Algorytmy stadne w problemach optymalizacji

Bogusław Filipowicz, Joanna Kwiecień

AGH Akademia Górniczo-Hutnicza w Krakowie, Wydział EAliE, Katedra Automa

Streszczenie: W artykule przedstawiono zastosowanie algorytmu optymalizacji rojem cząstek, algorytmu pszczelego i algorytmu świetlika do wyznaczenia optymalnego rozwiązania wybranych testowych funkcji ciągłych. Przedstawiono i porównano wyniki badań dla funkcji Rosenbrocka, Rastrigina i de Jonga.

Słowa kluczowe: optymalizacja nieliniowa, algorytm PSO, algorytm pszczeli, algorytm świetlika

1. Wprowadzenie

Problemy optymalizacji spotykane są na każdym kroku, zarówno w nauce, w przemyśle, administracji, jak i codziennym życiu, gdy chcemy zminimalizować koszty czy też zmaksymalizować zyski. W wielu współczesnych problemach optymalizacji konieczne jest stosowanie takich algorytmów, które łatwo dostosowują się do ograniczeń niezależnie od rozmiarów przestrzeni rozwiązań i pozwalają jednocześnie skrócić czas obliczeń.

W ostatnich latach intensywnie rozwijają się algorytmy stadne, które naśladując zachowania istniejące w świecie zwierząt czy roślin, dostarczają bardzo wydajnych procedur optymalizacyjnych. Najnowszymi tego typu algorytmami są: algorytmy optymalizacji rojem cząstek PSO (ang. *particle swarm optimization*), algorytmy pszczele BA (ang. *bee algorithms*) oraz algorytm świetlika FA (ang. *firefly algorithm*). Metody te należą do klasy dynamicznie rozwijających się technik optymalizacji.

2. Algorytm optymalizacji rojem cząstek

Algorytm optymalizacji rojem cząstek opracowany został przez J. Kennedy'ego i R. Eberharta w 1995 r. Algorytm ten bazuje na zachowaniu całej populacji, w której istnieje możliwość komunikowania się między osobnikami-cząstkami i dzielenia się informacjami, przy czym każda cząstka ma określone położenie i prędkość. Cząstki przemieszczają się do nowych położeń, poszukując optimum i zmieniają kierunek, jeśli lepsze rozwiązanie zostanie znalezione. Każda cząstka zna swoich sąsiadów, pamięta swoje najlepsze położenia i położenie swoich sąsiadów, wartość funkcji ewaluacyjnej dla swojego położenia i położenia sąsiadów [1, 3, 5, 12].

Parametry i-tej cząstki, takie jak położenie oraz prędkość, można przedstawić w postaci wektorów $x_i = [x_{i1}, x_{i2}, ..., x_{id}]$ oraz $v_i = [v_{i1}, v_{i2}, ..., v_{id}]$, gdzie d oznacza wymiar przestrzeni. Najlepsza pozycja cząstki $p_i = [p_{i1}, p_{i2}, ..., p_{id}]$ odpowiada najlepszej uzyskanej dotychczas wartości funkcji celu, zaś najlepsza pozycja cząstki-przywódcy w całym roju określona jest jako $p_d = [p_{d1}, p_{d2}, ..., p_{dd}]$. Algorytm optymalizacji rojem cząstek jest algorytmem iteracyjnym. W każdej iteracji uaktualniona zostaje prędkość i kierunek przemieszczania się cząstek. Zadana liczba iteracji jest często stosowanym kryterium zakończenia algorytmu.

Algorytm optymalizacji rojem cząstek można przedstawić w kilku etapach [1, 3, 5, 12]:

Etap 1: Losowa inicjalizacja pozycji i prędkości początkowych cząstek.

Etap 2: Ocena położenia cząstek za pomocą funkcji dopasowania.

Etap 3: Porównanie zachowania każdej cząstki z jej najlepszym dotychczasowym zachowaniem.

