska i T. Jelinek, „The Bee Hive At Work: Exploring its Searching and Optimizing," *INFOCOMP Journal of Computer Science,* tom 11, nr 1, pp. 32–40, 2012.

[356] W. K. Lai, J. E. Gan i P.M. Koh, „Artificial Honey Bee Swarm Intelligence for the Autograding of EBN," w *ICNC-FSKD 2019. Advances in Intelligent Systems and Computing, Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, 2020.

[357] S.H. Jung, „Queen-bee evolution for genetic algorithms," *Electronics Letters,* tom 39, nr 6, 2003.

[358] R. Solgi i H.A. Loáiciga, „Bee-inspired metaheuristics for global Optimisation: a performance comparison," *Artificial Intelligence Review,* nr 7, pp. 4967–4996, 2021.

[359] A. Colorni, M. Dorigo i V. Maniezzo, „Distributed Optimisation by Ant Colonies," w *Proceedings of the European Conference on Artificial Life, ECAL'91*, Paris, Elsevier Publishing, Amsterdam, 134–142, 1991.

[360] A. Rezvanian, S.M. Vahidipour i A. Sadollah, „An Overview of Ant Colony Optimisation Algorithms for Dynamic Optimisation Problems," w *Ant Colony Optimisation – Recent Variants, Application and Perspectives*, IntechOption, 2023.

[361] M.K. Heris, „Ant Colony Optimisation in MATLAB," Yarpiz, 2015.

[362] X.-S. Yang, J. M. Lees i C. T. Morley, „Application of Virtual Ant Algorithms in the Optimisation of CFRP Shear Strengthened Precracked Structures," w *6th International Conference Computational Science – ICCS 2006,*, Reading, UK, 2006.

[363] D. Zhao, L. Liu, F. Yu, A.A. Heidari, M. Wang, G. Liang, K. Muhammad i H. Chen, „Chaotic random spare ant colony Optimisation for multi-threshold image segmentation of 2D Kapur entropy," *Knowledge-Based Systems,* tom 216, p. nr. 106510, 2021.

[364] S. Mirjalili, „The Ant Lion Optimizer," *Advances in Engineering Software,* tom 83, pp. 80–98, 2015.

[365] X. Feng, F.C.M. Lau i D. Gao, „A New Bio-inspired Approach to the Traveling Salesman Problem," tom 5, pp. 1310–1321, 2009.

[366] H.A.R. Akkar i S.A. Salman, „Cicada Swarm Optimisation: A New Method for Optimizing Persistent Problems," *International Journal of Intelligent Engineering & Systems ,* tom 13, nr 6, 2020.

[367] G.-G. Wang, S. Deb i Z. Cui, „Monarch Butterfly Optimisation," *Neural Computing and Applications,* tom 31, nr 7, pp. 1995–2014, 2015.

[368] G.-G. Wang, „Moth search algorithm: a bio-inspired metaheuristic algorithm for global Optimisation problems," *Memetic Computing,* tom 10, p. 151–164, 2016.

[369] S. Saremi, S. Mirjalili i A. Lewis, „Grasshopper Optimisation Algorithm: Theory and application," *Advances in Engineering Software,* tom 105, pp. 30–47, 2017.

[370] M. Mafarja, I. Aljarah, A.A. Heidari i A.I. Hammouri, „Evolutionary Population Dynamics and Grasshopper Optimisation Approaches for Feature Selection Problems," *Knowledge-Based Systems,* tom 145, pp. 25–45, 2017.

[371] X. Jiang i S. Li, „BAS: Beetle Antennae Search Algorithm for Optimisation Problems," *BAS: Beetle Antennae Search Algorithm for Optimisation Problems,* tom 1, nr 1, 2018.

[372] J.J. Q. Yu i V.O.K. Li, „A social spider algorithm for global Optimisation," *Applied Soft Computing,* tom 30, pp. 614–627, 2015.

[373] V. H. S. Pham i N. T. Nguyen Dang, „Portia spider algorithm: an evolutionary computation approach for engineering application," *Artificial Intelligence Review,* tom 57, p. artykuł nr. 24, 2024.

[374] A.H. Gandomi i A.H. Alavi, „Krill herd: A new bio-inspired Optimisation algorithm," *Communications in Nonlinear Science and Numerical Simulation,* tom 17, nr 12, pp. 4831–4845, 2012.

[375] A.L. Bolaji, M.A. Al-Betar, M. A. Awadallah, A. Y. Khader i L. M. Abualigah, „A comprehensive review: Krill Herd algorithm (KH) and its applications," *Applied Soft Computing,* tom 49, pp. 437–446, 2016.

[376] L.M.Q. Abualigah, Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering, Springer Cham, 2019.

[377] L.M.Q. Abualigah, A.T.A. Khader i E. S. Hanandeh, „A hybrid strategy for krill herd algorithm with harmony search algorithm to improve the data clustering," *Intell. Decision Technol.,* tom 12, pp. 3–14, 2018.

[378] G.-G. Wang, L. Guo, A. H. Gandomi i G. Hao, „Chaotic Krill Herd algorithm," *Information Sciences ,* tom 274, pp. 17–34, 2014.

[379] G.-G. Wang, S. Deb i L. d. S. Coelho, „Earthworm Optimisation algorithm: a bio- inspired metaheuristic algorithm for global Optimisation problems," *International Journal of Bio-Inspired Computation,* tom 2, nr 1, 2018.

[380] B. Niu i H. Wang, „Bacterial Colony Optimisation," *Discrete Dynamics in Nature and Society,* 2012.

