Zastosowanie algorytmów rojowych w zadaniu planowania sieci WLAN

Application of rogue algorithms in the WLAN planning task

Adam Pieprzycki[a,*], Wiesław Ludwin[b]

[a]*Państwowa Wyższa Szkoła Zawodowa w Tarnowie, ul. Mickiewicza 8, 33-100 Tarnów, Polska*
[b]*Akademia Górniczo-Hutnicza w Krakowie, Al. Mickiewicza 30, 30-059 Kraków, Poland*
*Corresponding author: a_pieprzycki@pwsztar.edu.pl

Streszczenie

Celem artykułu jest sprawdzenie i porównanie rojowych metod optymalizacji w zadaniu planowania wewnątrzbudynkowej sieci łączności bezprzewodowej (WLAN).

W tym celu, w procesie poszukiwania ekstremum funkcji kryterialnej, która jest wskaźnikiem optymalizacyjnych, zastosowano sześć algorytmów rojowych: sztucznej kolonii pszczół, nietoperza, pszczeli, kukułki, świetlika, optymalizacji rojem cząstek (ptasi).

Słowa kluczowe: optymalizacja, algorytmy optymalizacji, planowanie, WLAN, optymalizacja nieliniowa, algorytmy rojowe

Abstract

The aim of this article is to examine and compare swarm optimization methods in the task of planning indoor wireless networks (WLAN).

For this purpose, in the process of searching for the extremum of the criterion function, which is an optimization indicator, six swarm algorithms were used: artificial bees colony, bat, bee, cuckoo, firefly, particle swarm (bird).

Key words: Artificial Bee Colony ABC, Bat Algorithm BA, Bee Algorithm BeA, Cuckoo Search CS, Firefly Algorithm FA, Particle Swarm Optimization PSO, WLAN planning, non–linear optimization

Wstęp

Obecnie znanych jest ponad 100 algorytmów wykorzystujących zjawiska zachowań występujących w świecie roślin i zwierząt.

Algorytmy rojowe

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

Mechanizm samoorganizacji polega na tworzeniu wzorców globalnej struktury na podstawie interakcji niższego poziomu, takich jak [1]:

– sprzężenia zwrotne lokalnych interakcji, a wśród nich sprzężenia pozytywne – powodujące wzmocnienie aktywności i wynikające z realizacji prostych reguł behawioralnych 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. Akcja wykonana przez osobnika 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 jako:

$$F_{fit} = \begin{cases} \frac{1}{1+F_c}, \text{dla} F_c \geq 0 \\ 1 + |F_c|, \text{dla } F_c < 0 \end{cases} \quad (1)$$

– sprawdzenie czy spełnione jest kryterium stopu, a jeżeli nie jest to następuje:
  • identyfikacja sąsiedztwa bieżących rozwiązań,
  wybór najlepszych rozwiązań z sąsiedztwa,
  • akceptacja rozwiązań kandydujących lub ich odrzucenie,
  • utworzenie nowej populacji rozwiązań;
– przedstawienie najlepszych rozwiązań.

Charakterystyka wybranych algorytmów rojowych

Dla potrzeb niniejszej pracy, w procesie poszukiwania ekstremum funkcji kryterialnej za pomocą algorytmów rojowych, posłużono się algorytmem [2]:

– sztucznej kolonii pszczół ABC (*Artificial Bee Colony*),
– nietoperza BA (*Bat Algorithm*),
– pszczelim BeA (*Bee Algorithm*),
– kukułki CS (*Cukko Serach*),
– świetlika FA (*Firefly Algorithm*)
– optymalizacji rojem cząstek PSO (*Particle Swarm Optimization*), nazywanym też ptasim.

Pierwszy z rozważanych algorytmów rojowych, **algorytm pszczeli BeA** umożliwia przeszukiwanie przestrzeni rozwiązań problemu wyrażonego funkcją kryterialną (F_c) w sposób naśladujący zdobywanie nektaru przez rój pszczół.

