Adam SŁOWIK, Marek POPŁAWSKI

Politechnika Koszalińska, Wydział Elektroniki i Informatyki

doi:10.15199/48.2016.08.03

Optymalizacja procesu wiercenia otworów w elektronicznych płytach drukowanych przy użyciu algorytmu roju cząstek

Streszczenie. *W niniejszej pracy przedstawiono optymalizację procesu wiercenia otworów w elektronicznych płytach drukowanych. Do realizacji tego zadania zastosowano algorytm optymalizacji rojem cząstek w wersji dostosowanej do optymalizacji problemów kombinatorycznych. Opracowany algorytm przetestowano przy użyciu ogólnie dostępnych danych benchmarkowych z biblioteki VLSI Data Set. Biblioteka ta zawiera dane odnośnie przykładowych elektronicznych płyt drukowanych. Otrzymane wyniki porównano z wynikami otrzymanymi przy użyciu standardowego algorytmu rojowego przystosowanego do optymalizacji problemów o dyskretnych dziedzinach. Trasa ramienia wiercącego uzyskana przy użyciu proponowanego algorytmu jest krótsza od trasy uzyskanej standardowym algorytmem roju dla dziedzin dyskretnych.*

Abstract. *In this paper, the optimization of the drilling holes process in the electronic printed circuit boards is presented. The particle swarm optimization algorithm in the version dedicated to the optimization of the combinatorial problems is applied for this task realization. The algorithm elaborated in this paper was tested with the use of global accessible benchmark data sets with the VLSI Data Set library. This library contains the data of the exemplary electronic printed boards. The results obtained using proposed algorithm were compared with the results obtained using standard particle swarm optimization algorithm in the version dedicated for optimization of the problems with discrete domains. The route of drilling arm obtained using proposed algorithm was shorter than the route of drilling arm obtained using standard particle swarm optimization algorithm for discrete domains. (**Optimization of the drilling holes process in the electronic printed circuit boards using particle swarm optimization algorithm**).*

Słowa kluczowe: algorytm roju, inteligencja roju, optymalizacja, układy elektroniczne, elektroniczne płyty drukowane.
Keywords: particle swarm optimization algorithm, swarm intelligence, optimization, electronic circuits, printed circuit boards.

Wstęp

Proces wiercenia otworów w elektronicznych płytach drukowanych (PCB) jest obecnie przemysłowo wykonywany przy użyciu maszyn CNC (Computer Numerical Controlled). W procesie tym dąży się do minimalizacji czasu wiercenia otworów w pojedynczej płycie PCB. Oczywiście wiele maszyn nie wybiera optymalnej drogi dla ramienia wiercącego w procesie nawiercania otworów. Z tego względu do optymalizacji długości tej drogi coraz częściej stosuje się algorytmy optymalizacji globalnej bazujące na naturze. Wśród tych algorytmów możemy wymienić: algorytmy genetyczne [1, 2], algorytmy mrówkowe [3, 4], algorytmy roju [5], algorytmy ewolucji różnicowej [6, 7]. Algorytmy te dość często są stosowane do rozwiązywania problemów inżynierskich w tym problemów związanych z elektroniką. Jako przykład można podać prace [18-22]. Każdy z tych algorytmów obecnie jest szeroko rozwijany w wyniku czego powstaje wiele ich różnych modyfikacji [23-26].