[381] B. Xing i W.-J. Gao, „Bacteria Inspired Algorithms," w *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms*, Springer International Publishing, 2014, pp. 21–38.

[382] K.M. Passino, „Biomimicry of bacterial foraging for distributed Optimisation and control," *EEE Control Systems Magazine,* tom 22, nr 3, pp. 52–67, 2002.

[383] H. Bremermann, „Chemotaxis and Optimisation," *Journal of the Franklin Institute,* tom 297, nr 5, pp. 397–404, 1974.

[384] S.D. Muller, J. Marchetto, S. Airaghi i P. Kournoutsakos, „Optimisation based on bacterial chemotaxis," *IEEE Transactions on Evolutionary Computation,* tom 6, nr 1, pp. 16–29, 2002.

[385] R. Zhang, J. Zhou, Y. Lu, H. Qin i H. Zhang, „A PSO-Based Bacterial Chemotaxis Algorithm and Its Application," w *Advances in Neural Networks – ISNN 2011 International Symposium on Neural Networks, Lecture Notes in Computer Science (LNTCS)*, Berlin, Heidelberg, 2011.

[386] M.D. Li, H. Zhao, X. W. Weng i T. Han, „A novel nature-inspired algorithm for Optimisation: Virus colony search," *Advances in Engineering Software,* tom 92, pp. 65–88, 2016.

[387] P. Cortés, J. M. García, J. Muñuzuri i L. Onieva, „Viral systems: A new bio-inspired optimisation approach," *Computers & Operations Research,* tom 35, nr 9, pp. 2840–2860, 2008.

[388] J.R.C. Juarez, H.-J. Wang, Y.-C. Lai i Y.-C. Liang, „Virus Optimisation Algorithm (VOA): A Novel Metaheuristic for Solving Continuous Optimisation Problems," w *Proceedings of 10thAsia Pacific Industrial Engineeering & Management System Conference (APIEMS)*, Kitakyushu, Japan, 2009.

[389] M. Jaderyan i H. Khotanlou, „Virulence Optimisation Algorithm," *Applied Soft Computing,* tom 43, pp. 596–618, 2016.

[390] A. Brabazon i S. McGarraghy, Foraging-Inspired Optimisation Algorithms, Spronger, 2018.

[391] S. A. Uymaz, G. Tezel i E. Yel, „Artificial algae algorithm (AAA) for nonlinear global Optimisation," *Applied Soft Computing,* tom 31, pp. 153–171, 2015.

[392] F. Merrikh-Bayat, „The runner-root algorithm: A metaheuristic for solving unimodal and multimodal Optimisation problems inspired by runners and roots of plants in nature," *Applied Soft Computing,* tom 33, pp. 292–303, 2015.

[393] A. Cheraghalipour, M. Hajiaghaei-Keshteli i M.M. Paydar, „Tree Growth Algorithm (TGA): A novel approach for solving Optimisation problems," *Engineering Applications of Artificial Intelligence,* tom 72, pp. 393–414, 2018.

[394] D.R. Monismith i B.E. Mayfield, „Slime Mold as a model for numerical Optimisation," w *2008 IEEE Swarm Intelligence Symposium*, St. Louis, MO, USA, 2008.

[395] A.R. Mehrabian i C. Lucas, „A novel numerical Optimisation algorithm inspired from weed colonization," *Ecological Informatics,* tom 1, nr 4, pp. 355–366, 2006.

[396] M.K. Heris, „Invasive Weed Optimisation (IWO) in MATLAB," Yarpiz, 2015.

[397] X.-S. Yang, „Flower Pollination Algorithm for Global Optimisation," w *Unconventional Computation and Natural Computation. UCNC 2012. Lecture Notes in Computer Science*, 2012.

[398] Z. Cui, S. Fan, J.-C. Zeng i Z. Shi, „APOA with parabola model for directing orbits of chaotic systems," *International Journal of Bio-Inspired Computation,* tom 5, nr 1, pp. 62–72, 2013.

[399] A.S. Eesa, A. M. Abdulazeez i Z. Orman, „A Novel Bio-Inspired Optimisation Algorithm," *International Journal of Scientific and Engineering Research,* tom 4, nr 9, pp. 1978–1986, 2013.

[400] A.S. Eesa, Z. Orman i A. M. A. Brifcani,, „A novel feature-selection approach based on the cuttlefish Optimisation algorithm for intrusion detection systems," *Expert Systems with Applications,* tom 42, nr 5, pp. 2670–2679, 2015.

[401] X. Li, J. Zhang i M. Yin, „Animal migration Optimisation: an Optimisation algorithm inspired by animal migration behavior," *Neural Computing and Applications,* tom 24, p. 1867–1877, 2014.

[402] L. Rosenberg, „Artificial Swarm Intelligence, a Human-in-the-Loop Approach to A.I.," w *Proceedings of the 13th AAAI-16 Conference on Artificial Intelligence*, 2016.

[403] A. Farasat, M. B. Menhaj, T. Mansouri i M. R. S. Moghadam, „ARO: A new model- free Optimisation algorithm inspired from asexual reproduction," *Applied Soft Computing,* tom 10, nr 4, pp. 1284–1292, 2010.

[404] X. Yuan, Y. He i L. Liu, „Parameter extraction of solar cell models using chaotic asexual reproduction Optimisation," *Neural Computing and Applications,* tom 26, p. 1227–1239, 2014.

[405] S.H. Pakzad-Moghaddam, H. Mina i P. Mostafazadeh, „A novel Optimisation booster algorithm," *Computers & Industrial Engineering,* tom 136, pp. 591–613, 2019.