Etap 4: Uaktualnienie prędkości każdej cząstki w każdym kroku k:

$$v_i(k) = \omega v_i(k-1) + c_1 r_1[p_i(k-1) - x_i(k-1)] + \\ + c_2 r_2[p_d(k-1) - x_i(k-1)] \quad (1)$$

gdzie ω oznacza współczynnik inercji ruchu cząstki, c_1 – parametr określający zaufanie do kierunku swojego najlepszego położenia, c_2 to wskaźnik zaufania do położeń swoich sąsiadów, r_1 oraz r_2 to losowe liczby o rozkładzie równomiernym w przedziale [0, 1].

Etap 5: Uaktualnienie położenia każdej cząstki:

$$x_i(k) = x_i(k-1) + v_i(k) \quad (2)$$

Badając wpływ sąsiadów na zachowanie cząstek, należy uwzględnić sposoby określenia najlepszej dotychczas znalezionej pozycji sąsiadów. Istnieją głównie dwa podejścia: *gbest* (ang. *global best particle*) i *lbest* (ang. *local best particle*). W wersji *gbest* uwzględnia się najlepsze znalezione rozwiązanie z całej populacji cząstek. W *lbest* uwzględnione jest najlepsze znalezione rozwiązanie z cząstek z sąsiedztwa i-tej cząstki [12].

3. Algorytmy pszczele

Do klasy algorytmów stadnych należą również algorytmy pszczele. Prace dotyczące matematycznego modelowania i komputerowej symulacji roju pszczelego powstawały już na przełomie lat 70. i 80. ubiegłego wieku [8–10]. W literaturze można znaleźć wiele rodzajów algorytmów pszczelich. Jednym z nich jest algorytm pszczół miodnych HBA (ang. *honey bee algorithm*), który został po raz pierwszy sformułowany w 2004 r. C. Tovey i S. Nakrani przedstawili ten algorytm jako metodę rozwiązania problemu alokacji komputerów. W latach 2004–2005 X.S. Yang przedstawił wirtualny algorytm pszczeli VBA (ang. *virtual bee algorithm*) do rozwiązania problemów optymalizacji ciągłej i dyskretnej. B. Basturk oraz D. Karaboga zaprezentowali

algorytm kolonii pszczół ABC (ang. *artificial bee colony*) do numerycznej optymalizacji funkcji w 2006 roku [13].

Algorytmy pszczele pozwalają przeszukać przestrzeń rozwiązań danego problemu w oparciu o proces zdobywania nektaru przez rój pszczół. Początkowo grupa pszczół-zwiadowców zostaje losowo rozesłana celem przeszukania obszarów zasobnych w kwiatostany. Po powrocie do ula powiadamiają one pozostałe pszczoły o swoim najlepszym odkryciu. W trakcie wykonywania tańca pszczelego przekazywana jest informacja o jakości, kierunku i odległości pożywienia od ula. Do najlepszych miejsc rozsyłane są pozostałe pszczoły, które rozpoczynają zbiory nektaru. Im zasobniejsze źródło, tym więcej pszczół dowiaduje się o tym miejscu. Powracające pszczoły przekazują następnym informacje na temat eksploatowanego obszaru, aby mogły podążyć za śladem wybranej pszczoły-zwiadowcy [2, 3, 11].

Ogólną strukturę algorytmu pszczelego można przedstawić następująco [2–4, 11]:

Etap 1: Losowa inicjalizacja n rozwiązań początkowych (pszczół zwiadowców).

Etap 2: Obliczenie funkcji celu dla całej populacji.

Etap 3: Dopóki niespełnione jest kryterium stopu (zadana liczba iteracji) należy:
- wybrać m sąsiedztw do przeszukiwania,
- zrekrutować pszczoły do wybranych miejsc (liczba pszczół *nep* do najlepszych e miejsc),
- wyliczyć funkcję celu,
- wybrać najlepszą pszczołę w danym miejscu (najlepsze lokalne rozwiązanie),
- przypisać pozostałe pszczoły do losowych poszukiwań i wyliczyć ich funkcje dopasowania.

Etap 4: Jeśli spełnione kryterium stopu – wyznaczyć najlepsze rozwiązania.

