ALGORYTM PRZESZUKIWANIA Z ZABRONIENIAMI DLA DWUKRYTERIALNEGO PROBLEMU PRZEPŁYWOWEGO

Jarosław PEMPERA, Dominik ŻELAZNY

Streszczenie: Praca poświęcona jest problemowi przepływowemu z dwukryterialną funkcją celu optymalizacji. Rozważane są następujące funkcje kryterialne: suma czasów zakończenia zadań oraz czas zakończenia realizacji zadań. W pracy proponuje się nowy algorytm oparty na technice przeszukiwania z zabronieniami. Badania eksperymentalne przeprowadzone na literaturowych przykładach testowych jednoznacznie potwierdzają wysoką skuteczność zaproponowanego algorytmu.

Słowa kluczowe: problem przepływowy, optymalizacja wielokryterialna, przeszukiwanie z zabronieniami.

1. Wprowadzenie

Problem przepływowy, w szczególności permutacyjny problem przepływowy, jest obiektem badań naukowych od wielu dekad. Zdecydowana większość prac naukowych poświęcona jest optymalizacji jednokryterialnej. W rzeczywistych systemach wspomagania planowania produkcji najczęściej jest ona niewystarczająca. Dlatego w ostatnich latach pojawiają się prace dotyczące optymalizacji wielokryterialnej w takich systemach.

Ze względu na NP-trudny charakter problemu przepływowego wysiłki projektantów algorytmów optymalizacyjnych skoncentrowane są na metodach przeszukiwań przestrzeni rozwiązań. Wysoce efektywne algorytmy optymalizacyjne dla jednokryterialnego problemu przepływowego oparte są na takich metodach jak: przeszukiwanie z zabronieniami [1,2], symulowanego wyżarzania [3,4], przeszukiwania mrówkowego [5], przeszukiwania harmonicznego [6], przeszukiwania rojowego [7]. Wymienione algorytmy stanowią reprezentację szerokiej gamy algorytmów.

Wśród algorytmów optymalizacji wielokryterialnej, w tym algorytmów szeregowania zadań, do oceny poszczególnych rozwiązań wykorzystuje się podział na zadania zdominowane i niezdominowane (Pareto efektywne/optymalne). Zbiór punktów niezdominowanych nazywany jest frontem Pareto. Ze względu na swój schemat, działanie wielo-agentowe, najczęściej wykorzystywane w optymalizacji wielokryterialnej były algorytmy ewolucyjne, które w szybkim czasie są w stanie znajdywać dobrą aproksymację frontu Pareto. Oprócz metod ewolucyjnych, w wielokryterialnym szeregowaniu zadań wykorzystuje się często metody lokalnego przeszukiwania, które pozwalają na dokładniejsze zbadanie aproksymacji i znalezienie nowych punktów Pareto.

Dubois-Lacoste i inni w pracy [8] zaproponowali hybrydowy algorytm lokalnego przeszukiwania, który zintegrował dwufazowe przeszukiwanie lokalnej (ang: *Two-Phase Local Search*) z przeszukiwaniem lokalnym Pareto (ang: *Pareto Local Search*). Algorytm rozwiązywał permutacyjny problem przepływowy z wielokryterialną funkcją celu, będącą kombinacją dwóch równocześnie optymalizowanych kryteriów, takich jak: a) czas

zakończenia wszystkich zadań, b) suma czasów zakończenia wykonywania zadań, c) całkowite (w tym ważone) spóźnienie i d) całkowity czas przepływu.

Zaproponowany przez Muratę, Ishibuchiego i Gena [9] algorytm *Cellular Multi-Objective Genetic Algorithm* (CMOGA), będący poprawioną wersją algorytmu MOGA [10], posiadał nowy schemat dystrybucji wag między kryteriami oraz wykorzystywał strukturę komórkową, która pozwoliła na lepszą selekcję wag i poprawę aproksymacji frontu Pareto. Oba w/w algorytmy zaprojektowane były do rozwiązywania wielokryterialnego problemu przepływowego i posiadały mechanizm zachowywania elitarnych rozwiązań, polegający na kopiowaniu unikalnych rozwiązań z frontu Pareto do kolejnej generacji osobników.

Algorytm symulowanego wyżarzania zaproponowali Chakravarthy i Rajendran w pracy [11]. Rozwiązywał on problem przepływowy z minimalizacją ważonej sumy dwóch kryteriów. Do zbioru rozwiązań początkowych dodawane były rozwiązania pochodzące z poniższych metod: a) *Earliest Due Date* (EDD), b) *Least Static Slack* (LSS) oraz c) heurystyki NEH [12]. Generowanie sąsiedztwa w zaproponowanym algorytmie odbywało się poprzez zamianę dwóch bezpośrednio sąsiadujących zadań.