[406] M.-Y. Cheng i D. Prayogo, „Symbiotic Organisms Search: A new metaheuristic Optimisation algorithm," *Computers & Structures,* tom 139, pp. 98–112, 2014.

[407] H.T. Kahraman, S. Aras i E. Gedikli, „Fitness-distance balance (FDB): A new selection method for meta-heuristic search algorithms," *Knowledge-Based Systems,* tom 190, p. nr. 105169, 2020.

[408] Y. Fu, W. Zghang, C. Qu i B. Huang, „Optimal Foraging Algorithm Based on Differential Evolution," *IEEE Access,* tom 8, p. 19657–19678, 2020.

[409] G. Kyrou, V. Charilogis i I. G. Tsoulos , „EOFA: An Extended Version of the Optimal Foraging Algorithm for Global Optimisation Problems," *MDPI Computation,* tom 12, nr 8, 2024.

[410] D. Baczyński, „Metody inteligencji obliczeniowej w elektroenergetyce," *Prace Naukowe Politechniki Warszawskiej. Elektryka,* tom 145, pp. 3–157, 2013.

[411] W. Zhao, L. Wang i Z. Zhang, „Artificial ecosystem-based Optimisation: a novel nature-inspired meta-heuristic algorithm," *Neural Computing and Applications,* tom 32, nr 13, p. 9383–9425, 2019.

[412] W. Zhao, „Artificial Ecosystem-based Optimisation (AEO)," MathWorks, 2020.

[413] S. Salcedo-Sanz, P. García-Díaz, A. Portilla-Figueras i J. Del Ser, „A Coral Reefs Optimisation Algorithm for Optimal Mobile Network Deployment with Electromagnetic Pollution Control Criterion," *Applied Soft Computing,* tom 24, nr 3, p. 239–248, 2014.

[414] P. Kumar, „Red Kite praveen kumar (2025). Red Kite Optimisation Algorithm," matlab Central File Exchange.

[415] P. Niu, S. Niu, N. Liu i L. Chang, „The defect of the Grey Wolf Optimisation algorithm and its verification method," *Knowledge-Based Systems,* tom 171, pp. 37–43, 2019.

[416] S. Roy i S.S. Chaudhuri, „Cuckoo Search Algorithm using Lèvy Flight: A Review," *International Journal of Modern Education and Computer Science (IJMECS),* tom 5, nr 12, pp. 10–15, 2013.

[417] M. Mareli i B. Twala, „An adaptive Cuckoo search algorithm for optimisation," *Applied Computing and Informatics,* tom 14, nr 2, 2017.

[418] S. Surjanovic i D. Bingham, „Virtual Library of Simulation Experiments: Test Functions and Datasets".

[419] E. Taillard, „Benchmarks for basic scheduling problems," *European Journal of Operational Research,* tom 64, nr 2, pp. 278–285, 1993.

[420] N. Awad, M. Ali, J. Liang, B. Qu i P. Suganthan, „Problem Definitions Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real – Parameter Numerical Optimisation," w *IEEE Congress on Evolutionary Computation (CEC)*, San Sebastian, Spain, 2017.

[421] S. Kirkpatrick, C.D. Gelatt i M.P. Vecchi, „Optimisation by Simulated Annealing," *Science,* tom 220, nr 4598, pp. 671–680, 1983.

[422] D.H. Ackley, „An Empirical Study of Bit Vector Function," w *Genetic Algorithms and Simulated Annealing*, Londyn, Pittman Publishers, 1987.

[423] E.H. Aarts i J.K. Lenstra, Local Search in Combinatorial Optimisation, Princeton University Press, 2003.

[424] I. Demirkol, C. Ersoy, M. U. Caglayan i H. Delic, „Location Area Planning in Cellular Networks Using Simulated Annealing," w *INFOCOM*, Anchorage, AK, 2001.

[425] R.J. Katulski i M. Kopciuszuk, „Komputerowy system ekspertowy do projektowania bazowej stacji radiokomunikacyjnej," *Przegląd Telekomunikacyjny,* 11.12.2002.

[426] M. Wikelski, The Internet of Animals: Discovering the Collective Intelligence of Life on Earth, Greystone Books, 2024.

[427] E. Cassell, Algorytmy zwierząt. Ewolucja a tajemnica zadziwiających instynktów, Fundacja En Arche, 2022.

[428] B. Filipowicz i J. Kwiecień, „Algorytmy stadne w problemach optymalizacji," *Pomiary Automatyka, Robotyka,* tom 12, pp. 152–157, 2011.

[429] M.K. Heris, „Bees Algorithm (BeA) in MATLAB," Yarpiz, 2015.

[430] K. Trojanowski, Metaheurystyki praktycznie, wyd. 2, Warszawa: WIT, 2008.

[431] „Particle swarm Optimisation – MATLAB particleswarm," MathWorks.

[432] M.K. Heris, „Implementation of Particle Swarm Optimisation in MATLAB," Yarpiz, 2015.

[433] X.-S. Yang, „Multiobjective firefly algorith for continuous Optimisation," *Engineering with Computers,* tom 29, nr 2, pp. 175–184, 2013.

[434] Y.Y. Haimes, L.S. Lasdon i D.A. Wismer, „On A Bicriterion Formulation of the Problems of Integrated System Identyfication and System Optimisation," *IEEE Trans. Syst. Man. Cybern.,* tom 1, nr 4, pp. 296–297, 1971.

[435] „Strona internetowa firmy MathWorks zawierająca dokumentację: Multiobjective Optimisation Algorithms".

[436] J.D. Shaffer, *Some Experiments in Machine Learnin Using Vector Evaluated Genetic Algorithms,* Nashville: Ph. D. Dissertation, Vanderblit University, 1984.

