Jarosław RUDY, Jarosław PEMPERA, Czesław SMUTNICKI
Politechnika Wrocławska

RÓWNOLEGŁY ALGORYTM TSAB DLA PROBLEMU GNIAZDOWEGO

Streszczenie. W pracy zaproponowano oryginalny wielowątkowy wariant znanego algorytmu TSAB dla gniazdowego problemu szeregowania. Badania wykazały, że zaproponowany algorytm pozwala dwukrotnie zmniejszyć wartość błędu do najlepszych znanych rozwiązań w czasie zdecydowanie krótszym od jego sekwencyjnego odpowiednika. Algorytm dostarcza także górnego ograniczenia minimalnego czasu cyklu w pewnych problemach wytwarzania cyklicznego.

PARALLEL TSAB ALGORITHM FOR THE JOB SHOP PROBLEM

Summary. We propose a multithread variant of the well-known algorithm TSAB for the jobshop scheduling problem. Performed research show that the proposed algorithm is able to reduce twice the error to the best known solutions in the time evidently shorter than its sequencing counterpart. Algorithm provides also an upper bound on the minimal cycle time for a cyclic scheduling problems.

1. Wprowadzenie

Problem gniazdowy (*ang.* Job Shop Scheduling Problem, JSSP) jest jednym ze sztandarowych (klasycznych) problemów szeregowania zadań. Faktycznie, JSSP pozwala modelować wiele praktycznych procesów wytwarzania zbioru niezależnych, różnych detali zgodnie z zamówieniem w tzw. modelu wytwarzania wsadowego (*ang.* batch scheduling). JSSP jest problemem silnie NP-trudnym, co przez długi czas sprawiało poważne trudności w jego rozwiązaniu dla instancji o praktycznym rozmiarze. Spowodowało to duże zainteresowanie przypadkiem JSSP zarówno ze strony praktyków (ewidentne korzyści ekonomiczne), jaki i ze strony badaczy (nowe algorytmy rozwiązywania), prowadząc do pojawienia się licznych interesujących metod rozwiązania.

Jedną z najbardziej znanych i najczęściej stosowanych technologii rozwiązywania problemu gniazdowego jest algorytm TSAB z pracy [13] oraz jego następca *i*-TSAB z pracy [14]. Wysoka skuteczność obu wymienionych algorytmów jest wynikiem wykorzystania kilku nowatorskich koncepcji, mianowicie: (1) metody wyboru dobrego punktu startowego (algorytm INSA), (2) generowania jedynie rozwiązań dopuszczalnych, (3) zastosowania tzw. własności blokowych dla eliminacji rozwiązań nieperspektywicznych, (4) szybkiej metody obliczania wartości funkcji celu, (5) wykorzystania tzw. ubogiego sąsiedztwa, (6) metody skoków powrotnych, (7) metody wykrywania cykli w algorytmie *tabu search* oraz (8) metody dywersyfikacji procesu poszukiwań w oparciu o trajektorie penetrujące/badające przestrzeń rozwiązań. Zauważmy, że TSAB jest algo-

rytmem deterministycznym z kryterium minimalizacji długości uszeregowania.

Celem niniejszej pracy jest poprawa jakości/szybkości wciąż doskonałego numerycznie i teoretycznie ponad 20-letniego już dziś algorytmu TSAB przy efektywnym wykorzystaniu możliwości współczesnych technik obliczeniowych. W szczególności przeprowadzone jest badanie możliwości istotnego zwiększenia jakości/szybkości TSAB poprzez wykorzystanie coraz powszechniejszych środowisk obliczeń równoległych. W pracy badano eksperymentalnie m.in. szybkość metody TSAB i możliwość jej użycia w sytuacjach, w których krótki czas działania algorytmu jest szczególnie istotny. Ponieważ rozwiązanie dostarczane przez TSAB stanowi górne ograniczenie minimalnego czasu cyklu w gniazdowym wariancie zagadnienia wytwarzania cyklicznego, zatem szybki równoległy algorytm TSAB może zostać użyty jako metoda wspomagająca/składowa w ramach pewnego ogólniejszego hybrydowego algorytmu minimalizacji czasu cyklu. Odsyłamy dalej czytelnika do pracy [19] dla szczegółów tego podejścia.