4. Algorytm świetlika

W 2007 r. Xin-She Yang z Uniwersytetu Cambridge, opracował algorytm w oparciu o zachowanie świetlików. Ich bioluminescencyjne sygnały służą m.in. jako element zalotów, metody przyciągania ofiar lub jako sygnał ostrzegawczy. W algorytmie tym rozwiązanie problemu optymalizacji oparte jest na różnicy w intensywności światła, która jest proporcjonalna do wartości funkcji celu. Każdy jaśniejszy świetlik przyciąga do siebie pozostałe, co pozwala na efektywne badanie przestrzeni przeszukiwań [7, 13, 14].

Algorytm świetlika zakłada trzy reguły [13]:
- wszystkie świetliki są dla siebie atrakcyjne bez względu na płeć,
- atrakcyjność świetlików jest proporcjonalna do ich jasności świecenia, maleje wraz ze wzrostem odległości między świetlikami; jeśli wszystkie świetliki są tak samo atrakcyjne to poruszają się losowo,
- intensywność światła określona jest przez wartość funkcji celu.

Każdy świetlik ma swoją atrakcyjność opisaną przez funkcję odległości między dwoma dowolnymi świetlikami:

$$\beta(r) = \beta_0 e^{-\gamma r^m}, \qquad m \geq 1, \tag{3}$$

gdzie β_0 oznacza atrakcyjność w $r = 0$ zaś γ jest współczynnikiem absorpcji światła.

Odległość między dwoma świetlikami i oraz j w pozycjach x_i i x_j jest określona jako:

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2}, \tag{4}$$

gdzie d oznacza liczbę wymiarów.

Ruch świetlika i jest określony przez następującą formułę:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(rand - \frac{1}{2}), \tag{5}$$

gdzie pierwszy składnik określa bieżącą pozycję świetlika i, drugi składnik określa atrakcyjność, trzeci składnik używany jest w przypadku losowego przemieszczania (*rand* jest liczbą losową z zakresu [0, 1], zaś $\alpha \in (0, 1)$). W większości przypadków $\beta_0 = 1$ i $\gamma = 1$.

Ogólna struktura algorytmu jest następująca [7, 13]:

Etap 1: Inicjalizacja parametrów algorytmu (n, β_0, γ, liczba iteracji, α).
Zdefiniowanie funkcji celu $f(x)$, $x = (x_1, ..., x_d)^T$.
Wygenerowanie początkowej populacji świetlików.
Intensywność światła i-tego świetlika I_i jest określona przez funkcję celu $f(x_i)$.

Etap 2: Dopóki nie jest spełniony warunek stopu (zadana liczba iteracji):
dla wszystkich n świetlików należy:
- jeśli $(I_j > I_i)$ to wykonać ruch świetlika i w kierunku świetlika j,
- wyznaczyć atrakcyjność,
- znaleźć nowe rozwiązanie i uaktualnić intensywność światła.

Ocena świetlików i znalezienie najlepszego.

Etap 3: Spełniony warunek stopu – wskazanie świetlika z najwyższą intensywnością światła, wizualizacja wyników.

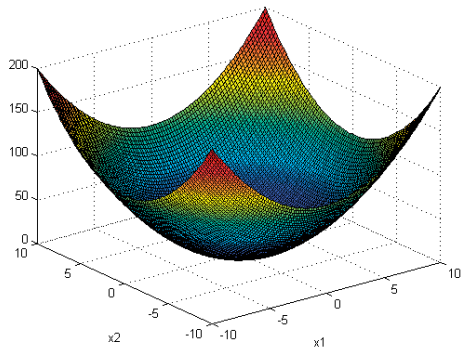
5. Eksperymenty

Przeprowadzono badania dotyczące zastosowania algorytmów PSO, BA i FA do optymalizacji wybranych funkcji testowych [4]. Wyniki prezentowane w pracy zostały ograniczone do funkcji dwuwymiarowych.