[437] C.M. Fonseca i P.J. Fleming, „An Overview of Evolutionary Algorithms in Multiobjective Optimisation," *Evolutionary Computation,* tom 3, nr 1, pp. 165–180, 1995.

[438] E. Zitzler i L. Thiele, „An Evolutionary Algorithm for Multiobjective Optimisation: The Strength Pareto Approach," Swiss Federal Institute of Technology Technical Report 43, Zurich, 1998.

[439] E. Zitzler, M. Laumanns i L. Thiele, „SPEA2 : Improving the Strength Pareto Evolutionary Algorithm," w *Eurogen*, Ateny, 2001.

[440] N. Srinivas i K. Deb, „Multiobjective Optimisation Using Nondominated Sorting in Genetic Algorithm," *Evolutionary Computation,* tom 2, nr 3, pp. 221–248, 1994.

[441] K. Deb, S. Agrawal, A. Pratap i T. Mayerivan, „A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimisation: NSGA II," w *IEEE Transactions on Evolutionary Computation*, 2000.

[442] D. Kalyanmoy, Multi-Objective Optimisation using Evolutionary Algorithms, Wiley, 2001.

[443] K. Deb i H. Jain, „An Evolutionary Many-Objective Optimisation Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation,* tom 18, nr 4, 2014.

[444] S. Mostapha Kalami Heris, „NSGA-III: Non-dominated Sorting Genetic Algorithm, the Third Version," Yarpiz, 2016.

[445] C.A. Coello i G.T. Pulido, „A Micro-Genetic Algorithm for Multiobjective Optimisation," *Lecture Notes in Computer Science,* pp. 126–140, 1 2001.

[446] C. Coello, A. Coello i M. Salazar Lechuga, „MOPSO: A Proposal for Multiple Objective Particle Swarm Optimisation," w *CEC Congress on Evolutionary Computation*, New Jersey, USA, 2002.

[447] S. Mostapha Kalami Heris, „Multi-Objective PSO in MATLAB," 2015.

[448] X.-S. Yang i S. Deb, „Multiobjective cuckoo search for design Optimisation," *Computers & Operations Research,* tom 40, nr 6, pp. 1616–1624, 2013.

[449] K. Deb, Multi-objective Optimisation using evolutionary algorithms, New York: John Wiley & Sons, 2001.

[450] X.-S. Yang, „Multiobjective Cuckoo Search (MOCS) – File Exchange – MATLAB Central," [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/74752-multiobjective- cuckoo-search-mocs. [Data uzyskania dostępu: 23.10.2024].

[451] „Multi-Objective Goal Attainment Optimisation - MATLAB, Simulink," [Online]. Available: https://www.mathworks.com/help/optim/ug/multiobjective--pole- placement.html. [Data uzyskania dostępu: 23.10.2024].

[452] „Generate and Plot Pareto Front – MATLAB; Simulink," [Online]. Available: https://www.mathworks.com/help/optim/ug/generate-and-plot-a-pareto-front.html. [Data uzyskania dostępu: 23.10.2024].

[453] P. Roshan i J. Leary, „Bezprzewodowe sieci LAN 802.11 Podstawy", Mikom, 2004.

[454] „Methodology for Testing Wireless LAN Performance with Chariot," http://www.atheros.com/media/resource/resource_21_file2.pdf.

[455] P. Wertz, M. Sauter, G. Wölfle, R. Hoppe i F. M. Landstorfer, „Automatic Optimisation Algorithms for the Planning of Wireless Local Area Networks," w *VTC*, 2004.

[456] A.A. Kannan, G. Mao i B. Vucetic , „Simulated Annealing based Location in wireless Sensor Network," w *LCN*, 2005.

[457] S. Nistal-Ariza, A. Fernandez-Duran i J.I. Alonso, „Simulation based Algorithm Comparison for planning and optimisation of Indoor Wireless Networks," w *RWS*, 2008.

[458] M. Kamenetsky i M. Unbehaun, „Coverage planning for Outdoor Wireless LAN Systems," w *International Seminar on Broadband Communications*, Zurich, 2002.

[459] H.D. Sherali, C.M. Pendyala i T.S. Rappaport, „Optimal Location of Transmitters for Nicrocellular Radio Comminication System Design," *IEEE JSAC,* tom 14, nr 4, 1996.

[460] M. Unbehaun i M. Kamenetsky, „On the Deployment of Picocellular Wireless Infrastructure," *IEEE Wireless Communication,* nr 12, 2003.

[461] S. Kouhbor, J. Ugon, A. Kruger i A. M. Rubinov, „Optimal Placement of Access Point in WLAN Based on a New Algorithm," w *ICMB*, 2005.

[462] S. Kouhbor, J. Ugon, A. Kruger, A. Rubinov i P. Branch, „A New Algorithm for the Placement of WLAN Access Points Based on Nonsmooth Optimisation Technique," w *ICACT*, Phoenix Park, Korea, 2005.

[463] T. Jiang i G. Zhu, „Uniform Design Simulated Annealing for Optimal Access Point Placement of High Data Rate Indoor Wireless LAN Using OFDM," w *PIMRC*, 2003.

[464] M. Kobayashi, S. Haruyama, R. Kohno i M. Nakagawa, „Optimal Access Point Placement in Simultaneous Broadcast System using OFDM for Indoor Wireless LAN," w *PIMRC*, 2000.

[465] M. Kobayashi, T. Aritatt, T. Udagawatt, A. Kajiwaratt i M. Nakagawa, „Overlapped – Spot Diversity using Orthogonal Frequency Division Multiplexing for 60 GHz Indoor Wireless Local Area Network," w *ICC*, 2000.