Przegląd literatury odnośnie algorytmów bazujących na naturze w aspekcie ich zastosowania do problemu wiercenia otworów w elektronicznych płytach drukowanych PCB można rozpocząć do roku 1996. W tym roku Kolahan i Liang opublikowali pracę [8] w której proponowali zastosowanie algorytmu tabu search do rozwiązania problemu wiercenia otworów w płytach PCB (modyfikacja tego algorytmu została opublikowana w pracy [9]). W roku 2004 Clerc w pracy [10] zaproponował zastosowanie algorytmu roju do rozwiązania tego samego problemu. Prawdopodobnie praca [10] jest jedną z pierwszych w której wykorzystano standardowy algorytm roju do optymalizacji trasy w procesie wiercenia otworów w płytach PCB. W pracy [11] Sigl i Mayer zaproponowali wykorzystanie algorytmu ewolucyjnego z dodatkowymi heurystykami do rozwiązania tego samego problemu. W artykule [12] Quedri przedstawił wykorzystanie algorytmu genetycznego do optymalizacji procesu wiercenia otworów w płytach PCB, a w pracy [13] Ghaiebi i Solimanpur zastosowali algorytm mrówkowy do optymalizacji procesu wiercenia otworów o różnych średnicach. W roku 2006 w pracy [14] Zhou odkrył, że standardowy algorytm roju nie może dać dobrych wyników w problemie wiercenia otworów w płytach PCB, a następnie w pracy [15] przedstawił modyfikację algorytmu roju w celu jego zastosowania do problemu wiercenia otworów w płytach PCB. Modyfikacja ta polegała między innymi na wprowadzeniu nowych operatorów genetycznych. Operatory te miały na celu zapewnienie spełnienia ograniczeń wynikających z konieczności stosowania kodowania rozwiązań w formie liczb całkowitych. Ponieważ pierwotnie algorytm roju został zaprojektowany do optymalizacji problemów o ciągłych dziedzinach. Natomiast problem optymalizacji procesu wiercenia otworów w płytach PCB jest problemem kombinatorycznym.

W niniejszej pracy przedstawiono wykorzystanie algorytmu C3DPSO [16] do optymalizacji procesu wiercenia otworów w płytach PCB. Algorytm C3DPSO jest specjalizowanym algorytmem roju dostosowanym do rozwiązywania problemów kombinatorycznych. Opracowaną metodę przetestowano przy użyciu ogólnie dostępnych danych benchmarkowych z biblioteki VLSI Data Set. Biblioteka ta zawiera dane odnośnie 102 elektronicznych płytek drukowanych PCB. Dla płyt PCB z tej biblioteki liczba otworów zawiera się w przedziale między 131 a 744710. Wyniki otrzymane przy użyciu proponowanej metody (którą nazwano PSOPCB – Particle Swarm Optimization for Printed Circuit Boards) porównano z wynikami otrzymanymi przy użyciu standardowego algorytmu rojowego przystosowanego do optymalizacji problemów o dyskretnych dziedzinach.

Optymalizacji wiercenia otworów w płytach PCB

W procesie optymalizacji procesu wiercenia otworów w płytach PCB dąży się do minimalizacji długości drogi *R* jaką ma przebyć ramię robota z narzędziem wiercącym. W przypadku, gdy przyjmiemy założenia, że po pierwsze – przejście ramienia robota jest możliwe z każdego miejsca wiercenia otworu do innego wcześniej zdefiniowanego miejsca wiercenia otworu na płycie PCB, po drugie – po wywierceniu wszystkich otworów na płycie PCB ramię robota z narzędziem wiercącym wraca do otworu, w którym wiercenie było rozpoczynane, wówczas problem można zamodelować jako problem TSP (Travelling Salesman Problem). Natomiast długość drogi R jest sumą odcinków pomiędzy poszczególnymi otworami jaką pokonuje w

danym procesie wiercenia ramię robota. Odległość $dist$ pomiędzy dwoma otworami A i B o współrzędnych odpowiednio (x_A, y_A) i (x_B, y_B) określano przy użyciu metryki Euklidesowej zgodnie z zależnością:

$$dist(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \quad (1)$$

Podstawowy algorytm optymalizacji rojem cząstek

W podstawowej wersji algorytmu optymalizacji rojem cząstek (PSO) dąży się do optymalizacji problemu w D-wymiarowej ciągłej przestrzeni rozwiązań. Każda i-ta cząstka z populacji N cząstek posiada swój D-elementowy wektor położenia x_i (bieżące rozwiązanie problemu) oraz D-elementowy wektor prędkości v_i (zmiana rozwiązania w następnym kroku). Dodatkowo jakość każdej cząstki oceniana jest przez funkcję $f(.)$ oraz każda i-ta cząstka posiada jeszcze wektor $pbest_i$ w którym zapisane jest jej najlepsze położenie (rozwiązanie) z dotychczas znalezionych. W całym roju występuje jeszcze jedna zmienna $gbest$ w której zapisane jest najlepsze położenie (rozwiązanie) cząstki jakie udało się znaleźć do obecnej chwili. Generalnie standardowy algorytm roju w wersji globalnej (szukamy pojedynczego najlepszego rozwiązania) dla zadania minimalizacji funkcji celu $f(.)$ może być przedstawiony w formie pseudokodu:

Losowo utwórz N cząstek w roju
For i:=1 to N do
  Dokonaj oceny $f(x_i)$
End For
While (warunek stopu nie jest spełniony) do
  For i:=1 to N do
    if $f(x_i)<f(pbest_i)$ then $pbest_i:=x_i$
    if $f(x_i)<f(gbest)$ then $gbest:=x_i$
    Uaktualnij wektory:
      v_i (według zależności (2))
      x_i (według zależności (3))
    Dokonaj oceny $f(x_i)$
  End For
End While

Aktualizacja wektorów v_i oraz x_i dla k+1 iteracji algorytmu odbywa się następująco:

$$v_i^{k+1} = w * v_i^k + c_1 * rand(\,) * \left(pbest_i^k - x_i^k\right) + c_2 * rand(\,) * \left(gbest^k - x_i^k\right) \quad (2)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (3)$$

gdzie: - w jest współczynnikiem inercji, - c_1 i c_2 są to tzw. współczynniki uczenia, które odpowiednio determinują czy i-ta cząstka ma być bardziej „przyciągana" do jej najlepszej wcześniej znalezionej pozycji $pbest_i$ (współczynnik uczenia c_1) czy do najlepszej globalnej pozycji *gbest* z całego roju (współczynnik c_2). Wartości tych współczynników zawierają się przeważnie w przedziale [0; 1].

- *rand()* jest funkcją zwracającą losową wartość z przedziału [0; 1).

Proponowana metoda PSOPCB

Proponowana metoda PSOPCB bazuje na algorytmie C3PDSO [16]. Generalna koncepcja metody PSOPCB bazuje na pseudokodzie algorytmu który przedstawiono w poprzednim podrozdziale. W tym podrozdziale opisano różnice w metodzie PSOPCB w stosunku do standardowego algorytmu optymalizacji rojem cząstek.

Załóżmy, że problem wiercenia otworów w płycie PCB składa się z M otworów. Rozwiązanie zapisane w i-tej cząstce w wektorze x_i jest zbiorem krawędzi po których może poruszać się ramię robota z narzędziem wiercącym. Niech $X(x,y)$ reprezentuje krawędź (x, y) łączącą punkt x z punktem y z prawdopodobieństwem X. Dodanie krawędzi do trasy polega na wylosowaniu liczby S z przedziału [0; 1). Jeśli wartość S jest mniejsza równa X wówczas dana krawędź jest dodana do trasy. Należy zauważyć, że w niniejszej pracy rozpatrujemy symetryczny graf połączeń pomiędzy punktami na płycie PCB. Tzn. krawędź połączenia punktu x z punktem y, jest taka sama jak krawędź połączenia punktu y z punktem x. Wektor prędkości jest zbiorem elementów $X(x,y)$.

Dodatkowo w algorytmie zdefiniowano operację odjęcia dwóch wektorów położenia od siebie, operację mnożenia wektorów położenia przez skalar oraz operację dodania dwóch wektorów prędkości.

Efektem odjęcia wektora x_1 od x_2 jest zbiór krawędzi, które istnieją w wektorze x_1, ale nie ma ich w wektorze x_2. Dodatkowo prawdopodobieństwo 1 jest dodawane do tych krawędzi.