2. Przegląd literatury

W literaturze istnieje szereg alternatywnych metod rozwiązywania problemu gniazdowego, głównie opartych o algorytmy metaheurystyczne, w tym ewolucyjne i hybrydowe (używające TSAB lub jego wariantu do wspomagania procesu poszukiwania). Można wśród nich znaleźć m.in. hybrydowy algorytm ewolucyjny (*ang.* Hybrid Evolutionary Algorithm, HEA) zaproponowany przez Chenga i innych [2]. Algorytm ewolucyjny wykorzystuje w nim procedurę "tabu search" jako poszukiwanie lokalne. HEA dostarcza dobrych jakościowo rozwiązań, ale czas działania dla wielu instancji jest bardzo długi. Z kolei Gonçalves i Resende [4] zaproponowali algorytm genetyczny BRKGA (*ang.* Biased Random-Key Genetic Algorithm) oparty o poszukiwanie lokalne (tabu search) oraz obciążony klucz losowy. Pardalos i Shylo w swojej pracy rozważają algorytm oparty o metodę poszukiwania globalnej równowagi (*ang.* Global Equilibrium Search, GES), [15]. GES jest metodą w wielu aspektach podobną do metody symulowanego wyżarzania i dostarcza dobrej jakości rozwiązań, w czasie krótszym od wielu metod konkurencyjnych. Wykorzystując ideę algorytmu GES i zjawisko wielkiej doliny, Pardalos i inni [16] zaprezentowali algorytm zwany AlgFix. Algorytm ten przejawiał dobry stosunek czasu pracy do jakości rozwiązania dla małych instancji, jednak dla dużych instancji stosowano dość duży limit czasowy 10 000 sekund. Kolejnym podejściem hybrydowym jest algorytm TS-PR (*ang.* tabu search-path relinking) Penga i innych [18], w którym zastosowano metodę pośrednią pomiędzy metodą ścieżek przejściowych i metodą tabu search. Podejście to pozwoliło znaleźć nowe górne ograniczenia dla 49 znanych instancji testowych, co niestety zostało okupione długim czasem pracy algorytmu. Z kolei Zhang i inni zaproponowali szybki algorytm hybrydowy TS/SA [25]. Podejście opiera się na wykorzystaniu symulowanego wyżarzania w celu wygenerowania rozwiązań startowych dla algorytmu tabu search w celu dalszej intensyfikacji poszukiwań.

Z innych podejść, Kurdi [11] zastosował algorytm genetyczny wykorzystujący nowy model wyspowy oraz metodę migracji. Autor przedstawił obszerne wyniki porównujące różne wersje algorytmu i ich wzajemne relacje. Suganthan i inni [20] zaproponowali metodę poszukiwania rojem pszczół (*ang* Artificial Bee Colony, ABC) uzupełnioną o poszukiwanie lokalne dla problemu gniazdowego z ograniczeniami bez czekania (*ang.* no-wait). Wyniki wykazały przewagę metody nad dwoma konkurencyjnymi

algorytmami. Innym podejściem jest praca Asadzadeha [1], gdzie zastosowano system wieloagentowy w celu implementacji różnych metod lokalnego poszukiwania dla algorytmu genetycznego, zwiększając jego skuteczność. Spotykane są też prace poświęcone własnościom sąsiedztw dla problemu gniazdowego. Przykładem jest praca Kuhpfahla i Bierwirtha [10], w której przedstawiono 6 nowych sąsiedztw i ich analizę.

3. Opis problemu

W gniazdowym systemie produkcyjnym składającym się z m maszyn ze zbioru $M = \{1, ..., m\}$ należy wykonać n zadań ze zbioru $J = \{1, ..., n\}$. Zadanie $j \in N$ składa się z sekwencji o_j operacji indeksowanych kolejno przez $(l_{j-1} + 1, ..., l_j)$, które powinny być przetwarzane w takiej kolejności, gdzie $l_j = \sum_{s=1}^{j} o_s$ to suma liczby operacji z pierwszych j zadań ($j = 1, ..., n$ oraz $l_0 = 0$). Zbiór operacji $O = \{1, 2, \dots o\}$, $o = l_n$ reprezentuje zadania będące rozłącznym łańcuchami operacji.

Operacja $i \in O$ musi być wykonywana bez przerw na maszynie $\mu_i \in M$ w czasie $p_i > 0$, $i \in O$. Ze względu na miejsce wykonywania operacji, zbiór operacji O możemy w sposób naturalny podzielić na podzbiory operacji wykonywanych na tej samej maszynie. Oznaczmy przez $O_k = \{i \in O : \mu_i = k\}$ zbiór operacji wykonywanych na maszynie $k \in M$. Tylko jedna operacja danego zadania może być wykonywana w danej chwili. Ponadto, każda maszyna może przetwarzać w danej chwili tylko jedną operację. Harmonogram wykonywania operacji opisujemy przez czasy rozpoczęcia wykonania operacji $S_i \geqslant 0$, $i \in O$, takie, że wszystkie wyżej wymienione ograniczenia są spełnione. Problem polega na znalezieniu możliwego do zrealizowania harmonogramu wykonania operacji, który minimalizuje czas zakończenia realizacji wszystkich operacji $C_{\max}$ tj. minimalizuje funkcję $\max_{i \in O}(S_i + p_i)$.