Pierwszą analizowaną funkcją jest n wymiarowa funkcja de Jonga (rys. 1), przedstawiona w postaci:

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{6}$$

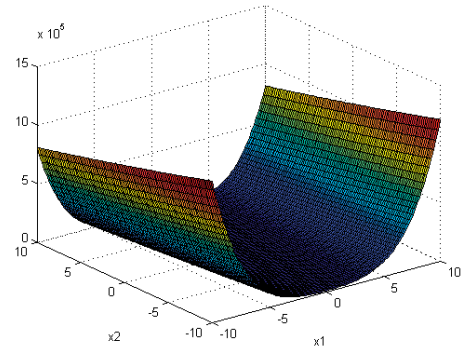
i posiadająca minimum $f_1(0,..,0) = 0$. Zakres zmiennych analizowanej funkcji de Jonga został ograniczony do przedziału [-10, 10].



Rys. 1. Dwuwymiarowa funkcja de Jonga
Fig. 1. 2D de Jong's function

Kolejna testowana funkcja to funkcja Rosenbrocka (rys. 2), z minimum globalnym $f_2(1,..,1) = 0$ w przedziale [-10, 10], którą można przedstawić jako:

$$f_2(x) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{7}$$

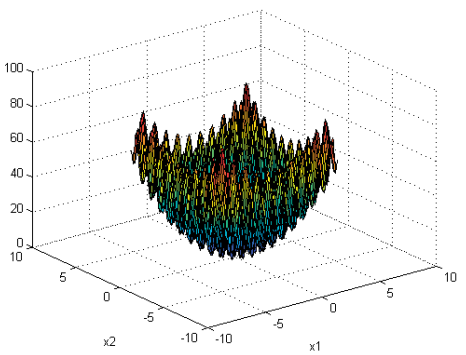


Rys. 2. Dwuwymiarowa funkcja Rosenbrocka
Fig. 2. 2D Rosenbrock's function

Ostatnia wybrana do badań funkcja Rastrigina (rys. 3), o zmiennych z przedziału [-5,12, 5,12], ma postać:

$$f_3(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{8}$$

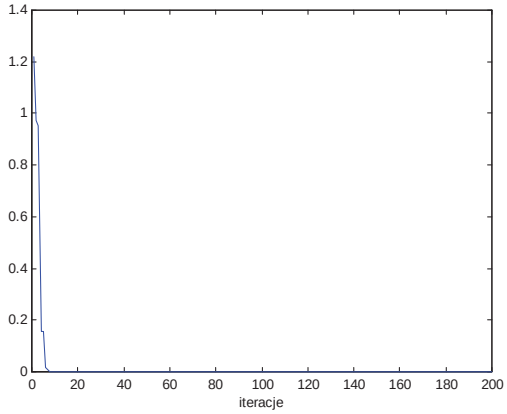
Ma ona minimum globalne $f_3(0,..,0) = 0$.



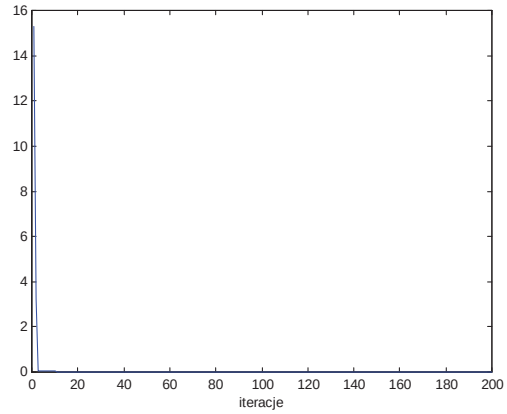
Rys. 3. Dwuwymiarowa funkcja Rastrigina
Fig. 3. 2D Rastrigin's function

Dla każdego algorytmu wykonano eksperymenty obliczeniowe w środowisku MATLAB 7.1, przy różnych ustawieniach parametrów, które wpływają na jakość rozwiązania i czas obliczeń. Wykorzystano również wartości parametrów podawane w literaturze [5, 7, 11]. Algorytm świetlika zaimplementowano bazując na pracy [15].