Chang i inni w pracy [13] rozwinęli podejście *gradual-priority weighting* (GPW) służące do poszukiwania rozwiązań Pareto optymalnych w algorytmach genetycznych i genetycznych metodach lokalnego przeszukiwania. Zastosowali zaprojektowane przez siebie podejście między innymi do problemu przepływowego: a) dwukryterialnego z minimalizacją czasu zakończenia realizacji zadań i całkowitego spóźnienia oraz b) trójkryterialnego z minimalizacją czasu zakończenia realizacji zadań, całkowitego spóźnienia oraz sumarycznego czasu przepływu. Porównano je z różnymi podejściami ważonymi i wykazano iż podejście GPW daje znacznie lepsze rezultaty.

Kompleksowa metoda hybrydowa *Multi-Objective Particle Swarm optimization* (MOPS), zaproponowana przez Rahimi-Vaheda i Mirghorbaniego [14], optymalizowała kryteria przepływu oraz całkowitego spóźnienia. Elitarny algorytm przeszukiwania z zabronieniami wykorzystywany jest do inicjalizacji roju, podczas gdy procedura lokalnego przeszukiwania zastosowana jest w celu poprawy rozwiązania reprezentowanego przez każdą z cząstek.

2. Opis problemu

W permutacyjnym systemie przepływowym park maszynowy składa się z m maszyn ze zbioru $M=\{1,\ldots,m\}$. W systemie należy wykonać n zadań ze zbioru $J=\{1,\ldots,n\}$. Zadnie $j\in J$ wykonywane jest najpierw na maszynie 1, potem 2 itd. Czas wykonania zadania j, $j\in J$ na maszynie k, $k\in M$ wynosi $p_{jk}>0$. Zadania na maszynach muszą być wykonywane w identycznej kolejności. W dowolnej chwili maszyna może wykonywać tylko jedno zadanie oraz zadanie może być wykonywane tylko na jednej maszynie.

Niech $\pi=(\pi(1),\ldots,\pi(n))$ będzie kolejnością wykonywania zadań na każdej maszynie. Moment rozpoczęcia (zakończenia) wykonywania zadania j na maszynie k będziemy oznaczali symbolem S_{jk} (C_{jk}). Dopuszczalny harmonogram opisany momentami rozpoczęcia i zakończenia realizacji zadań na maszynach dla zadanej kolejności wykonywania zadań π musi spełniać nierówności (1−4):

$$S_{jk}\geq 0,\quad j=1,\ldots n,\ k=1,\ldots m, \tag{1}$$

$$S_{\pi(j),k} \geq C_{\pi(j-1),k}, \quad j=2,\ldots n,\ k=1,\ldots m, \tag{2}$$

$$S_{\pi(j),k} \geq C_{\pi(j),k-1}, \quad j=1,\ldots n,\ k=2,\ldots m, \tag{3}$$

$$C_{jk} = S_{jk} + p_{jk}, \quad j=1,\ldots n,\ k=1,\ldots m, \tag{4}$$

Momenty zakończenia spełniające ograniczenia (1)−(4) można wyznaczyć ze znanego wzoru rekurencyjnego (5), który można rozwiązać w sposób iteracyjny w czasie O(*nm*).

$$C_{\pi(j),k} = \max(C_{\pi(j-1),k}, C_{\pi(j),k-1}) + p_{\pi(j),k}, \tag{5}$$

gdzie $\pi(0)=0$, $C_{j,0}=0$ dla $j=1,\ldots,n$ oraz $C_{0,k}=0$ dla $k=1,\ldots,m$.

Zadanie optymalizacji polega na wyznaczeniu kolejności oraz harmonogramu wykonywania zadań minimalizującego następujące funkcje kryterialne:

1. moment zakończenia realizacji wszystkich zadań

$$C_{\max}(\pi) = \max_{1\leq j\leq n} \{C_{\pi(j)m}\}`, \tag{6}$$

2. sumę czasów zakończenia zadań

$$C_{sum}(\pi) = \sum_{j=1}^{n} C_{\pi(j)m}. \tag{7}$$

Wartości $C_{j,k}$ wyznaczone ze wzoru (5) są najmniejsze z możliwych, zatem minimalizują obie funkcje kryterialne dla zadanej kolejności wykonywania zadań π.

3. Algorytm oparty na technice przeszukiwania z zabronieniami (Tabu Search)

Algorytmy oparte na metodzie przeszukiwania z zabronieniami (ang. tabu serach - TS) należą do jednych z najefektywniejszych metod konstruowania iteracyjnych algorytmów heurystycznych dla problemów szeregowania zadań produkcyjnych. Potwierdzają to wyniki badań przedstawione w licznych artykułach naukowych [1,2].