Harmonogram S w TSAB jest reprezentowany kolejnością wykonywania operacji wyrażoną m-tką permutacji, z których każda jest określona na odpowiednim zbiorze O_k. Wyznaczenie dopuszczalnej kolejności wykonywania operacji na maszynach oraz harmonogramu ich wykonania można sprowadzić do problemu znajdowania najdłuższej drogi w odpowiednio skonstruowanym grafie skierowanym (patrz [13]).

4. Algorytm TSAB

Algorytmy TSAB oraz *i*-TSAB przyczyniły się znaczącego rozwoju przybliżonych metod rozwiązywania problemu gniazdowego w ostatnich 10–20 latach. Dowodem tego jest, oprócz wielu opracowań zawierających porównania z tymi algorytmami, kilkadziesiąt prac czynnie wykorzystujących (lub rozszerzających) różne elementy pochodzące z TSAB oraz *i*-TSAB. Niektóre z tych publikacji omówiono poniżej.

Częstym podejściem jest wykorzystanie sąsiedztwa N5 [1] w innych algorytmach poszukiwania lokalnego (przykładem jest praca [24], w której N5 zastosowano dla algorytmu symulowanego wyżarzania) lub nawet w algorytmach populacyjnych (algorytm genetyczny [3] czy poszukiwanie rojem cząsteczek [5]). Kolejnymi wykorzystywanymi elementami są technika skoków powrotnych [6] oraz procedura wykrywania cykli [25].

Osobną kategorią są prace, wykorzystujące algorytm TSAB w całości lub z niewielkimi modyfikacjami. W pracy [8] TSAB pełni rolę poszukiwania lokalnego dla al-

[1] jest to sąsiedztwo z TSAB wg powszechnie znanej klasyfikacji sąsiedztw

gorytmu genetycznego. Lokalne poszukiwanie z użyciem TSAB ma również miejsce w pracy [4]. W pracy [7] zaproponowano algorytm podobny do TSAB, lecz z innym sąsiedztwem. W pracy [9] wykorzystano algorytm TSA/TSAB, dla którego rozwiązanie początkowe znajduje algorytm mrówkowy.

Na zakończenie warto podkreślić, że w literaturze znajdują się prace w całości poświęcone analizie algorytmu TSAB oraz sąsiedztwa N5 (patrz praca [23]). Pokazuje to wpływ TSAB na rozwój metod rozwiązywania problemu gniazdowego.

5. Równoległe TSAB

Stosowanie środowisk obliczeń równoległych jest w obecnych czasach powszechną praktyką przy projektowaniu algorytmów. Ogólnie, technologie równoległe możemy implementować jako drobno- i/lub gruboziarniste. Odnosząc się do TSAB, do tych pierwszych można by zaliczyć równoległy przegląd sąsiedztwa lub równoległe obliczanie wartości funkcji celu. Są to jednak techniki o ograniczonej skuteczności spowodowanej m.in. niewielkim rozmiarem sąsiedztwa N5 oraz wielokrotnym i względnie krótkim czasem obliczania wartości funkcji celu.

Techniki obliczeń gruboziarnistych są prostsze i bardziej obiecujące do zastosowania w przypadku TSAB. Zauważmy, że jedną z cech TSAB jest występowanie kilku liczbowych parametrów konfiguracyjnych, które zwykle trzeba dobrać eksperymentalnie. Niektóre z nich mają bezpośredni wpływ na czas obliczeń (potencjalnie zwiększając jakość rozwiązań), podczas gdy dla innych takiej zależności nie ma. Dobór parametrów jest zależny od klasy rozwiązywanych instancji i implikuje pewien problem badawczy. W celu uniezależnienia się od trudności strojenia algorytmu skorzystano z możliwości oferowanych przez nowoczesne technologie obliczeń, implementując algorytm TSAB w wielościeżkowej wersji równoległej. Powstały w ten sposób algorytm nazwiemy P-TSAB (*ang.* Parallel TSAB). Przyjęto, że każda ścieżka realizacji algorytmu startuje z tego samego rozwiązania początkowego, natomiast różne ścieżki realizacji algorytmu są uzyskiwane przez dobór innych parametrów algorytmu. Należy zauważyć, że w takiej wersji algorytm ten można w łatwy sposób wykonać na wieloprocesorowych komputerach osobistych, klastrach obliczeniowych oraz w chmurach obliczeniowych. Ponadto, w przypadku użycia odpowiedniej liczby procesorów, czas działania takiego algorytmu jest praktycznie identyczny z czasem działania podstawowego algorytmu TSAB.