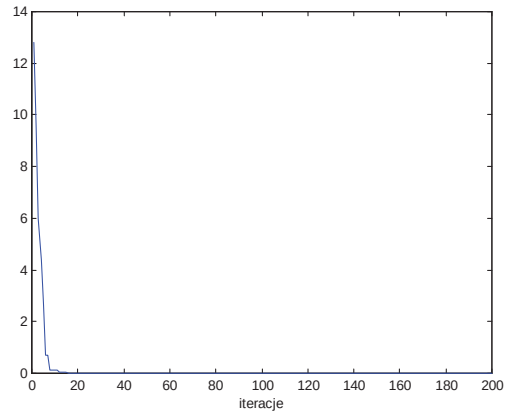
Na rys. 4 przedstawiono wykresy zależności wartości funkcji od liczby iteracji dla algorytmu PSO.



a) De Jong (x_1 = 0,19e-016, x_2 = 0,31e-016)



b) Rosenbrock (x_1 = 1, x_2 = 1)

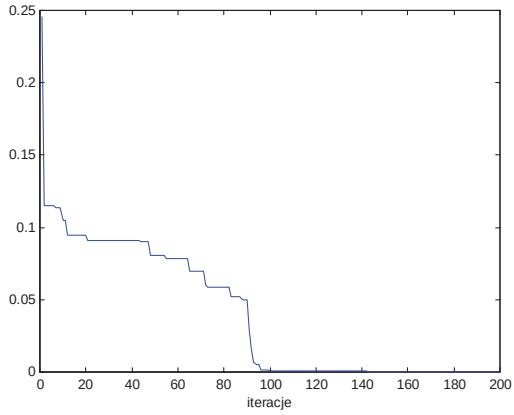


c) Rastrigin (x_1 = 0,26e-006, x_2 = 0,13e-006)

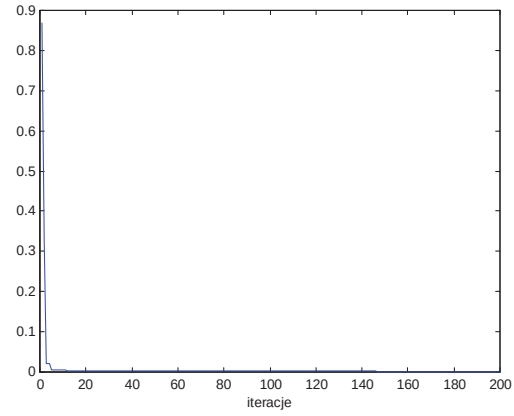
Rys. 4. Wykresy zależności wartości wybranych funkcji od liczby iteracji
Fig. 4. The relationship of value of selected functions with iterations

Przyjęto następujące ustawienia algorytmu PSO: populacja składa się z 40 cząstek, kryterium stopu – wykonanie 200 iteracji, $c_1 = c_2 = 2$, $\omega = 0{,}8$. Okazało się, że wprowadzenie większej liczby iteracji (500) nie miało większego

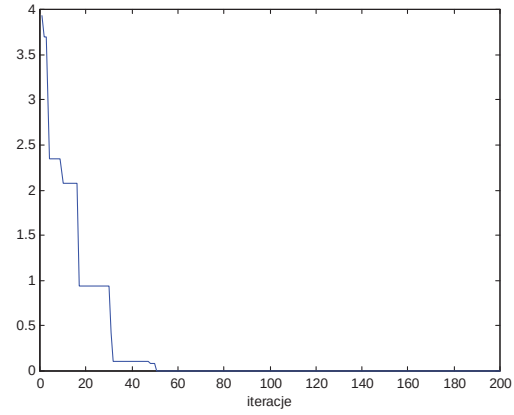
wpływu na poprawę rozwiązania. Dla większej liczby cząstek (100) uzyskiwano nieco lepsze rezultaty.



a) De Jong (x_1 = 0,14e-003, x_2 = -0,51e-003, f(x) = 0,0003e-003)



b) Rosenbrock (x_1 = 1,0136, x_2 = 1,025, f(x) = 0,0008)