Prace nad symulacją roju pszczół były prowadzone już na przełomie lat 70. i 80. XX wieku [3, 4]. Obecnie istnieje duża liczba modyfikacji i rozszerzeń podstawowej wersji tego algorytmu, w tym np., 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 obszaru i odnalezienia miejsc zasobnych w kwiaty. Po powrocie do ula pszczoły, podczas tańca pszczelego, przekazują informację innym pszczołom o swoim najlepszym odkryciu. Podczas wykonywania wywijanego tańca pszczelego (*waggle dance*) przekazana zostaje informacja o: lokalizacji znalezionego źródła pożywienia, czyli kierunku oraz odległości od ula oraz jakości źródła (nektaru). Do najlepszych miejsc (źródeł pożywienia) zostają 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 etapów:

Etap 1: Losowej inicjalizacji rozwiązań początkowych – pszczół zwiadowców (N_{sb}).

Etap 2: Obliczeniu wartości funkcji kryterialnej (F_c) – rozwiązań początkowych dla całej populacji.

Etap 3: Dopóki niespełnione jest kryterium stopu (zadana maksymalna liczba iteracji N_{iter}^{max}) to należy:

- wybrać (N_{ss}) spośród N_{sb} odwiedzonych miejsc,
- zrekrutować pszczoły (N_{esb}) do najlepszych (N_{es}) miejsc,
- wyznaczyć wartość funkcji celu (F_c),
- wybrać najlepszą pszczołę w danym miejscu (najlepsze lokalne rozwiązanie),
- przypisać pozostałe pszczoły do losowych poszukiwań,
- obliczyć dla każdej z pszczół wartość funkcji dopasowania (wz. 1).

Etap 4: Jeżeli kryterium stopu zostanie spełnione (liczba iteracji osiągnie N_{iter}^{max}), to wybierane jest najlepsze rozwiązanie.

Algorytm **sztucznej kolonii pszczół ABC** po raz pierwszy został zaimplementowany przez D. Karaboga [6], a zainspirowany został inteligentnym zachowaniem pszczół miodnych. W algorytmie tym, w populacji pszczół możemy wyróżnić trzy typy pszczół: zwiadowców, obserwatorów oraz robotnice.

W algorytmie ABC „sztuczne pszczoły" poruszają się w trójwymiarowej przestrzeni poszukiwań, a pszczoły pracujące lub obserwujące 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}) [7].

Pseudokod algorytmu ABC można opisać następującymi etapami:

Etap 1: Losowe wyznaczenie (N_s) początkowych źródeł nektaru $\overrightarrow{x_i} \in (1, \ldots, N_s)$, dla pszczół zbierających (robotnic).

Etap 2: Wykonanie iteracji aż do spełnienia warunku stopu (osiągnięcia N_{iter}^{max}):

- wysłanie pszczół pracujących (zbierających nektar) do zapamiętanych miejsc pożywienia oraz wyznaczenie ilości nektaru,
- obliczenie wartości prawdopodobieństw dla „reklamowanych miejsc z pożywieniem", na podstawie których będą uaktualnione pozycje preferowane przez pszczoły obserwatorki:

$$p_i = \frac{F_{fit}(\overrightarrow{x_i})}{\sum_{j=1}^{N_s} F_{fit}(\overrightarrow{x_j})}, \tag{2}$$

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

$$v_{ij} = x_{ij} + (rand[1,-1]) \cdot (x_{ij} - x_{kj}),$$
$$k = \lfloor (rand[0,1] \cdot N_S) \rfloor + 1, j \epsilon [1, \ldots, N_D], \tag{3}$$

Gdzie: N_D – wymiar problemu.

- zaprzestanie eksploatacji źródeł porzuconych,
- wysłanie pszczół zwiadowców w celu odkrycia w sposób losowy nowych źródeł pożywienia (nektaru),

$$x_{ij} = x_j^{min} + \left(x_j^{min} - x_j^{min}\right| \cdot rand[0,1]), \quad (4)$$

- zapamiętanie najlepszego znalezionego do tej pory źródła pożywienia.