6. Badania eksperymentalne

Głównym celem badań eksperymentalnych była ocena efektywności algorytmu P-TSAB oraz możliwości stosowania tego algorytmu jako modułu optymalizacyjnego w systemach informatycznych wspomagających optymalizację harmonogramowania produkcji w przedsiębiorstwach produkcyjnych. Ze względu na podane wymagania praktyczne przyjęto, że czas przebiegu programu nie może być dłuższy od 60 sekund. Dodatkowo, działanie algorytmu przyspieszono przez zastosowanie strategii przeszukiwania otoczenia, w której właściwe wyznaczenie wartości funkcji celu poprzedzone jest szybkim jej oszacowaniem (podobnie jak w pracy [17]).

Algorytm P-TSAB został zakodowany w C++. Badania i pomiar czasu trwania algorytmu zostały przeprowadzone na maszynie wyposażonej w procesor Intel Core i7-980X o częstotliwości 3,33 GHz i posiadającej 12 rdzeni. Testy zostały przeprowadzone na 162 literaturowych instancjach testowych podzielonych na trzy główne grupy. Grupy 2 i 3 składają się z przykładów zaproponowanych odpowiednio przez Lawrence'a [12] i Tailarda [21]. W grupie pierwszej znajdują się przykłady pozostałych autorów.

W trakcie badań wstępnych wyznaczono 42 zestawy parametrów dla algorytmu P-TSAB, dzięki którym możliwe jest równoległe wykonanie 42 trajektorii (ścieżek) przeszukiwań. W celu porównania z podstawową wersją algorytmu TSAB wyznaczono również najlepszy zestaw parametrów (ścieżkę), dla których algorytm TSAB wygenerował najlepsze (w sensie średnim) rozwiązania dla wszystkich grup instancji.

Jakość rozwiązania π_I^A dla algorytmu A i instancji I określono jaką względną odległość wartości funkcji celu $C_{\max}(\pi_I^A)$ od najlepszego rozwiązania dla danej instancji C_I^{ref}. Powstały współczynnik Percentage Relative Deviation (PRD) dany jest wzorem:

$$PRD(\pi_I^A) = \frac{C_{\max}(\pi_I^A) - C_I^{ref}}{C_I^{ref}} \times 100\%. \tag{1}$$

Wartości C_I^{ref} dla poszczególnych instancji zaczerpnięto z suplementu do pracyn[22]. W przypadku algorytmu P-TSAB jako rozwiązanie końcowe

Tabela 1

Wartości PRD dla 1 grupy instancji

Grupa	$n \times m$	TSAB	P-TSAB	HEA	BRKGA	GES	TS-PR	TS/SA
ft06–20		0,00	0,00	0,00	0,00		0,00	
orb01–10	10×10	0,22	0,04		0,01	0,00	0,00	0,17
yn1–4	20×20	1,71	0,47	0,23	0,29		0,22	0,62
swv01–05	20×10	1,80	1,41	0,21	0,32		0,23	1,14
swv06–10	20×15	3,05	1,87	0,51	0,65		0,53	2,10
swv11–15		1,27	0,84		0,28		0,10	
swv16–20		0,00	0,00					
abz5–6	10×10	0,16	0,00					
abz7–9	15×20	1,44	0,75	0,23	0,28		0,20	0,89
średnia		1,07	0,60					

Tabela 2

Wartości PRD dla 2 grupy instancji

Grupa	$n \times m$	TSAB	P-TSAB	HEA	BRKGA	GES	TS-PR	TS/SA
la01–05	10×5	0,00	0,00	0,00	0,00	0,00	0,00	
la06–10	15×5	0,00	0,00	0,00	0,00	0,00	0,00	
la11–15	20×5	0,00	0,00	0,00	0,00	0,00	0,00	
la16–20	10×10	0,11	0,00	0,00	0,00	0,00	0,00	
la21–25	15×10	0,51	0,08	0,00	0,00	0,00	0,00	
la26–30	20×10	0,37	0,22	0,02	0,05	0,00	0,02	
la31–35	30×10	0,00	0,00	0,00	0,00	0,00	0,00	
la36–40	15×15	0,32	0,11	0,01	0,02	0,00	0,00	0,19
średnia		0,16	0,05	0,00	0,01	0,00	0,00	