[466] G.E. Athanasiadou, A.R. Nix i J.P. McGeehan, „A Microcellular Ray-Tracing Propagation Model and evaluation of its Narrow-Band and Wide-Band Predictions," *IEEE JSAC,* tom 18, nr 3, 2000.

[467] M. Ergen, B. Dundar i P. Varaiya, „Throughput analysis of an extended service set in IEEE 802.11," w *GLOBECOM*, 2004.

[468] J. Konorski, „Capacity-fairness performance of an ad hoc IEEE 802.11 WLAN with noncooperative stations," w *Proceedings of the 6th international IFIP-TC6 conference on Ad Hoc and sensor networks, wireless networks, next generation internet*, 2007.

[469] I. Rechenberg, „Evolutionsstrategie", Frommann-Holzboog, 1994.

[470] L.J. Fogel, „Autonomous automata," *Industrial Research,* tom 4, pp. 14–19, 1962.

[471] J. Branke, K. Deb i K. Miettinen, „Multiobjective Optimisation: Interactive and Evolutionary Approaches", Berlin Heidelberg, Germany: Springer, 2008.

[472] „IEEE Standard for Information technology--Local and metropolitan area networks--Specific requirements--Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput,"," IEEE, 2009.

[473] „IEEE Standard for Information technology--Telecommunications and information exchange between systemsLocal and metropolitan area networks--Specific requirements--Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, 2013.

[474] „IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks--Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, 2021.

[475] „IEEE Approved Draft Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks-- Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)," IEEE, 2024.

[476] „Overview of Wi-Fi 7 (IEEE," MathWorks, 2024. [Online]. Available: https://www.mathworks.com/help/wlan/ug/overview-of-wifi-7-or-ieee-802-11-be.html. [Data uzyskania dostępu: 29.12.2024].

[477] „VIEW Configuration Guide Cisco 1131, 1232 and 1242 Autonomous APs," 2012.

[478] L. Kleinrock i F. Tobagi, „Packet switching in radio channels. I. Carrier sense multiple access models and their throughput and delay characteristics", *IEEE Trans. on Commmun,* nr 12, pp. 1400–1416, 1975.

[479] T. Adame, M. Carrascosa-Zamacois i B. Bellalta, „Time-Sensitive Networking in IEEE 802.11be: On the Way to Low-Latency WiFi 7," *Sensors,* tom 21, p. nr 4954, 2021.

[480] D. Cavalcanti, C. Cordeiro, M. Smith i A. Regev, „WiFi TSN: Enabling Deterministic Wireless Connectivity over 802.11," *EEE Communications Standards Magazine,* tom 6, nr 4, pp. 22–29, 2022.

[481] S. Avallone, P. Imputato i D. Magrin, „Controlled Channel Access for IEEE 802.11- Based Wireless TSN Networks," *IEEE Internet of Things Magazine,* tom 6, nr 1, pp. 90–95, 2023.

[482] W.J. Krzyztofik i K. Jurczyk, „Implementacja systemów bezprzewodowych bluetooth i wlan w instytucjach na przykładzie uczelni," w *KKRRiT*, 2003.

[483] W. Lewis, „Przełączanie sieci LAN i sieci bezprzewodowe Akademia sieci Cisco. CCNA Exploration. Semestr 3", PWN, 2011.

[484] K. Staniec i M. Kowal, „A Simple Method for Determining an Optimal Number of Access Points in Distributed WLAN Networks," *ELEKTRONIKA IR ELEKTROTECHNIKA,,* tom 19, nr 9, pp. 101–104, 2013.

[485] *Rozporządzenie Ministra Transportu z dnia 3 lipca 2007 r. w sprawie urządzeń radiowych nadawczych lub nadawczo-odbiorczych, które mogą być używane bez pozwolenia radiowego (Dz.U. 2007 nr 138 poz. 972),* 2007.

[486] N. Levanon, „Lowest GDOP in 2-D scenarios," *IEEE Proceedings-Radar, Sonar and Navigation,* tom 147, nr 3, pp. 149–155, 2000.

[487] P. Di Barba, S. Hausman, P. Korbel i K. Piwowarczyk, „Ewolucyjne podejście do automatycznego rozmieszczania punktów dostępowych WLAN do jednoczesnej poprawy zasięgu i dokładności lokalizacji," *Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne,* nr 6, pp. 299–302, 6 2017.

[488] P. Korbel, S. Hausman i P. Di Barba, „Dwukryterialne ewolucyjne podejście do optymalizacji rozmieszczenia węzłów dostępowych dla poprawy zasięgu i dokładności lokalizacji terminali," *Przegląd Telekomunikacyjny – WiadomościTelekomunikacyjne,* pp. 493–496, 6 2019.

[489] A. Luntovskyy, S. Uhlig, D. Gutter i A. Schill, „Protocol stack and capacity modeling for WLAN," w *Next Generation Internet Networks, 3rd EuroNGI*, 2007.

[490] G. Bianchi, „Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *Journal on Selected Areas in Communications, JSAC IEEE,* pp. 535–547, 3 vol.18 2000.

[491] „Methodology for Testing Wireless LAN Performance," [Online]. Available: http://www.super-g.com/collateral/atheros_benchmark_whitepaper.pdf.

[492] A. Pieprzycki i W. Ludwin, „Analiza porównawcza wybranych modeli sieci WLAN standardu IEEE 802.11 z infrastrukturą," w *KSTiT*, 2012.

[493] W. Sobczak, Podstawy probabilistyczne teorii systemów informacyjnych, WNT, 1981.