Etap 3: Przedstawienie otrzymanego rozwiązania zadania optymalizacji.

Algorytm optymalizacji rojem cząstek PSO, zwany też **algorytmem ptasim**, został zaproponowany w 1995 roku przez J. Kennedy'ego i R. Eberharta [8, 9]. Algorytm ten bazuje na zachowaniu całej populacji, w której istnieje możliwość komunikowania się między osobnikami (cząstkami) i dzielenia się informacjami. W algorytmie PSO każda cząstka ma określone położenie i prędkość. W przypadku odnalezienia lepszego rozwiązania, cząstki przemieszczają się do nowych miejsc–położeń, poszukując optimum i zmieniają kierunek. Każda cząstka zna swoich sąsiadów, pamięta swoje najlepsze położenia oraz położenie swoich sąsiadów, a także wartości funkcji kryterialnej ($\boldsymbol{F}_c$) dla swojego położenia i położenia sąsiadów.

Algorytm PSO opiera swoje działanie na [10]:

Etap 1: Losowej inicjalizacji położenia i prędkości startowych cząstek.

Etap 2: Ocenie położenia cząstek za pomocą funkcji kryterialnej/dopasowania (wz. 1).

Etap 3: Porównaniu zachowania każdej cząstki z jej najlepszym dotychczasowym zachowaniem (aktualizacja informacji), wyłonienie lidera roju.

Etap 4: Uaktualnieniu prędkości każdej cząstki w każdym kroku, wyrażonej jako:

$$v_i^m = \omega v_i^{m-1} + c_1 r_1 (p_i^{m-1} - x_i^{m-1}) + c_2 r_2 (p_d^{m-1} - x_i^{m-1}), \quad (5)$$

gdzie:

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

Etap 5: Uaktualnieniu położenia każdej cząstki w kroku (*m*), zdefiniowanym jako:

$$x_i^m = x_i^{m-1} + v_i^m. \quad (6)$$

W pochodzącym z 2007 roku, **algorytmie świetlika FA** [10] opracowanym przez Xin-She Yanga z Uniwersytetu Cambridge, rozwiązanie zadania optymalizacji oparto na różnicy intensywności światła, która jest proporcjonalna do wartości funkcji kryterialnej ($\boldsymbol{F}_c$). Każdy jaśniejszy świetlik przyciąga do siebie pozostałe osobniki, co pozwala na skuteczne badanie przestrzeni przeszukiwań.

Algorytm świetlika FA opiera się na trzech założeniach:

- niezależności od płci osobnika – wszystkie świetliki mogą się wzajemnie przyciągać i są równie atrakcyjne,
- atrakcyjności proporcjonalnej do jasności świecenia, która maleje wraz z odległością między świetlikami – jeżeli wszystkie świetliki są tak samo atrakcyjne to poruszają się w sposób losowy,
- intensywności świecenia – określonej przez wartość funkcji kryterialnej $\boldsymbol{F}_c$.

Ruch świetlika (położenie w kolejnym kroku – *m*) można zdefiniować pojęciami: eksploatacji – podczas którego świetlik stara się zbliżyć do najjaśniejszego osobnika i eksploracji – czyli losowego błądzenia.

Ruch *i*-tego świetlika określa formuła:

$$x_i^m = x_i^{m-1} + \beta_0 e^{-\gamma d_{ij}^2}\left(x_j^{m-1} - x_i^{m-1}\right) + \alpha_{fa}\left(rand[0,1] - \frac{1}{2}\right), \quad (7)$$

gdzie:

$\boldsymbol{x_i^{m-1}}$ – bieżące położenie świetlika *i*,
d_{ij} – odległość między świetlikami *i* oraz *j*,
β_0 – parametr określający atrakcyjność świetlika,
m – numer rozpatrywanego kroku,

α_{fa} – parametr losowości przemieszczenia świetlika z przedziału [0, 1].

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

Algorytm kukułki CS można opisać, tzw. pseudokodem [1, 10], uwzględniającym:

Etap 1: Losowe wygenerowanie początkowej populacji gniazd N_{nest}.

