AUTOMATYCZNE STEROWANIE OTOCZENIEM W ALGORYTMACH POPRAW

Mariusz MAKUCHOWSKI, Bartosz WIELEBSKI

Streszczenie: W pracy proponuje się modyfikację znanych algorytmów popraw poprzez dodanie nadrzędnego aparatu sterującego. Aparat ten śledzi trajektorię poszukiwań algorytmu wykonawczego i na bieżąco modyfikuje jego otoczenie. Efektywność proponowanej metody została przebadana na przykładzie algorytmu poszukiwania z zabronieniami i symulowanym wyżarzaniu. Proponowane algorytmy wraz z ich klasycznymi odpowiednikami zaimplementowane zostały do rozwiązywania permutacyjnego problemu przepływowego z kryterium będącym sumą zakończeń zadań. W pracy przedstawiono wyniki numeryczne pokazujące znaczne zwiększenie wydajności algorytmu poszukiwania z zabronieniami przy zastosowaniu proponowanej modyfikacji.

Słowa kluczowe: automatyczne strojenie, poszukiwanie z zabronieniami, symulowane wyżarzanie, problem przepływowy

1. Wstęp

Znaczna część problemów optymalizacji dyskretnej należy do klasy problemów NP-trudnych. Oznacza to, że nie istnieją dla nich efektywne, działające w czasie wielomianowym względem rozmiaru instancji, algorytmy dokładne (o ile klasa problemów P jest różna od klasy problemów NP). Sytuacja ta wymusza konieczność projektowania algorytmów aproksymacyjnych i heurystycznych. W algorytmach aproksymacyjnych błąd względny uzyskiwanych rozwiązań jest ograniczony przez pewną wartość ε. Oznacza to, że wartość funkcji celu rozwiązania uzyskanego takim algorytmem jest zawsze mniejsza lub równa (w przypadku problemu szukającego minimum) niż wartość funkcji celu rozwiązania optymalnego przemnożona przez ε. W algorytmach heurystycznych nie ma, żadnej gwarancji znalezienia rozwiązania znajdującego się dostatecznie blisko od rozwiązania optymalnego (odległość między rozwiązaniami rozumiana jest jako względna różnica wartości funkcji celu tych rozwiązań). Pomimo tego, algorytmy heurystyczne cieszą się dużym zainteresowaniem w śród badaczy i praktyków a spowodowane jest to ich dużą efektywnością. Wśród algorytmów heurystycznych są zarówno algorytmy konstrukcyjne jak i algorytmy popraw. Algorytmy popraw działają iteracyjnie i w każdym kroku starają się zmodyfikować swoje aktualne rozwiązanie na rozwiązanie statystycznie lepsze (choć w pojedynczych iteracjach dopuszcza się przejście do rozwiązań gorszych). Do klasy algorytmów popraw należą między innymi algorytmy oparte na metodzie poszukiwania z zabronieniami i metodzie symulowanego wyżarzania.

W niniejszej pracy proponuje się dodatkową technikę podnoszącą wydajność metod poszukiwania z zabronieniami i symulowanego wyżarzania. Ocena zaproponowanej modyfikacji, wystawiona zostaje na podstawie porównania klasycznych algorytmów z ich zmodyfikowanymi odpowiednikami. Wybór problemu, który będą rozwiązywać testowane

algorytmy padł na klasyczny problem z teorii szeregowania zadań, a dokładniej na permutacyjny problem przepływowy (ang. flow-shop sequencing problem) z kryterium będącym sumą terminów zakończenia wykonywania wszystkich zadań (ang. total flowtime).

Ponieważ, permutacyjny problem przepływowy jest szczególnym przypadkiem szerszego zagadnienia, jakim jest ogólny problem przepływowy, wybór wymaga uzasadnienia.

- Zdecydowana większość prac zajmująca się problemem przepływowym ogranicza się do wersji permutacyjnej.
- Sterowanie systemem permutacyjnym jest prostsze.
- Dla niektórych klas problemów przepływowych rozwiązanie optymalne jest rozwiązaniem permutacyjnym.
- Błąd pomiędzy rozwiązaniami optymalnymi problemu ogólnego i permutacyjnego jest zazwyczaj nieznaczny.