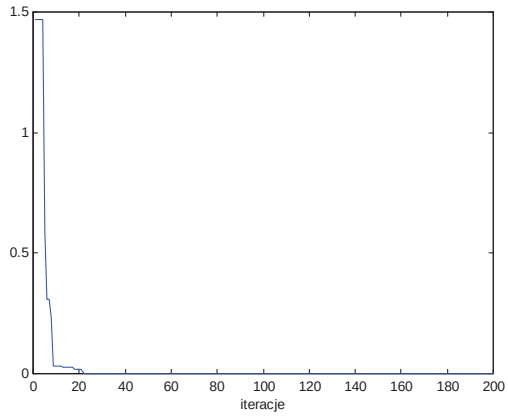


c) Rastrigin (x_1 = 0,0003, x_2 = 0,0019, f(x) = 0,0007)

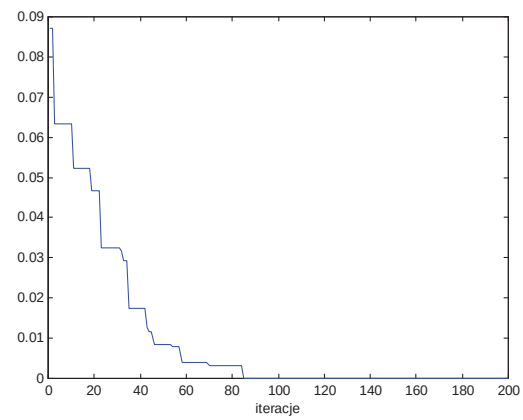
Rys. 5. Wykresy wartości wybranych funkcji od liczby iteracji
Fig. 5. The relationship of value of selected functions with iterations

Na rys. 5 przedstawiono wyniki dla algorytmów pszczelich, przy następujących ustawieniach: 40 pszczół zwiadowców, 5 sąsiedztw, 5 najlepszych miejsc, liczba pszczół wysłanych do najlepszych miejsc $nep = 25$, 200 iteracji. W wielu eksperymentach algorytm potrzebował ponad 100 iteracji na znalezienie minimum.

Dla algorytmu świetlika przyjęto następujące wartości parametrów: 40 świetlików, 200 iteracji, atrakcyjność $\beta_0 = 1$, współczynnik absorpcji światła $\gamma = 1$, $\alpha = 0{,}2$. Wykresy zależności wartości wybranych funkcji od liczby iteracji dla rozpatrywanego algorytmu przedstawiono na rys. 6.



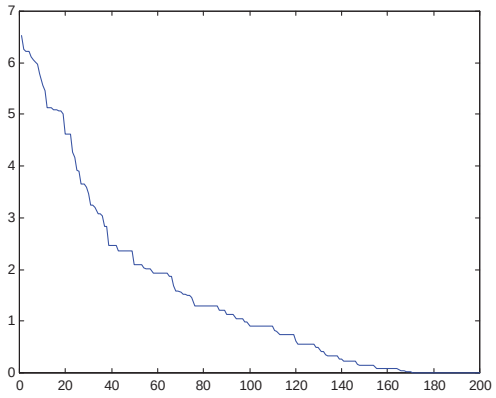
a) De Jong (x_1 = 0, x_2 = 0, f(x) = 0)



b) Rosenbrock (x_1 = 1,0023, x_2 = 1,0048, f(x) = 9,08e-006)

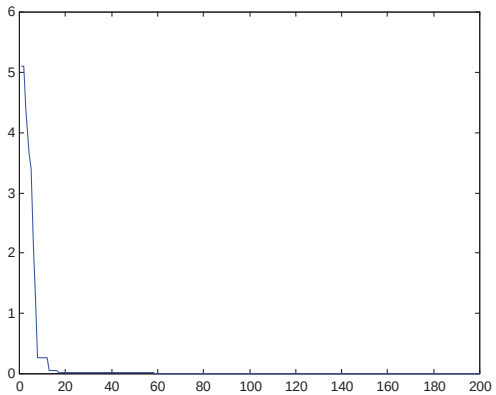
Rys. 6. Wykresy zależności wartości funkcji de Jonga i Rosenbrocka od liczby iteracji (α = 0,2, γ = 1)
Fig. 6. The relationship of value of de Jong and Rosenbrock functions with iterations (α = 0,2, γ = 1)