W każdej iteracji algorytmu TS generowane jest sąsiedztwo rozwiązania bieżącego. Sąsiedztwo składa się z rozwiązań powstałych z rozwiązania bieżącego przez jego niewielką modyfikację zwaną ruchem. Zbiór ruchów jednoznacznie definiuje zbiór rozwiązań sąsiednich rozwiązania bieżącego. Z sąsiedztwa wybierane jest rozwiązanie najlepsze, które staje się rozwiązaniem bieżącym w następnej iteracji algorytmu. Najistotniejszym elementem metody jest mechanizm zabronień. Zapobiega on powrotom do rozwiązań wcześniej generowanych. Mechanizm zabronień implementuje się w postaci listy stanowiącej pamięć krótkoterminową historii przeszukiwań. Na liście pamiętane są atrybuty rozwiązań i/lub ruchów. Zawartość listy dzieli sąsiedztwo na dwa podzbiory: zbiór rozwiązań niezabronionych oraz zbiór rozwiązań zabronionych. Rozwiązania ze zbioru rozwiązań zabronionych nie są brane pod uwagę podczas wyboru rozwiązania najlepszego.

W przypadku optymalizacji jednokryterialnej ocena rozwiązań oraz kierunek poszukiwań jest jednoznacznie określony, tzn. poszukujemy rozwiązania o najmniejszej wartości funkcji kryterialnej. W przypadku optymalizacji wielokryterialnej należy

wyznaczyć rozwiązania, które minimalizują wartości obu funkcji kryterialnych oraz punkty pośrednie co jest znacznie trudniejsze.

1. $\pi=\pi_0$, $PS=\emptyset$.
2. for rep=1,…,$lrep$
2.1 $w=w_{rep}$, $maxiter=maxiter_{rep}$,
2.2 for $iter$=1,…,$maxiter$
2.2.1 foreach v in V
2.2.1.1 generate π_v, compute $C_{max}(\pi_v)$, $C_{sum}(\pi_v)$,
2.2.1.2 update PS with π_v
2.2.2 select not forbidden π_v with lowest $f(\pi_v)$
2.2.3 update L with π and v
2.2.4 set $\pi=\pi_v$

Rys. 1. Schemat algorytmu TS

Proponowany algorytm TS bazuje na algorytmie [1] zaproponowanym dla permutacyjnego problemu przepływowego z kryterium czasu zakończenia realizacji zadań. W algorytmie zastosowano sąsiedztwo generowane przez otoczenie typu *wstaw* (ang. *insert*). Zbiór ruchów wstaw dla permutacji składającej się z n elementów zdefiniowany jest następująco: $V=\{(x,y): x\neq y,\ x,y=1,\ldots,n\}$. Rozwiązanie sąsiednie π_v generowane przez ruch $v=(x,y) \in V$ przyjmuje następującą postać:

$$\pi_v=(\pi(1),...,\pi(x-1),\pi(x+1),\ldots,\pi(y-1),\pi(x),\pi(y),\ldots,\pi(n)) \quad \text{dla} \quad x<y \tag{8}$$

oraz

$$\pi_v=(\pi(1),...,\pi(y-1),\pi(x),\pi(y),\ldots,\pi(x-1),\pi(x+1),\ldots,\pi(n)) \quad \text{dla} \quad x>y. \tag{9}$$

Dla ruchu $v=(x,y)$ generującego nowe rozwiązanie bieżące, na liście zabronień zapisywana była para $(\pi(x+1),\pi(x))$ dla $x<y$, oraz $(\pi(x),\pi(x-1))$ dla $x>y$. Rozwiązanie π_v jest zabronione jeżeli istnieje na liście zabronień co najmniej jedna para (a,b) taka, że zadanie b wykonywane jest przed zadaniem a w kolejności π_v. Lista zabronień ma ograniczoną długość i jest obsługiwana zgodnie z regułą FIFO.

W algorytmie TS przyjęto ważoną funkcję oceny $f(\pi)$ rozwiązania π określoną przez wyrażanie (10)

$$f(\pi)=w\cdot C_{max}(\pi)+C_{sum}(1-w)\cdot(\pi) \tag{10}$$

Łatwo można się przekonać, że dla w=1 wartość $f(\pi)=C_{max}(\pi)$, natomiast dla w=0·wartość $f(\pi)=C_{sum}(\pi)$.

Na Rysunku 1 przedstawiono schemat proponowanego algorytmu TS. Symbolem PS oznaczono zbiór Pareto, natomiast symbolem L listę zabronień. Właściwa realizacja algorytmu TS następuje w krokach 2.2.1-2.2.4. W krokach 2.2.1.1 i 2.2.1.2 dla każdego rozwiązania z otoczenia rozwiązania bieżącego π wyznaczane są wartości funkcji obu

kryteriów optymalizacyjnych oraz uaktualniany jest zbiór Pareto. W kroku 2.2.2 następuje wybór niezabronionego rozwiązania sąsiedniego o najmniejszej wartości funkcji $f()$. W kolejnym kroku następuje aktualizacja zawartości listy zabronień L na podstawie rozwiązania π oraz ruchu v generującego rozwiązanie π_v. Podstawienie, w ostatnim kroku, czyni rozwiązanie π_v rozwiązaniem bieżącym w następnej iteracji algorytmu.