Ponieważ najczęstszym kryterium optymalizacji w teorii szeregowania zadań jest najpóźniejszy moment zakończenia wykonywania zadań $C_{\max}$, a w pracy rozważa się kryterium będącym sumą czasów zakończeń zadań C_{sum}, także ten wybór wymaga wyjaśnienia. Decyzja o wyborze C_{sum} jako kryterium optymalizacji podjęta została na podstawie poniższych faktów.

- Problem z kryterium C_{sum} oceniany jest przez badaczy jako znacznie trudniejszy niż analogiczny problem z kryterium $C_{\max}$.
- W przeciwieństwie do problemu z kryterium C_{sum}, w problemie z kryterium $C_{\max}$ znane są specyficzne własności przestrzeni rozwiązań, które nie koniecznie występują w innych problemach.

Celem pracy jest ocena dodatkowej techniki zwiększającej efektywność metody poszukiwania z zabronieniami i symulowanego wyżarzania, więc wybór problemu, dla którego znane są dobre rozwiązania (najlepsze opublikowane wyniki dla znanych przykładów testowych) oraz w którym nie istnieją specyficzne własności wydaje się najlepszy.

2. Model matematyczny problemu

Wszystkie projektowane w niniejszej pracy algorytmy dedykowane są permutacyjnemu problemowi przepływowemu, z kryterium będącym sumą momentów zakończenia wykonywanych zadań. Problem ten oznaczany jest w notacji Grahama [1] poprzez $F^* \| C_{sum}$. Jego matematyczny model przedstawiony poniżej zaczerpnięty został z pracy [2].

Niech $J = \{1,2,...,n\}$ oznacza zbiór zadań do wykonania przy pomocy zbioru maszyn $M = \{1,2,...,m\}$ o ograniczonej jednostkowej przepustowości. Zadanie $j \in J$ składa się z ciągu operacji $O_{1j},...,O_{mj}$. Operacja O_{ij} zadania *j* wykonywana jest bez przerywania na maszynie *i* w czasie $p_{ij} \geq 0$. Rozwiązaniem jest harmonogram pracy maszyn reprezentowany poprzez macierze terminów rozpoczęcia $S = (S_1,...,S_n)$, gdzie

$S_j = (S_{1j},...,S_{mj})$ oraz zakończenia operacji $C = (C_1,...,C_n)$, gdzie $C_j = (C_{1j},...,C_{mj})$; $C_{ij} = S_{ij} + p_{ij}$. Dla regularnych funkcji celu optymalny harmonogram dosunięty jest w lewo na osi czasu, zatem można go poszukiwać w zbiorze takich rozwiązań. W tym przypadku każde rozwiązanie możemy jednoznacznie reprezentować kolejnością wykonywania zadań na poszczególnych maszynach. Analizowany w niniejszej pracy problem zakłada dodatkowo, iż kolejność wykonywania zadań na maszynach jest jednakowa i oznaczana poprzez $\pi = (\pi(1),...,\pi(n))$ permutację elementów zbioru *J*. Zbiór wszystkich możliwych permutacji zbioru *J* będziemy oznaczać poprzez Π. Dla znanej kolejności wykonywania zadań $\pi \in \Pi$ terminy rozpoczęcia i zakończenia wykonywania operacji muszą spełniać poniższe ograniczenia:

$$C_{\pi(i)j} \le S_{\pi(i+1),j}, \quad i = 1,...,n-1,\ j = 1,...,m \tag{1}$$

$$C_{i,j} \le S_{i,j+1}, \quad i = 1,...,n,\ j = 1,...,m-1 \tag{2}$$

Najmniejsze możliwe wartości terminów zakończenia wykonywanych operacji można wyznaczyć z rekurencyjnego wzoru:

$$C_{\pi(i)j} = \max\{C_{\pi(i-1),j}, C_{\pi(i),j-1}\} + p_{\pi(i),j} \tag{3}$$

w którym $i = 1,...,n$, $j = 1,...,m$, $\pi(0) = 0$, $C_{0,j} = 0$ dla $j = 1,...,m$ oraz $C_{i,0} = 0$ dla $i = 1,...,n$.
Kryterium w optymalizacji jest sumą terminów wykonania wszystkich zadań i można je opisać wzorem:

$$C_{sum}(\pi) = \sum_{i=1}^{n} C_{i,m}, \quad \pi \in \Pi, \tag{4}$$

gdzie wartości $C_{i,m}$ wyznaczone zostały dla permutacji π. Problem sprowadza się do poszukiwania kolejności $\pi^* \in \Pi$, minimalizującej kryterium (4),

$$\pi^* \in \arg\min_{\pi \in \Pi} C_{sum}(\pi). \tag{5}$$

3. Przegląd literatury

Pierwsze sformułowanie problemu przepływowego znajduje się w pracy [3]. Permutacyjny problem przepływowy z kryterium C_{sum} analizowany w niniejszym opracowaniu już dla dwóch maszyn; $F^*2 \,||\, C_{sum}$, należy do klasy problemów silnie NP-trudnych, co zostało udowodnione w pracy [4]. Dla rozważanego problemu zaproponowano kilka algorytmów dokładnych opartych na metodzie podziału i ograniczeń przedstawionych w pracach [5,6]. Szereg propozycji algorytmów konstrukcyjnych znajduje się w pracach [7,8,9,10]. Podejmowane też były próby tworzenia różnego rodzaju algorytmów popraw.

Algorytmy genetyczne przedstawione są w pracach [11,12,13]. Prace [14,15] opisują algorytmy mrówkowe, w pracy [16] opisany jest algorytm bazujący na estymacji rozkładu, a w pracy [17] algorytm optymalizacji wielocząsteczkowej.

Modyfikowane metody nadają się do rozwiązywania szerokiej klasy problemów optymalizacji dyskretnej. Z praktycznego doświadczenia najefektywniejszą z znanych heurystyk jest metoda poszukiwania z zabronieniami (ang. tabu search), choć istnieją specyficzne problemy, dla których wskazane jest stosowanie algorytmów bazujących na innych z heurystykach. Idea metody poszukiwania z zabronieniami zaproponowana została w pracach [18,19]. Od tej pory na jej bazie powstało wiele algorytmów dla różnych problemów optymalizacji. Algorytmy te są niezwykle wydajne i często są najlepszymi znanymi algorytmami dla danego problemu.

Kolejną analizowaną i modyfikowaną heurystyką jest symulowane wyżarzanie (ang. simulated annealing). Po raz pierwszy została ona opisana w pracy [20]. Metoda ta została zainspirowana zjawiskami termodynamicznym występującymi podczas procesu hartowania stali lub szkła. Z inspiracji tej wynikają także nazwy parametrów występujących w tej metodzie takich jak temperatura czy schemat chłodzenia. Metoda ta charakteryzuje się dużą wydajnością, zbliżoną do metody poszukiwania z zabronieniami, a algorytmy na niej oparte są stosunkowo łatwe w implementacji.

4. Automatyczne sterowanie otoczeniem

Metody takie jak poszukiwanie z zabronieniami czy symulowane wyżarzanie nie są gotowymi, jednoznacznymi przepisami budowy algorytmu, a jedynie ideą, na której bazuje projektowany algorytm. Na bazie danej metody można, więc tworzyć algorytmy dedykowane różnorodnym problemom. Ponadto nawet dla tego samego problemu można skonstruować wiele różnych algorytmów. Algorytmy rozwiązujące ten sam problem i bazujące na tej samej metodzie mogą się bardzo różnić nie tylko budową, ale przede wszystkim wydajnością. Dzieje się tak, dlatego, że to od twórcy algorytmu zależy nie tylko, w jaki sposób zaimplementuje dany byt występujący w metodzie, ale także precyzyjnie zdefiniuje pojęcie danego bytu.