[494] A. Pieprzycki i P. Świętojański, Optymalizacja rozmieszczenia punktów dostępowych (AP) wewnątrz budynku dla lokalnej sieci bezprzewodowej (WLAN) standardu 802.11b/g, KKRRiT 2009 / Przegląd Telekomunikacyjny" 6/2009, 2009.

[495] H. Bogucka, Projektowanie i obliczenia w radiokomunikacji, Politechnika Poznańska, 2005.

[496] R.J. Katulski, Propagacja fal radiowych w telekomunikacji bezprzewodowej, WKŁ, 2009.

[497] J. Szóstka i R. Wieczorkiewicz, „Pomiary kontrolne mocy w sieciach WLAN," w *KKRRiT*, 2005.

[498] K. Staniec i R. J. Zieliński, „Deterministyczne metody predykcji natężenia pola EM we wnętrzach budynków," w *KKRRiT*, 2003.

[499] T. S. Rappaport, Wireless Communications – Principle and Practice, Prentice-Hall, 1996.

[500] J.-F. Kiang, „Geometrical ray tracing approach for indoor wave propagation in corridor," w *Universal Personal Communications, ICUPC*, 1992.

[501] S. Haussman, „Modelowanie odbić i dyfrakcji fal radiowych w środowisku miejskim," w *Elektronika Prace Naukowe Politechniki Łódzkiej Zeszyt Nr 3*, 1998.

[502] W. Hornacharenko, H. L. Bertoni i J. Dailing, „Mechanisms governing propagation between different floors in building," *IEEE,* 1993.

[503] D. I. Laurenson, S. McLaughlin i U. H. Sheikl, „The application of ray tracing and the geometrical theory of diffraction to indoor channel moddeling," *IEEE Trans. Veh. Technol.,* tom 44, nr 3, pp. 494–505, 1995.

[504] „ITU-R P.1238-2 Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 900 MHz to 100 GHz," ITU.

[505] W. Mardini i T. Kacprzak, „Symulacja propagacji fal radiowych z wykorzystaniem metody śledzenia wiązek," w *KKRRiT*, 2006.

[506] K. Kurek i J. Modelski, „Analiza przestrzennego rozkładu sygnałów wielodrogowych propagujących wewnątrz pomieszczenia," w *KKRRiT*, 2003.

[507] A. Pieprzycki, „Budowa symulatora propagacji 3D wykorzystującego metodę śledzenia promieni," w *KSTiT*, 2008.

[508] P. Kułakowski i W. Ludwin, „Zastosowanie metod Ray Tracing do symulacji systemu MIMO (4,4) w środowisku wewnątrz budynków," w *KKRRiT*, 2004.

[509] „3D Indoor Intelligent Ray Tracing," [Online]. Available: http://www.awe-communications.com/Propagation/Indoor/IRT/index.htm.

[510] K. Krzyżak i E. Herko, „Wykorzystywanie uproszczonych modeli propagacji fal radiowych do symulacji działania bezprzewodowych sieci komputerowych we wnętrzach budynków," w *KKRRiT*, 2003.

[511] W. J. Krzysztofik i P. Herbatowski, „Modele propagacyjne wykorzystywane do projektowania sieci WLAN wewnątrz budynków," w *KKRRiT*, 2006.

[512] R.J. Katulski, A. Lipka i K. Turek, „Badania rozkładu mocy sygnału radiowego w sieciach WLAN," w *KKRRiT*, 2006.

[513] „RECOMMENDATION ITU-R M.1225 z aneksami „Guidelines for Evaluation of radio Transmission Technology FOR IMT-2000".

[514] M. Lott i I. Forkel, „A Multi-Wall-and-Floor Model for Indoor Radio Propagation," w *VTC*, 2001.

[515] A. Charytoniuk, „Uwarunkowania środowiskowe dokładności metod analitycznych predykcji natężenia pola fal radiowych propagujących w budynkach," w *KKRRiT*, 2002.

[516] K. Wesołowski, Systemy radiokomunikacji ruchomej, WKŁ, 1998.

???

[517] T. Klajbor, „Pakietowa stopa błędów IEE 802.11b w aspekcie koegzystancji z technlogią Bluetooth," w *KST*, 2006.

[518] E. Neufert, Podręcznik projektowania architektoniczno-budowlanego, Arkady, 1992.

[519] V. K. Garg, IS-95 CDMA and cdma2000 Cellular/PCS System Implementation, PH PTR, 2000.

[520] W. C. Jakes, Microwave Mobile Communications, New York,USA: IEEE Press .

[521] F.G. Gregorio, „802.11a-OFDM phy Coding and Interleaving," Helsinki University of Technology.

[522] S. Bradner i J. McQuaid, „Benchmarking Methodology for Network Interconnect Devices RFC 2544," IETF, 1999.

[523] Z. Hadzi-Velkov i B. Spasenovski, „Capture effect in IEEE 802.11 basic service area under influence of Rayleigh fading and near/far effect," w *13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Lizbona, Portugalia, 2002.

[524] X. Chen, H. Zhai, X. Tian i Y. Fang, „Supporting QoS in IEEE 802.11e Wireless LANs," *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS,* tom 5, nr 8, pp. 2217-2227, 2006.

[525] G. Prakash, i P. Thangaraj, „Non-Saturation Throughput Analysis of IEEE 802.11 Distributed Coordination Function," *European Journal of Scientific Research,* tom 51, nr 2, 2011.

[526] Z. Kong, D. H. Tsang, B. Bensaou i D. Gao, „Performance Analysis of IEEE 802.11e Contention-Based Channel Access," *IEEE JSAC,* tom 22, nr 10, 2004.