Realizacja głównej części algorytmu TS powtarzana jest *lrep* (krok 2) krotnie dla różnych wartości parametru w. Wartość parametru w oraz liczba iteracji algorytmu *maxiter* dla powtórzenia *rep* ustalana jest w kroku 2.1. W kroku 1 rozwiązanie bieżące π inicjowane jest rozwiązaniem początkowym π_0.

4. Eksperyment komputerowy

Sformułowano dwa cele eksperymentu komputerowego. Pierwszym celem było zbadanie zdolności zaproponowanego algorytmu do eksploracji różnych fragmentów frontu Pareto. Drugim celem było porównanie efektywności algorytmu z efektywnością algorytmów literaturowych. Algorytm TS został zakodowany w języku C++ w środowisku Visual Studio 2010. Testy przeprowadzono na komputerze z procesorem Intel i7 2.3 GHz. Podczas testów zastosowano zbiór przykładów zaproponowanych przez Ruiza [15] dla których zostały podane zbiory Pareto wygenerowane przy pomocy algorytmów literaturowych.

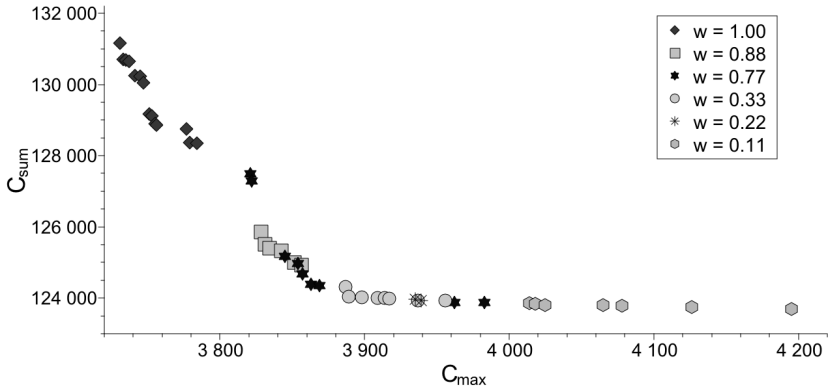
Rozwiązanie początkowe dla algorytmu TS wygenerowano algorytmem NEH [12], który jest najefektywniejszym algorytmem konstrukcyjnym dla problemu przepływowego z kryterium czasu zakończenia zadań. W każdym uruchomieniu algorytmu TS wykonano $lrep$=10 powtórzeń dla współczynnika w=(rep−1)/($lrep$−1) w powtórzeniu rep. W każdym powtórzeniu algorytm wykonywał $maxiter$=5000 iteracji. Wyniku działania algorytmu otrzymujemy zbiór punktów Pareto.

Na Rys 2. został przedstawiony zbiór Pareto wygenerowany przez algorytm TS dla instancji TA57. Punkty Pareto wygenerowane dla różnych wartości współczynnika w zostały zaznaczone innym symbolem graficznym. Z analizy Rysunku wynika, że wraz ze wzrostem współczynnika w zmienia się fragment Frontu Pareto przeszukiwany przez algorytm TS. Świadczą o tym punkty Pareto znajdowane dla tych współczynników. Wraz ze wzrostem w, obszar przeszukiwań przesuwa się z rozwiązań bliskich minimum funkcji $C_{\max}$ do rozwiązań bliskich minimum funkcji C_{sum}.

W celu oceny jakości generowanych zbiorów Pareto przez algorytm TS, dla każdej instancji testowej wyznaczono:

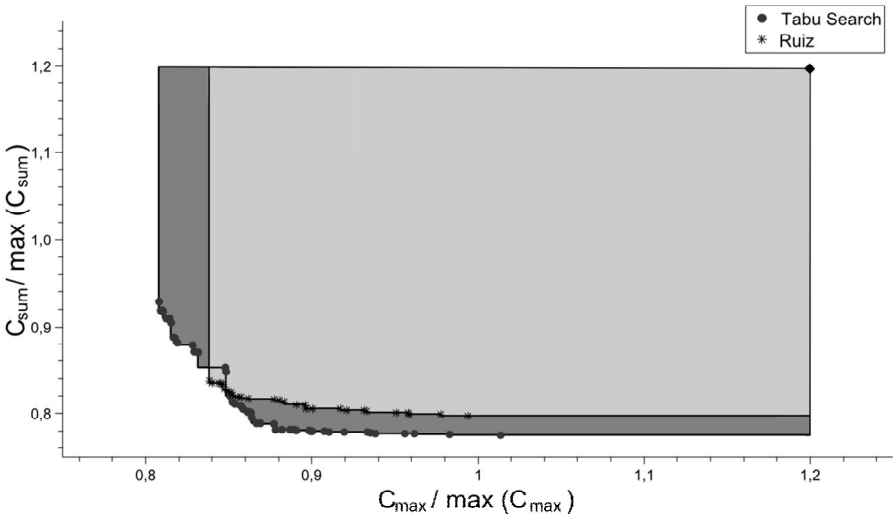
- P − zbiór punktów Pareto wyznaczony dla sumy zbiorów: wygenerowanego algorytmem TS (PS_{TS}) oraz zbioru Ruiza (PS_R),
- $P_{TS} = PS_{TS} \cap P$ − podzbiór zbioru P znajdujący się w zbiorze wygenerowanym przez algorytm TS,
- $P_R = PS_R \cap P$ − podzbiór zbioru P znajdujący się w zbiorze Ruiza,
- $D_{TS} = P_{TS} \setminus P_R$ − rozwiązania Pareto znajdujące się w P_{TS} i nie znajdujące się w P_R,
- $D_R = P_R \setminus P_{TS}$ − rozwiązania Pareto znajdujące się w P_R i nie znajdujące się w P_{TS}.

Przedstawione wskaźniki pozwalają na ocenę jakości zbiorów Pareto jedynie w sposób ilościowy, niestety pomijają one rozwiązania zdominowane przez rozwiązania konkurenta. Znacznie lepszymi wskaźnikiem, uwzględniającym tego typu rozwiązania jest wskaźnik Hyper-Volume [16][17].



Rys. 2. Zbiór Pareto wygenerowany algorytmem TS dla instancji TA57

Hyper-Volume Indicator jest numeryczną reprezentacją obszaru przestrzeni pomiędzy rozwiązaniem referencyjnym oraz aproksymacją frontu Pareto danego algorytmu. Punkt referencyjny pozostaje stały dla wszystkich porównywanych zbiorów (algorytmów) i przyjmowany jest jako punkt o wartościach kryteriów odpowiadających 120% najgorszych znalezionych dla nich wartości wśród wszystkich zbiorów rozwiązań. Wskaźnik ten jest zgodny z Pareto efektywnością. Dla każdej instancji wyznaczono: HVI_{TS} oraz HVI_R odpowiednio dla zbiorów PS_{TS} oraz PS_R. Na Rysunku 3 przedstawiono zbiory Pareto PS_{TS} oraz PS_R dla instancji Ta57 oraz zilustrowano sposób wyznaczenia indeksów HVI_{TS} i HVI_R.



Rys. 3. Wizualizacja wyznaczenia wskaźnika HVI – instancja TA57

Tab. 1. Wyniki badań eksperymentalnych algorytmu

Grupa	$\frac{\lvert P_{TS}\rvert}{\lvert P\rvert}$	$\frac{\lvert P_R\rvert}{\lvert P\rvert}$	$\frac{\lvert D_{TS}\rvert}{\lvert P\rvert}$	$\frac{\lvert D_R\rvert}{\lvert P\rvert}$	HVI_{TS}	HVI_R	$\frac{HVI_{TS}}{HVI_R}$	$\left(1-\frac{HVI_{TS}}{HVI_R}\right)$
	[%]	[%]	[%]	[%]				[%]
20×5	43,9	91,1	8,9	56,1	0,0480	0,0482	0,9960	−0,40
20×10	81,5	89,3	10,7	18,5	0,0521	0,0520	1,0025	0,25
20×20	80,4	84,0	16,0	19,6	0,0506	0,0507	0,9974	−0,26
50×5	5,3	94,7	5,3	94,7	0,0544	0,0564	0,9641	−3,59
50×10	66,5	33,5	66,5	33,5	0,0544	0,0529	1,0296	2,96
50×20	79,0	21,0	79,0	21,0	0,0513	0,0484	1,0586	5,86
Średnia	**59,4**	**69,0**	**31,0**	**40,6**	**0,0518**	**0,0514**	**1,0080**	**0,80**

W Tab. 1 zostały zebrane wyniki badań eksperymentalnych algorytmu. W drugiej kolumnie znajduje się średnia dla grupy instancji liczba rozwiązań wygenerowanych przez algorytm TS znajdujących się w zbiorze P odniesiona do liczby rozwiązań w tym zbiorze, w trzeciej kolumnie znajduje się ta sama wielkość dla zbiorów Ruiza. W czwartej kolumnie znajduje się średnia liczba rozwiązań wygenerowanych przez algorytm TS, które nie znajdują się w PS_R odniesiona do liczby rozwiązań w tym zbiorze P. Analogiczna wielkość dla zbiorów Ruiza została zebrana w czwartej kolumnie. W piątej i szóstej kolumnie zebrano średnią wartość współczynnika *HVI* dla zbiorów wygenerowanych algorytmem TS i Ruiza. Wyniki badań dla konkretnych instancji zostały zebrane w Tabeli 2 i Tabeli 3.