Najważniejszym elementem w algorytmach popraw jest sąsiedztwo $N(x)$ rozwiązania $x \in X$, gdzie x oznacza rozwiązanie, a X zbiór wszystkich możliwych rozwiązań. W celu zdefiniowania sąsiedztwa wprowadza się pojęcie ruchu v, gdzie ruchem nazywa się przekształcenie pewnego rozwiązania w inne; $v: X \to X$. Zbiór wszystkich możliwych ruchów operujących na rozwiązaniu $x \in X$, oznaczamy przez $V(x)$. Teraz sąsiedztwo danego rozwiązania możemy opisać jako:

$$N(x) = \{v(x): \quad v \in V(x), x \in X\} \tag{6}$$

Taka definicja sąsiedztwa pozwala konstruktorowi algorytmu na dużą swobodę. Do najważniejszych założeń przyjmowanych w tej definicji należy precyzyjne zdefiniowanie pojęcia ruchu. Dla problemów, w których rozwiązaniem jest permutacja (tak jak w problemie $F^* \| C_{sum}$) spotyka się w literaturze następujące rodzaje ruchów:

- zamień: zamienia dwa elementy w permutacji,
- zamień sąsiednie: zamienia dwa sąsiednie element w permutacji,

- wstaw: wyciąga dany element z permutacji, a następnie wkłada go ponownie w inne miejsce,
- inwersja: zamienia kolejność elementów w pewnej części permutacji.

Oczywiście definicji ruchów można konstruować znacznie więcej choćby przez ograniczenia typu: przestaw element, ale nie więcej niż o k pozycji. Różne założenia przyjęte do definiowania ograniczeń tworzą wiele innych definicji ruchów.

W pracy proponuje się dodatkową zaawansowaną technikę sterującą algorytmami popraw. Realizowana ona będzie poprzez nadrzędny człon sterujący, który będzie miał wgląd w przebieg algorytmu wykonawczego (klasyczny algorytmu popraw) i możliwość dokonywania zmian parametrów sterujących tym algorytmem. Zasada działania członu nadrzędnego jest następująca. Algorytm nadrzędny posiada listę cykliczną zawierającą typy sąsiedztw oraz wylicza szybkość poprawy, jaką uzyskuje algorytm wykonawczy. Szybkość ta obliczana jest na podstawie wartości funkcji celu ostatnich analizowanych rozwiązań. Jeżeli szybkość poprawy spadnie poniżej ustalonego poziomu to algorytm sterujący zmienia sąsiedztwo w algorytmie wykonawczym. Zmiany sąsiedztwa dokonywane są cyklicznie.

Dodatkowo, jeżeli dla żadnego z dostępnych sąsiedztw, nie uzyskuje się akceptowalnej szybkości popraw, to algorytm sterujący generuje nowe dywersyfikacyjne sąsiedztwo. Sąsiedztwo to składa się tylko z jednego rozwiązania uzyskanego poprzez wykonanie kilku losowych ruchów z najlepszego rozwiązania znalezionego do tej pory.

Podsumowując, automatyczne sterowanie zapewnia zarówno intensyfikację jak i dywersyfikację obliczeń. Intensyfikacja występuje, gdy trajektoria poszukiwań dojdzie w pobliże minimum lokalnego. Wtedy to algorytm sterujący będzie wymuszał na algorytmie wykonawczym sprawdzenie okolicy przestrzeni rozwiązań tego minimum lokalnego poprzez pracę przy zastosowaniu wszystkich możliwych otoczeń. W przypadku, gdy wyczerpią się możliwe do zastosowania otoczenia, a algorytm wykonawczy nadal nie wykazuje poprawy generowanych rozwiązaniach, najprawdopodobniej poszukiwania utknęły w minimum lokalnym (już dobrze zbadanym). W takim wypadku algorytm sterujący dokonuje dywersyfikacji poszukiwań w nowy obszar przestrzeni rozwiązań.

5. Badania numeryczne

Dla problemu $F^* \| C_{sum}$ stworzono sześć algorytmów heurystycznych. Algorytmy TS_{ex} i TS_{ins} oparte są na klasycznej metodzie poszukiwania z zabronieniami. W algorytmie TS_{ex} zastosowano sąsiedztwo bazujące na ruchach typu zamień, a w algorytmie TS_{ins} sąsiedztwo oparte na ruchach typu wstaw, pomniejszonych o zbiór ruchów typu zamień sąsiednie. Dodatkowo przez algorytm TS będziemy oznaczać algorytm uruchamiający algorytmy TS_{ex} i TS_{ins} oraz zwracający lepsze z dwóch otrzymanych rozwiązań. Następnie skonstruowano algorytm TS^*, który był wyposażony w automatyczne sterowanie otoczeniem, a otoczenie wybierane było z pośród otoczeń zastosowanych w algorytmie TS_{ex} i TS_{ins}. Analogicznie stworzono algorytmy SA_{ex} i SA_{ins} bazujące na metodzie symulowanego wyżarzania oraz ich zmodyfikowaną wersję algorytm SA^*. Podobnie przez SA będziemy oznaczać algorytm będący złożeniem SA_{ex} i SA_{ins}.