Etap 2: Wykonanie iteracji dopóki nie jest spełniony warunek stopu (osiągnięcia $\boldsymbol{N_{iter}^{max}}$):

wybrać losowo i – tę kukułkę, wygenerować rozwiązanie, np. przez lot Lévy'ego:

$$x_i^m = x_i^{m-1} + \alpha_{CS} Levy, \qquad (8)$$

gdzie:

m– numer kroku (iteracji),

x_i – rozwiązanie dla kukułki i,

α_{cs} – współczynnik skalujący powiązany z rozmiarem problemu,

$Levy$ – losowy krok o długości wyznaczonej na podstawie rozkładu Lévy'ego,

oraz obliczyć dla niej funkcję kryterialną $\boldsymbol{F_c^i}$,

- gdy spełniona jest nierówność: $\boldsymbol{F_c^i > F_c^j}$, należy zastąpić zawartość gniazda j nowym rozwiązaniem,
- porzucić część gorszych gniazd (z prawdopodobieństwem $\boldsymbol{p_a}$) i zastąpić je nowymi gniazdami,
- posortować osobniki i zapamiętać i przekazać najlepsze rozwiązania do kolejnych iteracji.

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

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

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

Na kolejne etapy algorytmu nietoperza BA [11] składają się:

Etap 1: Losowe wygenerowanie początkowej populacji nietoperzy N_{bat} i ich prędkości poruszania się. Zainicjowanie: częstotliwości, $\boldsymbol{r_i}$, $\boldsymbol{A_i}$.

Etap 2: Dopóki nie jest spełniony warunek stopu (osiągnięcie $\boldsymbol{N_{iter}^{max}}$), generowane są, przez dostosowanie częstotliwości, uaktualnienie prędkości i lokalizacji/rozwiązania, nowe rozwiązania. Położenie i-tego nietoperza określa formuła:

$$x_i^{m+1} = x_i^m + v_i^m + (x_i^m - x_*)[f_{min} + (f_{max} - f_{min})\beta_{BA}], \qquad (9)$$

gdzie:

x_i^m – bieżące położenie nietoperza i,

v_i^m – prędkość nietoperza i,

β_{BA} – losowa wartość z przedziału [0,1],

f_{min}, f_{max} – minimalna i maksymalna częstotliwość,

x_* – bieżące najlepsze rozwiązanie.

Jeżeli losowa wartość jest większa od , to należy:

- wybrać rozwiązanie spośród najlepszych rozwiązań,
- wygenerować lokalne rozwiązanie naokoło wybranego najlepszego rozwiązania.

Wygenerować nowe rozwiązania przez losowy lot.

Jeżeli losowa wartość jest mniejsza od $\boldsymbol{A_i}$ i spełniona jest nierówność: $\boldsymbol{F_c(x) < F_c(x_*)}$ należy:

- zaakceptować nowe rozwiązanie,
- zwiększyć $\boldsymbol{r_i}$ oraz zmniejszyć $\boldsymbol{A_i}$.

Etap 3: Przeglądniecie, ocenienie nietoperzy oraz odszukanie bieżącego najlepszego rozwiązania $\boldsymbol{x_*}$.

Materiały i Metody

Znanym sposobem analizy algorytmów jest użycie różnych funkcji testujących, np.: de Jong, Rosenbrock, Rastrigin, Easom, Salomon czy różnych

instancji Taillarda [11].

W artykule, zaprezentowane algorytmy rojowe zastosowaniu w rozwiązaniu zadania optymalizacji sieci WLAN.

W zadaniu planowania liczby punktów dostępu AP (*Access Point*) do sieci WLAN (*Wireless Local Area Network*), ich wzajemnego położenia zarówno względem siebie, jak i stacji ST (*station*), którymi posługują się użytkownicy oraz dobór parametrów pracy tych punktów (np.: mocy wyjściowej nadajnika, numeru kanału radiowego, parametrów techniki dostępu do łącza radiowego, rozmiaru pakietu) są wynikiem wyznaczenia optimum przyjętej funkcji kryterialnej.