Obserwując wyniki znajdujące się w Tabelach 1−3 należy zwrócić uwagę iż wyniki Ruiza są sumarycznymi wynikami uzyskanymi z szesnastu testowanych w pracy [15] algorytmów meta-heurystycznych, z których każdy uruchamiany był dziesięciokrotnie. Natomiast zaproponowany algorytm TS uruchamiany był jednokrotnie dla każdej z instancji.

Z analizy rezultatów badań prezentowanych w Tab. 1 wynika, że algorytm TS uzyskał zbliżone lub lepsze współczynniki *HVI* od współczynników uzyskanych w rozległych testach Ruiza. Średnia wartość *HVI* dla instancji o małych rozmiarach (składających się z n=20 zadań) różni się w granicach od −0.4 (na korzyść zbiorów Ruiza) do 0.25 (na korzyść zbiorów na korzyść zbiorów TS). Dla instancji o dużych rozmiarach (n=50) różnice są znacznie większe i wahają się w granicach od −0.59 (na korzyść zbiorów Ruiza dla m=5) do 5.86 (na korzyść zbiorów na korzyść zbiorów TS dla m=20). Średni *HVI* algorytmu TS dla wszystkich instancji jest o 0.8% większy od HVI dla zbiorów Ruiza.

Dodatkowo, dla instancji o rozmiarach 50×10 oraz 50×20 znalazł znacznie więcej rozwiązań Pareto optymalnych oraz unikalnych rozwiązań niezdominowanych. W przypadku instancji o liczbie maszyn m = 5 algorytm TS uzyskał zbliżone wartości współczynnika *HVI*, jednak znalazł przy tym mniej punktów Pareto.

Warto przy tym przypomnieć, iż zaproponowany algorytm TS zdołał w przybliżonym czasie uzyskać zbliżone lub nawet przewyższyć wyniki skumulowane z szesnastu zbadanych w pracy Ruiza algorytmów. Wiele spośród tych ostatnich uznawanych jest za jedne z najlepszych w literaturze.

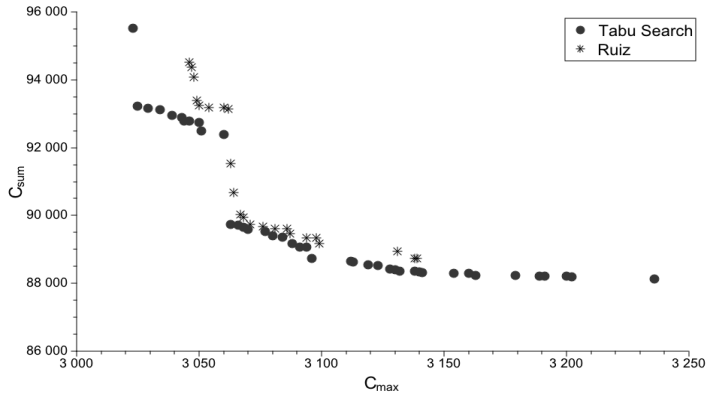
Tab. 2. Szczegółowe wyniki badań eksperymentalnych

LP	$\|P\|$	$\|P_{TS}\|$	$\|P_R\|$	$\|D_{TS}\|$	$\|D_R\|$	LP	$\|P\|$	$\|P_{TS}\|$	$\|P_R\|$	$\|D_{TS}\|$	$\|D_R\|$
1	5	4	5	0	1	**31**	10	0	10	0	10
2	9	2	7	2	7	**32**	26	4	22	4	22
3	15	11	11	4	4	**33**	18	0	18	0	18
4	18	11	15	3	7	**34**	32	6	26	6	26
5	20	1	20	0	19	**35**	7	0	7	0	7
6	22	8	22	0	14	**36**	24	0	24	0	24
7	15	7	14	1	8	**37**	15	0	15	0	15
8	19	7	17	2	12	**38**	13	1	12	1	12
9	17	12	16	1	5	**39**	27	3	24	3	24
10	15	1	15	0	14	**40**	18	0	18	0	18
11	16	15	15	1	1	**41**	41	41	0	41	0
12	26	23	26	0	3	**42**	35	11	24	11	24
13	15	14	14	1	1	**43**	48	39	9	39	9
14	25	20	17	8	5	**44**	31	27	4	27	4
15	15	12	14	1	3	**45**	39	37	2	37	2
16	36	17	27	9	19	**46**	46	30	16	30	16
17	16	15	16	0	1	**47**	39	12	27	12	27
18	18	14	15	3	4	**48**	40	20	20	20	20
19	21	21	20	1	0	**49**	36	29	7	29	7
20	23	14	21	2	9	**50**	39	17	22	17	22
21	27	17	27	0	10	**51**	49	34	15	34	15
22	28	21	24	4	7	**52**	79	65	14	65	14
23	42	32	42	0	10	**53**	40	40	0	40	0
24	19	17	19	0	2	**54**	41	30	11	30	11
25	31	28	17	14	3	**55**	38	34	4	34	4
26	20	15	15	5	5	**56**	51	34	17	34	17
27	23	21	15	8	2	**57**	51	44	7	44	7
28	49	37	45	4	12	**58**	44	39	5	39	5
29	33	30	26	7	3	**59**	64	41	23	41	23
30	27	21	24	3	6	**60**	47	33	14	33	14