Skonstruowane algorytmy testowane są na szczególnie trudnych 90 przykładach zaproponowanych przez Taillarda w pracy [21] (wszystkich przykładów jest 120, lecz w

pracy ograniczono się do pierwszych 90), których rozmiar waha się od 20 zadań i 5 maszyn do 100 zadań i 20 maszyn. Przykłady podzielone są na grupy po 10 przykładów o tej samej liczbie zadań i maszyn, a dana grupa jest notowana jako $n \times m$. Należy tu zauważyć, że trudność testowych przykładów wynika nie tylko z ich dużego rozmiaru, ale także z faktu, iż zostały one wybrane wśród dziesiątek tysięcy przykładów generowanych losowo, jako najbardziej „złośliwe". Dodatkowym elementem przemawiającym za wyborem tych przykładów są znane dla nich rozwiązania dobrej jakości. Rozwiązania te zostały otrzymane jako najlepsze rezultaty w wyniku badań wielu naukowców stosujących różne algorytmy. Dalej wartości funkcji celu dla tych rozwiązań będziemy nazywać wartościami referencyjnymi i notować jako C_{sum}^{REF}.

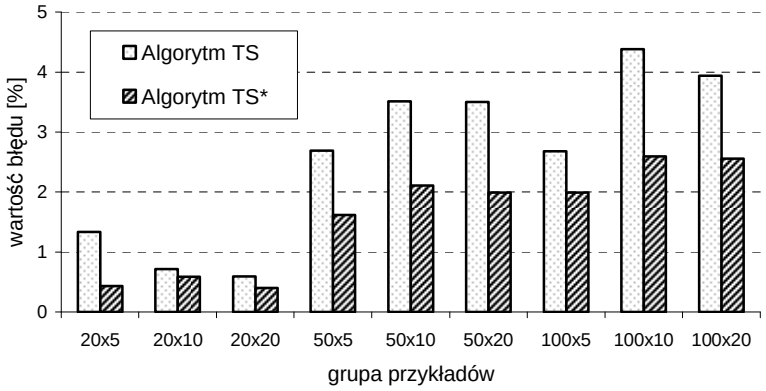
Każdym z opracowanych algorytmów rozwiązano wszystkie przykłady testowe. Wartość funkcji celu rozwiązania uzyskanego algorytmem $A \in \{TS, TS^*, SA, SA^*\}$ dla przykładu o numerze $i = 1,...,90$ oznaczono jako $C_{sum}^{A}(i)$. Następnie obliczono względną wartość błędu wygenerowanego rozwiązania w stosunku do rozwiązania referencyjnego według wzoru:

$$\delta^A(i) = 100\% \cdot \frac{C_{sum}^{A}(i) - C_{sum}^{REF}}{C_{sum}^{REF}}, \quad A \in \{TS, TS^*, SA, SA^*\}, i = 1,...,90. \qquad (7)$$

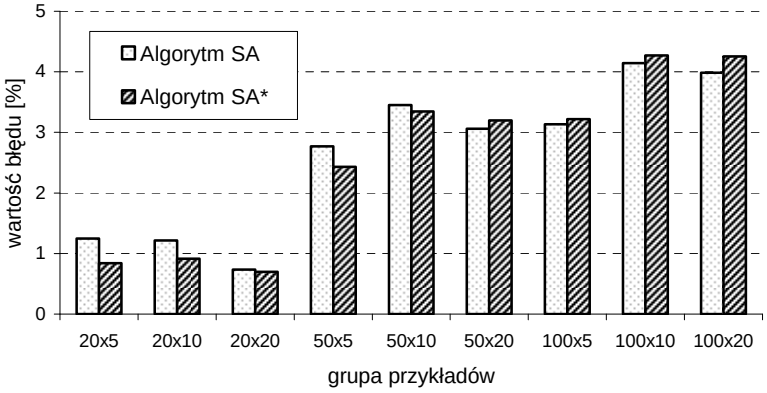
Uśrednione dla każdej grupy przykładów, wartości błędów względnych algorytmów typu poszukiwania z zabronieniami przedstawione zostały na rys. 1 a dla algorytmów typu symulowanego wyżarzania na rysunku 2.