[527] J. Mogul i S. Deering, „Path MTU Discovery RFC 1191," 1990.

[528] K. Staniec i R. J. Zieliński, „Analiza porównawcza teoretycznej i zmierzonej przepustowości systemów 802.11b i 802.11a," w *KKRRiT*, 2005.

[529] P. Peddabachagari, M. Sichitiu i J. Jun, „Theoretical Maximum Throughput of IEEE 802.11 and its Applications," w *NCA ,03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, 2003.

[530] Y. Xiao i J. Rosdahl, „Throughput and Delay Limits of IEEE 802.11," *IEEE COMMUNICATIONS LETTERS,* VOL. 6, NO. 8, 2002.

[531] J. Jun, P. Peddabachagari i M. Sichitiu, „Theoretical Maximum Throughput of IEEE 802.11 and its Applications," w *IEEE International Symposium on Network Computing and Applications NCA*, 2003.

[532] M. Heusse, F. Rousseau, G. Berger-Sabbatel i A. Duda, „Performance anomaly of 802.11b," w *IEEE Infocom*, 2003.

[533] M. Natkaniec, R. Kowalski i K. Kostecki, „Analiza zjawiska anomalii wydajności w sieciach standardu IEEE 802.11ac," w *KKRRiT / PT 4/2015*, Łódź, 2015.

[534] K. Szczypiorski i J. Lubacz, „Saturation troughput analysis of IEEE 802.11g (ERP-OFDM) systems," w *IFIP International Conference on Personal Wireless Communications PWC*, 2007.

[535] K. Szczypiorski i J. Lubacz, „Saturation troughput analysis of IEEE 802.11g (ERP-OFDM) networks," *Telecommunication Systems: Modelling, Analysis, Design and Management,* tom 38, nr 1–2, 2008.

[536] Q. Ni, T. Li, T. Turletti i Y. Xiao , „Saturation Throughput Analysis of Error-Prone 802.11 Wireless Networks," *Wiley Journal of Wireless Communications and Mobile Computing (JWCMC),* tom 5, nr 8, 2005.

[537] K. Kosek-Szott, M. Natkaniec i A. R. Pach, „A simple but accurate throughput model for IEEE 802.11 EDCA in saturation and non-saturation conditions," *Computer Networks,* tom 55, nr 3, 2011.

[538] F. Cali, M. Conti i E. Gregori, „IEEE 802.11 Wireless LAN: Capacity Analysis and Protocol Enhancement," w *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, San Francisco, USA, 1998.

[539] D. Yang, T. Lee, K. Jang i J. Choi, „Performance Enhancement of Multi-Rate IEEE 802.11 WLANs with Geographically-Scattered Stations," *IEEE Trans. Mobile Computing,* tom 5, nr 7, pp. 906–919, 2006.

[540] J. Choi, K. Park i C.-k. Kim, „Cross-Layer Analysis of Rate Adaptation, DCF and TCP in Multi-Rate WLANs," w *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, 2007.

[541] H. Wu, Y. Peng, K. Long, S. Cheng i J. Ma, „Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement," w *IEEE INFOCOM*, 2002.

[542] Y. Peng, K. Long, J. Cheng i M. H. Wu, „Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement," w *IEEE INFOCOM*, 2002.

[543] A. Pieprzycki i W. Ludwin, „Analiza porównawcza wybranych metod optymalizacji w zagadnieniach planowania sieci WLAN IEEE 802.11 z infrastrukturą," w *KKRRiT*, 2014.

[544] K. Kukuła, Metoda unitaryzacji zerowanej, WN PWN, 2000.

[545] G.G. Yen i Z. He, „Performance Metrics Ensemble for Multiobjective Evolutionary Algorithms," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,* tom 18, nr 1, pp. 131–144, 2014.

[546] P.A.N. Bosman i D. Thierens, „The Balance between Proximity and Diversity in Multiobjective Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation,* tom 7, pp. 174–188, 2003.

[547] F. Freschi i M. Repetto, „Multiobjective Optimisation by a Modified Artificial Immune System Algorithm," w *International Conference on Artificial Immune Systems ICARIS*, 2005.

[548] A. Pieprzycki i W. Ludwin, „Planowanie sieci WLAN za pomocą wielokrytaralnego algorytmu kukułki," w *KKRRiT*, Gdańsk, 2018.

[549] A. Pieprzycki i W. Ludwin, „Wybrane aspekty optymalizacji wielokryterialnej w planowaniuJ sieci WLAN," w *KSTiT*, 2018.

[550] A. Kaveh i T. Bakhshpoori, „An efficient multi-objective cuckoo search algorithm for design Optimisation," *Advances in Computational Design,* tom 1, nr 1, pp. 87–103, 2016.

[551] A. Pieprzycki i W. Ludwin, „Wybrane przykłady zadań optymalizacji wielokryterialnej w planowaniu sieci WLAN," *Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne,,* nr 7, pp. 699–704, 2019.

[552] A. Pieprzycki i W. Ludwin, „Zastosowanie algorytmów rojowych w zadaniu planowania sieci WLAN," *TCN,* 4/2017.

[553] A. Pieprzycki i W. Ludwin, „Wybrane zagadnienia wielokryterialnego planowania sieci WLAN," *Science, Technology and Innovation,* 2/2018.

[554] A. Pieprzycki i W. Ludwin, „Pomiary przepustowości w sieciach WLAN standardu IEEE 802.11 w paśmie ISM 2.4 GHz w środowisku zakłóceń interferencyjnych," w *Krajowa Konferencja Radiokomunikacji, Radiofonii i Telewizji KKRRiT*, Łódź, 2015.