Analizując Tabele 2 i 3 można zauważyć iż od instancji TA41 (rozmiar 50×10) do instancji TA60 włącznie (rozmiar 50×20) współczynnik HVI_{TS} za każdym razem przewyższa współczynnik HVI_{Ruiz}. Co więcej, dla niemal każdej z tych instancji algorytm TS znalazł więcej rozwiązań Pareto oraz unikalnych rozwiązań niezdominowanych. W niektórych przypadkach zdominował wszystkie porównywane rozwiązania.

Tab. 3. Szczegółowe wyniki badań eksperymentalnych

LP	HVI_{TS}	HVI_{Ruiz}	$\frac{HVI_{TS}}{HVI_{Ruiz}}$	LP	HVI_{TS}	HVI_{Ruiz}	$\frac{HVI_{TS}}{HVI_{Ruiz}}$
1	0,0344	0,0344	0,9999	**31**	0,0672	0,0692	0,9711
2	0,0357	0,0359	0,9938	**32**	0,0448	0,0466	0,9614
3	0,0624	0,0625	0,9993	**33**	0,0508	0,0528	0,9618
4	0,0456	0,0456	1,0000	**34**	0,0547	0,0552	0,9919
5	0,0513	0,0522	0,9832	**35**	0,0507	0,0547	0,9272
6	0,0560	0,0564	0,9939	**36**	0,0506	0,0533	0,9498
7	0,0384	0,0385	0,9996	**37**	0,0532	0,0549	0,9690
8	0,0506	0,0506	0,9999	**38**	0,0576	0,0600	0,9610
9	0,0481	0,0481	0,9994	**39**	0,0608	0,0626	0,9704
10	0,0572	0,0578	0,9906	**40**	0,0533	0,0545	0,9776
11	0,0475	0,0474	1,0017	**41**	0,0506	0,0484	1,0454
12	0,0563	0,0563	0,9998	**42**	0,0539	0,0535	1,0082
13	0,0552	0,0549	1,0069	**43**	0,0574	0,0542	1,0599
14	0,0518	0,0513	1,0087	**44**	0,0554	0,0538	1,0292
15	0,0391	0,0392	0,9993	**45**	0,0497	0,0474	1,0490
16	0,0728	0,0728	0,9992	**46**	0,0548	0,0528	1,0376
17	0,0472	0,0472	1,0000	**47**	0,0545	0,0544	1,0032
18	0,0493	0,0491	1,0029	**48**	0,0574	0,0564	1,0176
19	0,0451	0,0451	1,0000	**49**	0,0581	0,0565	1,0296
20	0,0572	0,0568	1,0061	**50**	0,0523	0,0514	1,0165
21	0,0453	0,0478	0,9482	**51**	0,0496	0,0475	1,0445
22	0,0514	0,0513	1,0009	**52**	0,0513	0,0481	1,0675
23	0,0576	0,0576	0,9994	**53**	0,0506	0,0475	1,0646
24	0,0443	0,0444	0,9999	**54**	0,0498	0,0470	1,0594
25	0,0579	0,0577	1,0039	**55**	0,0501	0,0468	1,0694
26	0,0476	0,0475	1,0018	**56**	0,0581	0,0550	1,0551
27	0,0441	0,0433	1,0183	**57**	0,0546	0,0506	1,0797
28	0,0604	0,0602	1,0028	**58**	0,0463	0,0438	1,0559
29	0,0516	0,0516	0,9999	**59**	0,0505	0,0478	1,0571
30	0,0454	0,0455	0,9988	**60**	0,0517	0,0500	1,0332



Rys. 4. Aproksymacja frontu Pareto - instancja TA41

W pozostałych grupach algorytm TS wykazał zbliżone wyniki, zarówno pod względem liczby rozwiązań jak i wskaźnika HVI, do rozwiązań Ruiza. Wyjątkiem jest grupa instancji o rozmiarze 50x5. Pomimo zbliżonych wartości współczynnika *HVI*, rozwiązania uzyskane przez algorytm TS zostały w większości zdominowane.