Zastosowana funkcja kryterialna realizuje koncepcję podziału zasięgu radiowego punktu dostępu AP na obszary, w których transmisja odbywa się z szybkościami zależnymi od SINR (*Signal to Interference plus Noise Ratio*) i uwzględnia przepustowości osiągane przez poszczególne stacje ST

Tabela 1. Dane opisujące sieć WLAN oraz zadanie optymalizacji

Sieć WLAN	
Liczba stacji *ST* N_{ST}	20
Liczba punktów *AP* N_{AP}	3
Poziom czułości odbiornika S_{thr} /szybkość transmisji M_{TR}	-75 dBm / 54 Mbit/s
Sprzęt komputerowy	
Komputer PC	CPU Intel Core i5-3450 3,1 [GHz] / Win 7 64b
Metody i algorytmy optymalizacji	
Kryterium stopu – liczba iteracji	N_{iter}^{max} =300 dla BeA oraz 1000
Wymiar problemu	N_D=N_{AP} · 2
ABC	
Liczba pszczół,	N_{pop}=N_s50
Górna granica współczynnika przyspieszenia	a_{ABC}={0,1 0,5 0,9 1,0}
BA	
Liczba nietoperzy	N_{bat} = 50
Szybkość emisji impulsów *Pulse rate*	r_i={0,5 0,7 1}
Głośność	A_i={0 0,5 0,7}
BeA	
Liczbę pszczół skautów	N_{sb}=30, 50
Liczbę: wybranych miejsc i pszczół do wybranych miejsc	N_{ss} = ⌊0,5N_{sb}⌋, N_{ssb} =⌊0,5N_{sb}⌋
Liczbę elitarnych miejsc i pszczół do elitarnych miejsc	N_{es} = ⌊0,4N_{ss}⌋, N_{esb} =2N_{ssb}
Promień sąsiedztwa	$0{,}1(X_{max} - X_{min})$
CS	
Liczba gniazd	N_{nest} = 50
Prawdopodobieństwo wykrycia jaja podrzuconego przez kukułkę	p_a = {0,25, 0,30, 0,35, 0,40, 0,45, 0,46, 0,47, 0,48, 0,49, 0,50, 0,55}

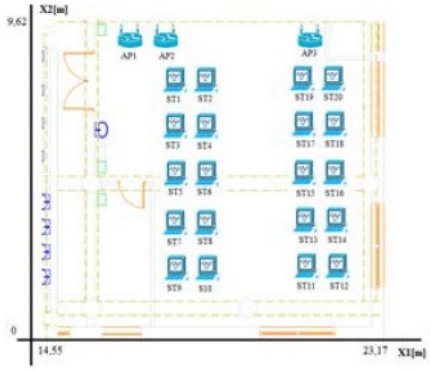
FA	
Liczba świetlików	N_{ff}=50
Parametr losowości	α_{ff} = {0,2 0,5 1}
Współczynnik „atrakcyjności" odniesienia	β_0={0,1 0,2 1}
Współczynnik absorpcji	γ ={0,01 0,1 1 10}
PSO	
Liczebność „roju"	N_{pso}=50
Współczynnik inercji	ω = 1
Współczynnik poznawczy *Cognition Coefficient*	c_1 = {1,0 1,5 1,6 1,7 1,8 1,9 2,0)
Współczynnik społeczny *Social Coefficient*	c_2 = {1,0 1,5 2,0 2,1 2,2 2,3 2,5 3,0)

wymieniające dane z punktami AP sieci WLAN. Analogicznie do wskaźnika wydajności sieci zdefiniowanego w pracy [12], w procesie planowania sieci WLAN przyjęto funkcje kryterialną:

$$F_C = \sum_{j=1}^{N_{AP}} \sum_{k=1}^{L_{TRj}} S_k \cdot \left(d_{jk}^2 - d_{j(k-1)}^2\right), \qquad (6)$$

gdzie: L_{TRj} – liczba klas ruchu dla *j*-tego punktu AP, S_k – przepustowość dla *k*-tej klasy (szybkości transmisji), – odległość między najdalszą stacją należącą do *k*-tej klasy ruchu a *j*-tym punktem AP.