Dla przykładu:
x_1={(1,2), (2,3), (3,4), (4,5), (5,1)},
x_2={(1,2), (2,4), (4,3), (3,5), (5,1)},
wówczas x_1–x_2={1(2,3), 1(4,5)}.

Efektem mnożenia wartości rzeczywistej A przez zbiór krawędzi jest nadal zbiór krawędzi, ze zmodyfikowanymi wartościami prawdopodobieństwa.

Dla przykładu:
x_1={B(2,3), C(3,5)},
wówczas A*x_1={A*B(2,3), A*C(3,5)}.

Efektem dodania wektorów v_1 i v_2 jest zbiór złożony ze wszystkich krawędzi występujących w v_1 i v_2. Dodatkowo w algorytmie [16] założono, że w wynikowym wektorze prędkości każdy punkt może występować maksymalnie tylko 4 razy, aby zapobiec nadmiernemu rozrostowi wektora prędkości. Dodatkowo w stosunku do standardowego algorytmu PSO zmianie uległa formuła wyznaczania wektora prędkości cząstki (2) oraz formuła wyznaczania nowego położenia cząstki (3). Formuły te przyjmują odpowiednio postać opisaną zależnościami (4) i (5).

$$v_i^{k+1} = c_2 * rand(\,) * \left(gbest_i^k - x_i^k\right) + c_1 * rand(\,) * \left(pbest_i^k - x_i^k\right) + w * v_i^k \quad (4)$$

$$x_i^{k+1} = v_i^{k+1} \oplus c_3 * rand(\,) * x_i^k \quad (5)$$

Wartość parametru mutacji c_3 zawiera się w przedziale [0; 1]. Więcej dyskusji na temat parametru c_3 można znaleźć w pracy [16]. Tworzenie nowej trasy dla ramienia robota z narzędziem wiercącym zawiera się w trzech krokach. W pierwszym kroku konstruowane jest rozwiązanie x_i' bazując na prawdopodobieństwach zapisanych w wektorze prędkości. W drugim kroku z wektora położenia x_i wybierane są z określonym prawdopodobieństwem krawędzie w celu skompletowania rozwiązania x_i'. Jeśli te, dwa kroki nie utworzą nam całego

cyklu Hamiltona, wówczas do rozwiązania dodawane są brakujące krawędzie zgodnie z zasadą najbliższego sąsiedztwa. W wyjątkowej sytuacji, jeśli w dwóch pierwszych krokach zostanie utworzone rozwiązanie niedopuszczalne (np. zawierające 2 takie same punkty w drodze) wówczas, to rozwiązanie jest tracone i cały proces tworzenie nowego rozwiązania jest ponawiany.

Jeśli w zależności (4) i (5) wartości c_1, c_2 lub c_3 będą większe od 1 wówczas po operacji mnożenia może się zdarzyć, że wartość prawdopodobieństwa wyboru danej krawędzi do trasy będzie większa od 1. W takim przypadku wartość ta jest ustalana na 1. Oczywiście funkcją celu w rozpatrywanym problemie jest długość trasy jaką musi przebyć ramię wiertnicze pomiędzy poszczególnymi punktami na płycie PCB. Algorytm podczas swojej pracy dąży do minimalizacji długości trasy ramienia wiertniczego.

Jako kryterium zatrzymania algorytmu przyjęto określony przy użytkownika maksymalny czas obliczeniowy dla danego problemu.

Opis przeprowadzonych eksperymentów

Wyniki uzyskane przy użyciu opracowanej metody (PSOPCB) porównano z wynikami otrzymanymi przy użyciu klasycznego algorytmu roju przystosowanego do rozwiązywania problemów dyskretnych (DPSO) [17].