[555] A. Pieprzycki, „Konfiguracja i pomiary w sieci WLAN na przykładzie systemu operacyjnego Linux," w *PWT*, Poznań, 2007.

[556] A. Pieprzycki i P. Świętojański, „Wpływ protokołu CSMA/CA na planowanie sieci 802.11b/g w środowisku wewnątrzbudynkowym," w *KSTiT*, 2011.

[557] A. Pieprzycki i W. Ludwin, „Analiza przepustowości w sieciach WLAN IEEE 802.11 dla potrzeb planowania," w *PWT*, Poznań, 2013.

[558] A. Pieprzycki, (Thesis) Planowanie i optymalizacja rozmieszczenia punktów dostępu do sieci WLAN standardu IEEE 802.11 w środowisku wewnątrzbudynkowym, Kraków: AGH, 2019.

[559] M. Kowal, „Analiza wpływu zakłóceń na działanie interfejsu radiowego wykorzystującego technikę MIMO-OFDM," Politechnika Wrocławska - rozprawa doktorska, 2011.

[560] J. Szóstka, Mikrofale, WKŁ, 2006.

[561] J. Tourrilhes. [Online]. Available: http://www.hpl.hp.com/personal/Jean_ Tourrilhes/Linux/Linux.Wireless.Extensions. html.

[562] J. Tourrilhes. [Online]. Available: http://www.hpl.hp.com/personal/Jean_ Tourrilhes/Linux/Tools.html.

[563] K. Wilczura i W. J. Krzysztofik, „Pomiary parametrów transmisyjnych systemów WLAN," w *KKRRiT*, 2006.

[564] A. Knaff, „GNU Operating System," [Online]. Available: http://www.gnu.org/ software/mtools/.

[565] J. Laine, S. Saaristo i R. Prior, „RUDE & CRUDE," Source Forge, [Online]. Available: http://rude.sourceforge.net/.

[566] R.J. Zieliński i K. Staniec, „Analiza porównawcza teoretycznej i zmierzonej przepustowości systemów 802.11b i 802.11a," w *KKRRiT*, 2005.

[567] J.R. Taylor, Wstęp do analizy błędu pomiarowego, PWN, 1999.

[568] S.M. Kot, J. Jakubowski i A. Sokołowski, Statystyka, Difin, 2011.

[569] „Iperf," [Online]. Available: http://iperf.sourceforge.net/.

[570] „UDPgenerator – Projects: The University of Michigan and Merit Internet2 Qbone Testbed," [Online]. Available: http://www.citi.umich.edu/projects/ qbone/generator.html.

[571] G.Z. Papadopoulos, *Experimental Assessment of Traffic Generators,* M. SC. UNIVERSITY CARLOS III of MADRID, 2012.

[572] R.B. Adamson, S. Gallavan, L. O'Ferrall, E. Klinker, J. Macker, V. Park, B. Phan, J. Zwiebel i L. Adamson, „Networks and Communication System Branch Multi-Generator (MGEN)," [Online]. Available: http://www.nrl.navy. mil/itd/ncs/products/mgen.

[573] S. Avallone, S. Guadagno, D. Emma, A. Pescap`e i G. Ventre, „D-ITG Distributed Internet Traffic Generator," w *QEST ,04 Proceedings of the The Quantitative Evaluation of Systems, First International Conference*, 2004.

[574] S. Brander i T. Alexander, „Benchmarking Methodology for WirelessLAN Devices," 2004. [Online]. Available: http://tools.ietf.org/html/draft-alexander-wlan-meth-00.

[575] S. Brandt, Metody statystyczne i obliczeniowe analizy danych, Warszawa: PWN, 1976.

[576] H. Schulzrinne, S. Casner, R. Frederick i V. Jacobson, „RTP: A Transport Protocol for Real-Time Applications," IETF, 1996.

[577] J. Crane, „Metageek Chanalyzer and Wi-Spy User Guide," [Online]. Available: https://support.metageek.com/hc/en-us/articles/201872824-Chanalyzer-Wi-Spy-User- Guide. [Data uzyskania dostępu: 15 07 2016].

[578] K. Kałuża i J. Szymański, *Analiza środowiska radiokomunikacyjnego systemów pracujących w paśmie ISM 2.4 GHz na przykładzie sieci WLAN IEEE 802.11,* Tarnów, 2016.

[579] A. Pieprzycki i W. Ludwin, „Ocena anomalii w sieciach WLAN standardu IEEE 802.11," *Przegląd Telekomunikacyjny,* nr 6, 2016.

[580] A. Pieprzycki i P. Świętojański, „Ocena wpływu techniki CSMA/CA na liczbę i rozmieszczenie punktów dostępu w sieci WLAN standardu IEEE 802.11b/g," w *KKRRiT*, Kraków, 2010.

[581] R. Mantenga, „Fast, accurate algorithm for numerical simulation of Levy stable stochastic process," *Phys. Rev. E,* tom 5, nr 49, pp. 4677–4683, 1994.

[582] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press (2nd Edition, MIT Press), 1992.

[583] D. Karaboga, „AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMISATION," Erciyes University, Engineering Faculty Computer Engineering Department, Kayseri/Türkiye, 2005.

[584] H. Shah-Hosseini, „Otsu's criterion-based multilevel thresholding by a nature-inspired metaheuristic called Galaxy-based Search Algorithm," w *Third World Congress on Nature & Biologically Inspired Computing, NaBIC 2011*, Salamanca, Spain, 2011.

[585] H.A. Shehadeh, „HSSOGSA," Matlab FileExchange Mathworks, 2024.

AKADEMIA TARNOWSKA

ISBN 978-83-976333-9-1