Dla funkcji de Jonga algorytm świetlika znajdował minimum już w 20 iteracji, zaś dla funkcji Rosenbrocka potrzebował więcej iteracji. W przypadku funkcji Rastrigina dla parametru $\alpha = 0{,}2$ nie udało się znaleźć wartości optymalnej Algorytm ten znajdował minima lokalne. Dla wartości $\alpha = 0{,}8$ uzyskano lepsze rezultaty (rys. 7). Podobnie w przypadku zwiększonego współczynnika absorpcji $\gamma = 10$, algorytm świetlika znajdował lepsze rozwiązanie (rys. 8), przy małej liczbie iteracji.



Rys. 7. Wykresy zależności wartości funkcji Rastrigina od liczby iteracji (α = 0,8, x_1 = -0,0015, x_2 = -0,0011, f(x) = 0,0007)

Fig. 7. The relationship of value of Rastrigin function with iterations (α = 0,8, x_1 = -0,0015, x_2 = -0,0011, f(x) = 0,0007)



Rys. 8. Wykresy zależności wartości funkcji Rastrigina od liczby iteracji (γ = 10, x_1 = -0,0041, x_2 = -0,0005, f(x) = 0,0033)

Fig. 8. The relationship of value of Rastrigin function with iterations (γ = 10, x_1 = -0,0041, x_2 = -0,0005, f(x) = 0,0033)

6. Podsumowanie

W artykule zaprezentowano zastosowanie trzech algorytmów stadnych do znalezienia optimum trzech funkcji testowych. Spośród rozważanych algorytmów najskuteczniejszymi okazały się algorytmy świetlika i PSO. Do znalezienia minimum globalnego algorytm pszczeli potrzebował najwięcej iteracji. Duże problemy pojawiły się w przypadku funkcji wielomodalnej Rastrigina. Podstawowym problemem rzutującym na funkcjonowanie algorytmów jest dobranie odpowiednich ustawień parametrów. W przypadku problemów o większych wymiarach algorytm świetlika daje najlepsze wyniki.

Obecnie w Polsce, jak i na świecie, prowadzone są liczne badania dotyczące zastosowania algorytmów stadnych do rozwiązania problemów istniejących w wielu dziedzinach [2, 3, 6, 11-14].

Bibliografia

1. Eberhart R., Shi Y., Kennedy J.: *Swarm Intelligence*. Morgan Kaufman, San Francisco, 2001.
2. Filipowicz B., Chmiel W., Kadłuczka P.: *Ukierunkowane przeszukiwanie przestrzeni rozwiązań w algorytmach rojowych*. „Automatyka", półrocznik AGH, 13(2), 2009.
3. Filipowicz B., Kwiecień J.: *Algorytmy stadne w optymalizacji problemów przydziału przy kwadratowym wskaźniku jakości* (*QAP*). „Automatyka", półrocznik AGH, 15(2), 2011.
4. Karaboga D., Akay B.: *Artificial Bee Colony* (*ABC*), *Harmony Search and Bees Algorithms on numerical optimization*. Artykuł dostępny na stronie: [http://conference.iproms.org/sites/conference.iproms.org/files/IPROMSABCv2.pdf], (28.08.2011).
5. Kennedy J., Eberhart R.: *Particle Swarm Optimization*. Materiały IEEE International Conference on Neural Networks, 4, 1942–1948, 1995.
6. Lewicki A., Tadeusiewicz R.: *An autocatalytic emergence swarm algorithm in the decision-making task of managing the process of creation of intellectual capital*. Human-Computer Systems Interaction. Backgrounds and Applications 2, Advances in Soft Computing, Springer-Verlag Co., Berlin, 2011, 173–188.
7. Łukasik S., Żak S.: *Firefly algorithm for continuous constrained optimization task*. Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems LNCS, 5796, 97–106, 2009.
8. Migacz A., Tadeusiewicz R.: *Model rodziny pszczelej*. Modelowanie Cybernetyczne Systemów Biologicznych, Akademia Medyczna, Kraków 1979, 133–144.
9. Migacz A., Tadeusiewicz R.: *The computer model of the bee colony*. System Science 3(9), 83–95, 1983.
10. Migacz A., Tadeusiewicz R.: *Komputerowy model zjawiska konkurencji rodzin pszczelich w terenie*. „Archiwum Automatyki i Telemechaniki", 26(2), 253–265, 1981.
11. Pham D.T., Ghanbarzadeh A., Koc E., Otri S., Rahim S., Zaidi M.: *The Bees Algorithm – A Novel Tool for Complex Optimisation Problems*. Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.
12. Trojanowski K.: *Metaheurystyki praktycznie*. Wydawnictwo WIT, Warszawa 2005.
13. Yang X.S.: *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.
14. Yang X.S.: *Firefly algorithms for multimodal optimization*. Stochastic Algorithms: Foundations and Applications, SAGA, Lecture Notes in Computer Sciences, 5792, 169–178, 2009.
15. [www.mathworks.com/matlabcentral/fileexchange/29693-firefly-algorithm]. ■