Średnia wartość błędu względnego, obliczona na podstawie wszystkich testowanych przykładach jest bardzo mała i wynosi odpowiednio 2.59, 1.59, 2.64, 2.58 dla algorytmów TS, TS^*, SA, SA^*.

Czasy działania algorytmów opartych na metodzie poszukiwania z zabronieniami zależą od liczności stosowanego sąsiedztwa. Dla algorytmu TS_{ex} jest on najmniejszy, a dla algorytmu TS_{ins} jest największy (nie licząc algorytmu TS). Ponieważ algorytm TS^* przełącza się pomiędzy dwoma tymi sąsiedztwami, jest on zawsze wolniejszy niż algorytm TS_{ex} i szybszy niż algorytm TS_{ins}. Oznacza to, iż w stosunku do konkurencyjnego algorytmu TS, algorytm TS^* jest zawsze szybszy. W przypadku algorytmów opartych na schemacie symulowanego wyżarzania czas działania algorytmu nie zależy od zastosowanego sąsiedztwa. Oznacza to, że algorytmy SA_{ex}, SA_{ins} i SA^* działają w identycznym czasie, zaś algorytm SA pracuje dwukrotnie dłużej.



Rys. 1. Uśrednione błędy względne dla algorytmów TS, TS^*



Rys. 2. Uśrednione błędy względne dla algorytmów SA, SA^*

6. Badania statystyczne

Uzyskane w 5 punkcie wyniki posłużyły do przeprowadzenia analizy statystycznej. Chcąc odpowiedzieć na pytanie, czy algorytm TS^* dostarcza rozwiązań statystycznie lepszych niż algorytm TS, wartości błędów w otrzymanych instancjach potraktowano jako próbkę dwóch zmiennych losowych o rozkładach normalnych i nieznanych wariancjach i nieznanych średnich. Następnie postawiono hipotezę, że wartość średniego błędu uzyskiwana przez algorytm TS^* jest nie mniejsza niż w przypadku algorytmu TS. Tak postawioną hipotezę zweryfikowano przy pomocy testu studenta z typowym dla tego rodzaju testów poziomem istotności $\alpha = 0.05$. Wynik testu wskazuje, że należy odrzucić tę hipotezę na korzyść hipotezy alternatywnej. Jest to potwierdzenie, iż algorytm z

sterowanym otoczeniem dostarcza rozwiązań statystycznie lepszych, w sensie wartości funkcji celu, niż jego klasyczny odpowiednik. Dodatkowo zbadano, iż nawet w przypadku sztucznego powiększenia wszystkich wartości błędów rozwiązań algorytmu TS^* o wartość 0,69%, wartości te dalej są mniejsze w statystycznym sensie (przy poziomie istotności wynoszącym $\alpha = 0.05$), niż odpowiadające im wartości błędów algorytmu TS.

W przypadku algorytmu SA^* niestety nie można powiedzieć tego samego, co przy analizie algorytmu TS^*. Oznacza to, że niewielka przewaga średniej wartości błędu rozwiązania uzyskiwanego algorytmem SA^* nad algorytmem SA może wynikać z losowego przypadku i nie jest istotna w statystycznym sensie.

7. Wnioski

Bardzo małe wartości średnich błędów rozwiązań uzyskiwanych przez testowane algorytmy świadczą o właściwiej ich konstrukcji. Klasyczne algorytmy TS i SA są bardzo dobrej jakości, a test wpływu sterowania otoczenia wykonywany jest w odpowiednich warunkach. Wprowadzenie poprawki polegającej na dodaniu członu sterującego sąsiedztwem, spowodowało znaczną poprawę efektywności algorytmu opartego na technice przeszukiwania zabronieniami i znikomą (nie istotną z statystycznego punktu) poprawę dla algorytmu bazującego na schemacie symulowanego wyżarzania.