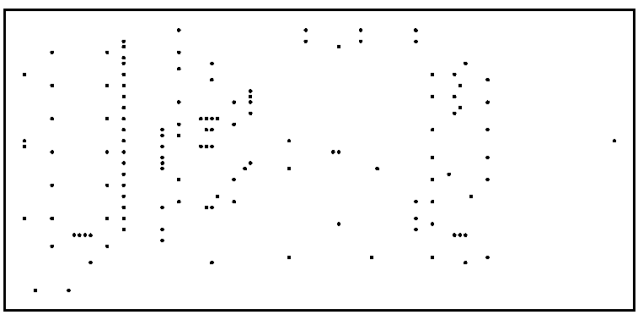
Jako benchmarki służące do porównania wyników uzyskanych przy użyciu metody PSOPCB z wynikami uzyskanymi przy użyciu metody DPSO wybrano losowo 6 płyt drukowanych PCB z biblioteki VLSI Data Set [29]. Wybrane płyty PCB posiadają zróżnicowaną liczbę otworów (od 131 do 411). Dodatkowo dla każdej topologii płyty PCB określono maksymalny czas obliczeniowy (C_{max}) identyczny dla każdego z testowanych algorytmów.

W tabeli 1 przedstawiono dane odnośnie wybranych problemów testowych łącznie z globalnie optymalnymi wartościami długości trasy ramienia wiercącego (*Optimum*).

Tabela 1. Dane dotyczące 6 problemów testowych wybranych losowo z biblioteki VLSI Data Set [29]

Nazwa topologii płyty PCB	Liczba otworów do wywiercenia	C_{max} [s]	*Optimum*
XQF131	131	7	564
XQG237	237	12	1019
PMA343	343	17	1368
BCL380	380	19	1621
PBL395	395	20	1281
PBK411	411	21	1343

Na rysunku 1 przedstawiono dla przykładu wygląd topologii płyty PCB o nazwie XQF131.



Rys. 1. Wygląd płyty PCB (XQF131) z zaznaczonymi miejscami (kropki) wiercenia otworów [27]

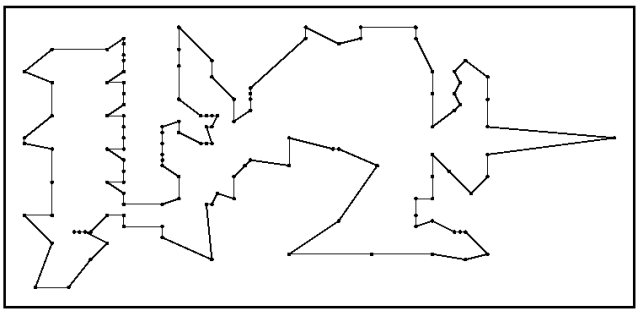
Optymalną trasę ramienia wiertniczego dla płyty PCB z rysunku 1 przedstawiono na rysunku 2.

W algorytmie DPSO oraz PSOPCB przyjęto następujące parametry:

- liczba cząstek N = 100,
- współczynnik c_1=0.7,
- współczynnik c_2=0.4,
- waga inercji w=0.5.

Dodatkowo w algorytmie PSOPCB przyjęto, że wartość parametru dodatkowego c_3=0.2. Każdy z algorytmów (PSOPCB i DPSO) uruchomiono 10-cio krotnie.

W tabeli 2 przedstawiono najlepsze wyniki (*Best*) uzyskane dla każdego testu z 10-ciu uruchomień każdego z algorytmów. Dodatkowo ukazano również wartość odchylenia (*Dev*) pomiędzy uzyskanym wynikiem (*Best*), a wartością *Optimum* z tabeli 1.