Na Rys 4. przedstawiono zbior Pareto Ruiza oraz wygenerowanego przez algorytm TS dla instancji TA41. Obraz jest typowym dla instancji o rozmiarach 50×10 oraz 50×20. Co warto podkreślić, w przypadku tych grup algorytm TS generuje rozwiązania o znacznie mniejszej wartości dla obu funkcji kryterialnych od rozwiązań znajdujących się w zbiorach Ruiza.

5. Podsumowanie

W pracy zaproponowano nowy algorytm oparty na metodzie przeszukiwania z zabronieniami dla problemu przepływowego z optymalizacją wielokryterialną. Zaproponowano oryginalny sposób sterowania procesem przeszukiwania algorytmu umożliwiającego eksplorację różnych fragmentów frontu Pareto. Z przeprowadzonych badań eksperymentalnych wynika, że zaproponowany algorytm generuje rozwiązania o porównywalnej jakości (dla instancji o małych rozmiarach) lub istotnie lepsze (dla instancji o dużych rozmiarach) od rozwiązań wygenerowanych przez algorytmy zaproponowane w literaturze światowej.

Literatura

1. Nowicki E., Smutnicki C., A fast tabu search algorithm for the permutation flow-shop problem, European Journal of Operational Research, 91, 1, 1996, 160–175.
2. Grabowski, J., Pempera, J., New block properties for the permutation flow shop problem with application in tabu search, Journal of the Operational Research Society, 52, 2, 2001, 210–220.
3. Osman IH., Potts CN., Simulated annealing for permutation flow-shop scheduling, Omega 17, 6, 1989, 551–557.
4. Chinyao L., Jinn-Yi Y., Kai-I H., A robust simulated annealing heuristic for flow shop scheduling problems, The International Journal of Advanced Manufacturing Technology, 2004, 23, 9-10, 762–767.

5. Ying K-C., Liao C-J., An ant colony system for permutation flow-shop sequencing, Computers & Operations Research Volume 31, Issue 5, April 2004, 791–801
6. Wang L., Panb Q-K., Fatih Tasgetiren M., Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms, Expert Systems with Applications 37, 12, 2010, 7929–7936
7. Fatih Tasgetiren M., Liang Y-C., Sevklic M., Gencyilmaz G., A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European Journal of Operational Research Volume 177, 3, 2007, 1930–1947
8. Dubois-Lacoste J, López-Ibáñez M, Stutzle T. A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 38, 2011, 1219–1236.
9. Murata T., Ishibuchi H., Gen M.: Specification of genetic search directions in cellular multiobjective genetic algorithms, EMO '01 Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, 2001, 82–95.
10. Murata T., Ishibuchi H., Tanaka H.: Multi-objective genetic algorithm and its applications to flowshop scheduling, Computers and Industrial Engineering 30, 1996, 957–968.
11. Charavarthy K., Rajendran C.: A heuristic for scheduling in a flow shop with the bicriteria of makespan and maximum tardiness minimization, Production Planning and Control 10, 1999, 707–714.
12. Nawaz M., Enscore Jr. E.E., Ham I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, OMEGA International Journal of Management Science 11, 1983, 91–95.
13. Chang P-C, Hsieh J-C, Lin S-G. The development of gradual-priority weighting approach for the multi-objective flowshop scheduling problem. International Journal of Production Economics 79 (3), 2002, 171–183.
14. Rahimi-Vahed, A. R., S. M. Mirghorbani., A multi-objective particle swarm for a flow shop scheduling problem. Journal of Combinatorial Optimization 2007 (13), 79–102.
15. Minella G., Ruiz R., Ciavotta M.: A Review and Evaluation of Multiobjective Algorithms for the Flowshop Scheduling Problem, INFORMS Journal on Computing Summer vol. 20 no. 3, 2008, 451–471.
16. Knowles J., Thiele L., Zitzler E.: A tutorial on the performance assessment of stochastic multiobjective optimizers., Tech. rep., ETH Zurich, 2006.
17. Zitzler E., Thiele L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactiions on Evolutionary Computation 3(4) , 1999, 257–271.

Dr inż. Jarosław Pempera
Mgr inż. Dominik Żelazny
Instytut Automatyki, Informatyki i Robotyki, Politechnika Wrocławska
50-372 Wrocław, ul. Wybrzeże Wyspiańskiego 27
tel./fax.: (71) 320-28-34
e-mail: jaroslaw.pempera@pwr.wroc.pl, dominik.zelazny@pwr.wroc.pl