Głównym rezultatem pracy jest opracowanie metody zwiększającej wydajność zarówno w sensie pracy działania jak i jakości uzyskiwanych rozwiązań dla algorytmu typu poszukiwania z zabronieniami dedykowanego problemowi $F^* \| C_{sum}$. Dalsza praca powinna dotyczyć weryfikacji efektywności zaproponowanej metody w przypadku zastosowania jej dla algorytmów poszukiwania z zabronieniami dedykowanymi innym problemom optymalizacji dyskretnej.

Literatura

1. Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Optimization and approximation in deterministic sequencing and scheduling: survey, Annals of Discrete Mathematics 5, 1979, 287-326.
2. Smutnicki C., Algorytmy szeregowania, Akademicka Oficyna Wydawnicza ELIT, Warszawa, 2002.
3. Johnson S.M., Optima two-and-tree stage production scheduling with setup limes included, Naval Research Logistics Quarterly 1, 1954, 61-68.
4. Garey M.R., Johnson D.S., Sethi R., The complexity of flowshop and jobshop scheduling, Mathematics of Operations Research 1, 1976, 117-129.
5. Ignall E., Schrage L., Application of the branch and bund technique to some flowshop scheduling problems, Operations research 13, 1965, 400-412.
6. Chung C.S., Flynn J., Kirca O., A branch and bound algorithm to minimize the total flow time form-machine permutation flowshop probles, International Journal of Production Economics 79, 2002, 185-196.
7. Woo H.S., Yim D.S., A heuristic algorithm for mean flowtime objective in flowshop scheduling, Computers & Operations Research 25, 1998, 175-182.

8. Framinan J.M., Leisten R., An efficient constructive heuristic for flowtime minimalization in permutation flowshops, Omega 31, 2003, 311-317.
9. Liu J., Reeves C.R., Constructive and composite heuristic solutions to the $P \parallel \Sigma C_i$ scheduling problem, European Journal of Operational Research 132, 2001, 311-317.
10. Rajendran C., Heuristic algorithm for scheduling in a flowshop to minimalize total flowtime, International Journal of Production Economics 29, 1993, 65-73.
11. Vempati V.S., Chen C.L., Bullngton S.F., An Effective Heuristic for Flow Shop Problems with Total Flow Time as Criterion, Computers and Industrial Engineering 25, 1993, 219-222.
12. Zhang Y., Li X., Wang Q., Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization, European Journal of Operational Research 196, 2009, 869-876.
13. Tseng L., Lin Y., A hybrid genetic local search algorithm for the permutation flowshop scheduling problem, European Journal of Operational Research 198, 2009, 84-92.
14. Gajpal Y., Rajendran C., An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshop, International Journal of Prduction Economics 101, 2006, 259-272.
15. Rajendran C., Ziegler H., Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs, European Journal of Operational Research 155, 2004, 426-438.
16. Jarboui B., Eddaly M., Siarry P., An astimation of distribution algorithm for minimizing the total flowtime in the permutation flow shop scheduling problems, Computers & Operations Research 36, 2009, 2638-2646.
17. Jarboui B., Ibrahim S., Siarry P., Rebai A., A combinatorial particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problems, Computers and Industrial Engineering 54, 2008, 526-538.
18. Glover F., Tabu Search. Part I, ORSA Journal of Computing 1, 1989, 190-206.
19. Glover F., Tabu Search. Part II, ORSA Journal of Computing 2, 1990, 4-32.
20. Kirkpatrick S., Gelatt C.D. Vecchi M.P., Optimisation by simulated annealing, Science 220, 1983, 671-680.
21. Taillard E.: Benchmarks for basic scheduling problems, European Journal of Operational Research, 64, 1993, str. 278-285.

Dr inż. Mariusz MAKUCHOWSKI
Mgr inż. Bartosz WIELEBSKI
Instytut Informatyki Automatyki i Robotyki
Politechnika Wrocławska
53-342 Wrocław, ul. Janiszewskiego 11/17
tel.: (0-71) 320 29 61
e-mail: mariusz.makuchowski@pwr.wroc.pl