Rozwiązanie zadania optymalizacji polegało na określeniu maksymalnej wartości funkcji kryterialnej wyrażonej wzorem (10).



Rysunek 1. Analizowany obszar działania sieci WLAN o 20 stacjach ST i trzech punktach AP

W poszukiwaniu ekstremum funkcji kryterialnej wykorzystano sześć algorytmów stadnych. Rozważono różne znane z literatury wartości parametrów sterujących dla zastosowanych algorytmów, które opracowano i zaadaptowano na potrzeby programu obliczeniowego opierając się na pracach dla: ABC [13], BA [10, 14], BeA [13], CS [15], FA [16], PSO [17].

Wyniki i Dyskusja

Dla analizowanej funkcji kryterialnej F_c, zapisanej wzorem (10), poszukiwano rozwiązania o największej wartości. Wartość średnią wyznaczono na podstawie 10 powtórzeń dla każdego parametru analizowanego algorytmu.

Tabela 2. Wyniki uzyskane za pomocą algorytmu sztucznej kolonii pszczół ABC

Parametr (a_{ABC})	$\overline{F_c} \pm \sigma$	F_c^{max}	Nr iteracji N_{iter} ostatniej zmiany F_c	T_{opt} [s]
0,1	2198 ± 108	2400	702	85385
0,5	2046 ± 108	2191	455	78297
0,9	1996 ± 111	2166	333	83115
1,0	2160 ± 182	**2599**	338	75517

Tabela 3. Wyniki uzyskane za pomocą algorytmu nietoperza BA

Parametry (r_i A_i)	$\overline{F_c} \pm \sigma$	F_c^{max}	Nr iteracji N_{iter} ostatniej zmiany F_c	T_{opt} [s]
(0,5 0,5)	1566 ± 559	2346	999	42078
(0,7 0,7)	1285 ± 598	2188	999	43677
(1,0 0,0)	1537 ± 653	**2438**	999	43452

Tabela 4. Wyniki uzyskane za pomocą algorytmu pszczelego BeA

Parametry (N_{sb} N_{iter}^{max})	$\overline{F_c} \pm \sigma$	F_c^{max}	Nr iteracji N_{iter} ostatniej zmiany F_c	T_{opt} [s]
(30 300)	2417 ± 563	**3699**	286	78806
(50 1000)	2643 ± 421	3495	330	808930

Tabela 5. Wyniki uzyskane za pomocą algorytmu kukułki CS

Parametr (p_a)	$\overline{F_c} \pm \sigma$	F_c^{max}	Nr iteracji N_{iter} ostatniej zmiany F_c	T_{opt} [s]
(0,25)	3826 ± 191	4201	916	84275
(0,3)	3319 ± 433	3787	920	77123
(0,35)	3488 ± 492	4237	962	78008
(0,4)	3613 ± 328	3787	486	76331
(0,45)	3825 ± 142	4230	868	78574
(0,46)	3477 ± 412	3786	996	77585
(0,47)	3779 ± 386	4295	683	83242
(0,48)	3579 ± 342	3786	671	90479
(0,49)	3352 ± 443	3786	920	88594
(0,50)	3770 ± 558	**4391**	855	67933
(0,55)	3587 ± 326	4137	664	78386