Rys. 2. Optymalna trasa ramienia wiertniczego dla płyty PCB z rysunku 1 [28]

Tabela 2. Porównanie najlepszych wyników uzyskanych metodą PSOPCB z wynikami uzyskanymi przy użyciu algorytmu DPSO

Nazwa topologii płyty PCB	PSOPCB		DPSO	
	Best	*Dev*	*Best*	*Dev*
XQF131	691	127	791	227
XQG237	1610	591	1981	962
PMA343	2571	1203	3224	1859
BCL380	2917	1296	4023	2402
PBL395	2399	1118	3201	1920
PBK411	2567	1224	3370	2027

W tabeli 3 przedstawiono średnie wartości (*Avg*) długości trasy ramienia wiercącego uzyskane po 10-krotnym uruchomieniu każdego z algorytmów (PSOPCB, DPSO) dla każdego z 6 wybranych problemów testowych.

Tabela 3. Porównanie średnich wyników uzyskanych metodą PSOPCB z wynikami uzyskanymi przy użyciu algorytmu DPSO

Nazwa topologii płyty PCB	PSOPCB	DPSO
	Avg	*Avg*
XQF131	1311.875	1168.125
XQG237	2587.0	3079.30
PMA343	7102.78	7376.0
BCL380	5490.55	7585.05
PBL395	5037.6	6090.71
PBK411	5674.8	6830.5

Podsumowanie

Z tabeli 2 można zauważyć, że wyniki uzyskiwane przy użyciu proponowanej metody PSOPCB są lepsze od wyników uzyskiwanych przy użyciu metody DPSO. Wartości odchylenia Dev wraz ze wzrostem wymiarowości problemu również wzrastają. Jest to zrozumiałe ponieważ algorytm ma coraz większą przestrzeń do przeszukania. Długość trasy ramienia wiercącego uzyskana przy użyciu proponowanego algorytmu PSOPCB jest od około 13% (topologia PCB – XQF131) do około 27% (topologia PCB - BCL380) krótsza od tej samej trasy uzyskanej w tym samym czasie przy użyciu algorytmu DPSO.

Z tabeli 3 widać, że wyniki uzyskiwane przy użyciu algorytmu PSOPCB są średnio lepsze od wyników uzyskanych przy użyciu metody DPSO w tym samym czasie obliczeniowym.

Autorzy pracy pragną podziękować Panu Piotrowi Dudzińskiemu za opracowanie aplikacji i przeprowadzenie testów opisanych w niniejszym artykule

Autorzy:
dr hab. inż. Adam Słowik, dr inż. Marek Popławski, Politechnika Koszalińska, Wydział Elektroniki i Informatyki, Katedra Inżynierii Komputerowej, ul. Śniadeckich 2, 75-453 Koszalin, E-mail: aslowik@ie.tu.koszalin.pl