Swarm Algorithms in Optimization Problems

Abstract: This paper presents particle swarm optimization, bee algorithm and firefly algorithm, used for optimal solution of selected continuous well-known functions. Results of these algorithms are compared to each other on Rosenbrock, Rastrigin and de Jong functions.

Keywords: non-linear optimization, particle swarm optimization, bee algorithm, firefly algorithm

prof. zw. dr hab. inż. Bogusław Filipowicz

Jest pracownikiem Wydziału EAIiE AGH w Krakowie w Katedrze Automatyki i pełni funkcję kierownika Laboratorium Badań Operacyjnych i Systemowych. Jest specjalistą w dziedzinie automatyki i robotyki oraz badań operacyjnych i systemowych. Jest autorem 11 książek oraz ponad 120 publikacji w czasopismach i materiałach konferencji krajowych i zagranicznych. Jego zainteresowania badawcze skupiają się wokół modelowania i optymalizacji systemów i sieci kolejkowych, metod i algorytmów optymalizacji kombinatorycznej, a także zastosowań badań operacyjnych do wspomagania zarządzania złożonymi kompleksami operacji. Bogusław Filipowicz jest członkiem Komisji Elektrotechniki, Informatyki i Automatyki PAN, Oddział w Krakowie oraz sekretarzem Komisji i Sekcji Informatyki i Robotyki PAN. Jest laureatem indywidualnej Nagrody Ministra Nauki i Szkolnictwa Wyższego za monografię pt. „Modele stochastyczne w badaniach operacyjnych" wydaną przez WNT w Warszawie oraz Nagrody II-go stopnia im. Prof. W. Taklińskiego. Został również nagrodzony Medalem Komisji Edukacji Narodowej za działalność dydaktyczną oraz Złotym Medalem za długoletnią służbę.

Pod jego kierunkiem naukowym obroniło prace dyplomowe magisterskie i inżynierskie ponad 130 dyplomantów w kraju i zagranicą. Był promotorem w 5 przewodach doktorskich zakończonych obroną w kraju i 2-ch III-cyklu w Algierii.
e-mail: *filip@ia.agh.edu.pl*

dr inż. Joanna Kwiecień

Jest adiunktem w Katedrze Automatyki AGH w Krakowie. Stopień doktora nauk technicznych w zakresie Automatyki i Robotyki uzyskała w 2004 roku. Jej główne zainteresowania naukowe koncentrują się w obszarze badań operacyjnych i systemowych oraz algorytmów sztucznej inteligencji. Jest autorką kilkudziesięciu artykułów o zasięgu krajowym i międzynarodowym.
e-mail: *kwiecien@ia.agh.edu.pl*