Tabela 6. Wyniki uzyskane za pomocą algorytmu świetlika FA

Parametry ($\alpha_{ff}\ \beta_0\ \gamma$)	$\overline{F_c} \pm \sigma$	F_c^{max}	Nr iteracji N_{iter} ostatniej zmiany F_c	T_{opt} [s]
(0,2 1,0 1,0)	2523 ± 246	2900	990	39134
(0,5 1,0 0,01)	2603 ± 184	2929	994	38462
(0,5 1,0 0,1)	2672 ± 96	2913	996	41369
(0,5 1,0 1,0)	2556 ± 271	2896	997	38390
(0,5 1,0 10)	2517 ± 167	2636	998	40175
(0,8 1,0 1,0)	2477 ± 213	3013	997	42252
(1,0 1,0 0,01)	2703 ± 351	3741	998	39893
(1,0 1,0 0,1)	2617 ± 272	3201	995	39857
1,0 1,0 1,0)	2627 ± 529	**3783**	997	48951

Tabela 7. Wyniki uzyskane za pomocą algorytmu optymalizacji rojem cząstek PSO

Parametry ($c_1\ c_2$)	$\overline{F_c} \pm \sigma$	F_c^{max}	Nr iteracji N_{iter} ostatniej zmiany F_c	T_{opt} [s]
(1,0 1,0)	977 ± 350	1554	5	38740
(1,0 3,0)	2009 ± 444	2569	300	45177
(1,5 1,5)	1744 ± 461	2286	16	39949
(1,5 2,5)	2221 ± 421	3324	935	39144
(1,6 2,4)	2234 ± 216	2592	991	43982
(1,7 2,3)	2036 ± 505	3020	768	43591
(1,8 2,2)	2616 ± 526	**3783**	956	43376
(1,9 2,1)	2298 ± 74	2415	819	42708
(2,0 2,0)	2186 ± 639	2911	273	31236

Tabela 8. Test kolejności par Wilcoxona dla najlepszych rozwiązań algorytmów i 10 ważnych prób

Algorytm 1 i 2	Wartość testu T	Poziom p (prawdopodobieństwa testowego)
CS i FA	3,00000	0,012516
CS i PSO	1,00000	0,006911
CS i BeA	0,00000	0,005062
CS i ABC	0,00000	0,005062
CS i BA	0,00000	0,005062

Tabela 9. Zestawienie wyników obliczeń numerycznych dla najlepszych rozwiązań

Algorytm	Współrzędne trzech AP [m]		N_{ST}	F_c^{max}
	X_{AP}	Y_{AP}		[(Mbit/s) m^2]
	21,11	0,01	9	
BA	22,44	0,92	10	2438
	21,63	0,44	1	
	16,45	9,60	14	
ABC	14,96	8,24	1	2599
	14,87	8,08	5	
	14,55	9,52	1	
BeA	14,60	9,46	11	3699
	21,83	9,52	8	
	14,55	9,62	1	
PSO	21,83	9,62	8	3783
	14,70	9,57	11	
	21,83	9,56	8	
FA	14,55	9,62	1	3783
	14,61	9,58	11	
	15,71	9,62	**1**	
CS	16,28	9,62	**1**	4391
	16,33	9,57	**18**	

Podsumowanie

Spośród sześciu zastosowanych algorytmów, największą wartość funkcji kryterialnej (10) uzyskano dla algorytmu CS i prawdopodobieństwa na poziomie 0,5, natomiast największą wartość średnią uzyskano dla na poziomie 0,25 (Tab. 5).

Dla każdego rozważonego przypadku z różną wartością algorytmu CS maksymalna wartość funkcji kryterialnej była co najmniej równa największym wartościom funkcji kryterialnej pozostałych algorytmów.

Z zestawienia wartości funkcji kryterialnych zaprezentowanych w tabelach 2-7 wynika, że w wielu przypadkach ostatnie zmiany funkcji kryterialnych były bliskie przyjętej wartości maksymalnej. Mimo dużej liczby iteracji dla każdego analizowanego przypadku, można przypuszczać, że zwiększenie mogło by prowadzić do uzyskania większych wartości np. dla algorytmu BA czy FA (Tab. 3 i 6).

Z przedstawionych w tabelach 2-7 wyników obliczeń *Fc* można zauważyć wpływ parametrów sterujących na osiągane wyniki. Dla algorytmu pszczelego BeA (Tab. 4) czas obliczeń T_{opt} był o rząd wielkości większy niż dla ABC i CS, a prawie 20 krotnie większy niż dla pozostałych algorytmów.