LITERATURA

[1] Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs. Springer, Heidelberg (1992)
[2] Goldberg D.E., Genetic algorithms in search, optimization, and machine learning, Addison-Wesley Publishing Company Inc., New York (1989)
[3] Socha, K., Doringo, M., Ant colony optimization for continous domains, *European Journal of Operational Research,* 185 (2008), n.3, 1155-1173
[4] Dorigo M., Stutzle T., Ant Colony Optimization, *The MIT Press*, (2004)
[5] Kennedy J., Eberhart R.C., Shi Y., Swarm intelligence, San Francisco, Morgan Kaufmann Publishers, 2001
[6] Price K., An Introduction to Differential Evolution, In Corne D., Dorigo M., Glover F., (eds.), New Ideas in Optimization, McGraw-Hill, London, UK, (1999) 79-108
[7] Price K.V., Storn R.M., Lampinen J.A., Differential Evolution: A Practical Approach to Global Optimization, Springer 2005
[8] Kolahan F., Liang M., Tabu search approach to optimization of drilling operations, *Computers and Industrial Engineering*, 31 (1996), n.1-2, 371-374
[9] Kolahan F., Liang M. Optimization of hole-making operations: a tabu search approach, *International Journal of Machine Tools and Manufacture*, 2 (2000), n.40, 1735-1753
[10] Onwubolu G.C., Clerc M., Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization, *International Journal of Production Research*, 3 (2004), n.42, 473-491
[11] Sigl S., Mayer H.A. Hybrid evolutionary approaches to CNC drill route optimization, *Proceedings of Computational Intelligence for Modeling, Control and Automation*, (2005) 905-910
[12] Qudeiri J.A., Yamamoto H., Ramli R. Optimization of operation sequence in CNC machine tools using genetic algorithm, *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 1 (2007), n.2, 272-282
[13] Ghaiebi H., Solimanpur M. An ant algorithm for optimization of hole-making operations, *Computers and Industrial Engineering*, 2 (2007), n.52, 308-319
[14] Zhu G.Y., Drilling path optimization based on swarm intelligent algorithm, *Proceedings of IEEE International Conference on Robotics and Biomimetics*, (2006), 193-196
[15] Zhu G.Y., Zhang W.B. Drilling path optimization by the particle swarm optimization algorithm with global convergence characteristics, *International Journal of Production Research*, 46 (2008), n.8, 2299-2311
[16] Zhong Wen-Liang, Guangzhou Zhang Jun, Chen Wei-Neng N., A novel discrete particle swarm optimization to solve traveling salesman problem, IEEE Congres on Evolutionary Computation 2007, (2007), 3283-3287
[17] Clerc M., Discrete Particle Swarm Optimization Illustrated by the Travelling Salesman Problem, Technical Report, 29 (2000)
[18] Słowik A., Zastosowanie algorytmu ewolucyjnego do minimalizacji poboru mocy podczas testowania układów cyfrowych, Przegląd Elektrotechniczny, (2009), n.11, 153-155
[19] Słowik A., Hybrydowa metoda ewolucyjnej optymalizacji kombinacyjnych układów cyfrowych, Przegląd Elektrotechniczny, (2009), n.11, 156-159
[20] Słowik A., Application of Evolutionary Algorithm to Design of Minimal Phase Digital Filters with Non-Standard Amplitude Characteristics and Finite Bits Word Length, Bulletin of The Polish Academy of Science - Technical Science, 59 (2011), n.2, 125-135
[21] Słowik A., Białko M., Partitioning of VLSI Circuits on Subcircuits with Minimal Number of Connections Using Evolutionary Algorithm", 8th International Conference on Artificial Intelligence and Soft Computing, ICAISC 2006, Lecture Notes in Artificial Intelligence, 4029 (2006), 470-478
[22] Słowik A., Białko M., Evolutionary Design of Combinational Digital Circuits: State of the Art, Main Problems, and Future Trends, First International Conference on Information Technology, IT 2008, Gdansk, May 18-21, 2008, 209-212
[23] Segura C., Coello Coello C.A., Segredo E., Leon C., An Analysis of the Automatic Adaptation of the Crossover Rate in Differential Evolution, in 2014 IEEE Congress on Evolutionary Computation (CEC'2014), pp. 459-466, IEEE Press, Beijing, China, 6-11 July 2014, ISBN 978-1-4799-1488-3
[24] Cagnina L.C., Esquivel S.C., Coello Coello C.A., A Fast Particle Swarm Algorithm For Solving Smooth and Non-smooth Economic Dispatch Problems, Engineering Optimization, 43 (2011), n.5, 485--505
[25] Słowik A., Steering of Balance Between Exploration and Exploitation Properties of Evolutionary Algorithms - Mix Selection, 10th International Conference on Artificial Intelligence and Soft Computing, June 13-17, 2010, Zakopane, Poland, Lecture Notes in Artificial Intelligence, L. Rutkowski et al. (Eds.): ICAISC 2010, Part II, LNAI 6114, 213--220
[26] Segura C., Coello Coello C.A., Segredo E., León C., On the Adaptation of the Mutation Scale Factor in Differential Evolution, Optimization Letters, 9 (2015), n.1, 189-198
[27] http://www.math.uwaterloo.ca/tsp/vlsi/xqf131.points.html
[28] http://www.math.uwaterloo.ca/tsp/vlsi/xqf131.tour.html
[29] http://www.math.uwaterloo.ca/tsp/vlsi/