Zastosowanie algorytmu świetlika FA, optymalizacji rojem cząstek PSO czy algorytmem pszczelim BeA w przedstawionym zadaniu optymalizacji dało wynik gorszy (Tab. 9), z innym niż dla CS, ale analogicznym sposobem przydzielenia stacji ST do trzech punktów AP. Całkowicie inne

rozwiązania z inną liczbę stacji ST przyporządkowanych do punktów AP uzyskano przy zastosowaniu algorytmów ABC oraz BA (Tab. 9).

Zbadano pary wyników obliczeń najlepszych rozwiązań z CS oraz pozostałych pięciu zastosowanych algorytmów testem Wilcoxona. Postawiono hipotezę zerową o braku istotnych różnic między wynikami uzyskanymi za pomocą zastosowanych algorytmów, dla najlepszych przypadków, w których uzyskano największą wartość funkcji kryterialnej. Z otrzymanych wyników (Tab. 8) należy wnioskować, że hipoteza ta powinna zostać odrzucona dla wszystkich przedstawionych par algorytmów przy poziomie istotności $p<0,05$.

Dla zaprezentowanego zadania optymalizacji, przy zastosowaniu algorytmu CS uzyskano wartości funkcji kryterialnej, które były większe niże te uzyskane z zastosowaniem pozostałych analizowanych rojowych algorytmów optymalizacji. Dla rozwiązania zadania optymalizacji z użyciem CS uzyskano także całkowicie inny przydział stacji ST do trzech punktów AP.

Podziękowania

Przedstawione badania powstały przy użyciu zasobów obliczeniowych PLGrid.

Literatura

1. J. Kwiecień, Algorytmy stadne w rozwiązaniu wybranych zagadnień optymalizacji dyskretnej i kombinatorycznej, AGH, Kraków, 2015.
2. B. Filipowicz i J. Kwiecień, *Pomiary Automatyka, Robotyka*, 2011, **12**, 152-157.
3. A. Migacz i R. Tadeusiewicz, Model rodziny pszczelej, Akademia Medyczna, Kraków, 1979.
4. A. Migacz i R. Tadeusiewicz, *System Science*, 1983, **3**, 83-95.
5. D. Karaboga, An idea based on honey bee swarm for numerical optimization, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
6. D. Karaboga i B. Akay, Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimization, W: IPROMS 2009 Innovative Production Machines and Systems Virtual Conference, Cardiff, UK, 2009.
7. J. Kennedy i R. Eberhart, Particle Swarm Optimization, IEEE International Conference on Neural Networks, 1995.
8. R. Eberhart, Y. Shi i J. Kennedy, Swarm Intelligence, Morgan Kaufman, San Francisco, 2001.
9. K. Trojanowski, Metaheurystyki praktycznie, wyd. 2, WIT, Warszawa, 2008.
10. X.-S. Yang, Nature-inspired optimisation algorithms, Elsevier, 2014.
11. E. D. Taillard, *European Journal of Operationals Research*, 1993, **64**, 278-285.
12. Methodology for Testing Wireless LAN Performance, http://www.super-g.com/collateral/atheros_benchmark_whitepaper.pdf [02.02.2017].
13. S. Mostapha Kalami Heris, Implementation of Standard Bees Algorithm in MATLAB, Yarpiz, http://yarpiz.com/315/ypea115-bees-algorithm [02.02.2017].
14. X.-S. Yang, Bat algorithm, https://www.mathworks.com/matlabcentral/fileexchange/37582--bat-algorithm--demo- [02.02.2017].
15. X. S. Yang, Cuckoo Search (CS) Algorithm, http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search--cs--algorithm [02. 02.2017].
16. X. S. Yang, Firefly Algorithm for Constrained Optimization, www.mathworks.com/matlabcentral/fileexchange/29693-firefly-algorithm [02.02.2017].
17. Particle swarm analysis, http://www.codeforge.com/article/259219 [02.02.2017].