Tabela 3

Wartości PRD dla 3 grupy instancji

Grupa	$n \times m$	TSAB	P-TSAB	BRKGA	GES	AlgFix	TS-PR	TS/SA
ta01–10	15×15	0,55	0,23	0,05	0,01	0,01	0,01	0,11
ta11–20	20×15	1,26	0,69	0,31	0,10	0,09	0,24	0,65
ta21–30	20×20	1,22	0,57	0,34	0,23	0,07	0,31	0,60
ta31–40	30×15	1,64	0,81	0,26	0,22	0,18	0,22	0,56
ta41–50	30×20	3,19	2,13	0,59	0,69	0,64	0,65	1,23
ta51–60	50×15	0,09	0,04		0,00			
ta61–70	50×20	0,13	0,02		0,01			
ta71–80	100×20	0,01	0,01		0,00			
średnia		1,01	0,56		0,16			

przyjęto najlepsze z rozwiązań wygenerowanych przez wszystkie uruchomienia równoległe algorytmu TSAB.

Wyniki badań dla poszczególnych instancji testowych kolejnych grup głównych przedstawione są w tabelach 1, 2 oraz 3. Z analizy wyników dla pierwszej grupy przykładów testowych wynika, że średni błąd algorytmu TSAB wynosi ok. 1%. Algorytm TSAB znalazł wszystkie rozwiązania optymalne dla przykładów z grup ft06–20 oraz swv16–20. Największy średni błąd obserwuje się w grupie swv06–10 i wynosi 3,05%. Zastosowanie przetwarzania równoległego pozwala na zwiększenie jakości otrzymywanych rozwiązań. W przypadku algorytmu P-TSAB średni błąd wynosi 0,6% i jest prawie dwukrotnie mniejszy niż dla TSAB. Co do wartości bezwzględnej, największą poprawę uzyskano w grupie swv06–10, w której błąd względny zmniejszył się z 3,05 do 1,87%.

Druga grupa przykładów testowych jest w łatwy sposób rozwiązywalna przez algorytm TSAB. Średni błąd tego algorytmu wynosi tylko 0,16% w związku z tym zastosowanie obliczeń równoległych redukuje ten błąd niewiele tj. do 0,05%. Średni błąd algorytmu TSAB w ostatniej grupie instancji podobny jest do średniego błędu w pierwszej grupie i wynosi 1,01%. Najmniejszy błąd obserwuje się w grupach o dużej liczbie operacji tj. ta51–80. Średni błąd algorytmu równoległego P-TSAB wynosi 0,56% i jest prawie dwukrotnie mniejszy od TSAB. Największą, bo prawie 5-cio krotną, poprawę wartości błędu można zauważyć w grupie ta61–70. Z kolei największa poprawa bezwzględna dotyczy grupy ta41–50 i wynosi 1,06%.

Podczas porównania jakości rozwiązań generowanych przez algorytm TSAB (oraz P-TSAB) z algorytmami literaturowymi, należy wziąć pod uwagę następujące utrudnienia: algorytmy testowane były na różnych zestawach plików testowych (autorzy najczęściej pomijali instancje o dużych rozmiarach), na różnych komputerach różniących się parametrami technicznymi. Dodatkowo, nie wszyscy autorzy podawali explicite czas obliczeń (jeżeli był podany to często przekraczał 60 sekund).

Niemniej z analizy rezultatów badań, wynika że jakość generowanych rozwiązań przez algorytm P-TSAB jest porównywalna z jakością rozwiązań generowanych przez najnowsze algorytmy dla problemu gniazdowego. Proste zrównoleglenie tego algorytmu poprawia jakość otrzymywanych rozwiązań. W rezultacie czego średni błąd algorytmu jest mniejszy od błędu algorytmów literaturowych dla wielu grup testowych lub jest gorszy tylko o dziesiąte części procenta.

Tabela 4

Procentowy udział wartości (W) poszczególnych parametrów (P) w końcowych 42 zestawach parametrów

	W_1	W_2	W_3	W_4	W_5	W_6
P_1	4.8%	11.9%	9.52%	**23.8%**	**23.8%**	**26.2%**
P_2	0.0%	0.0%	0.0%	0.0%	0.0%	**100.0%**
P_3	0.0%	0.0%	9.5%	**19.0%**	**31.0%**	**40.5%**
P_4	14.3%	**19.0%**	16.7%	**23.8%**	14.3%	11.9%

Drugim aspektem badań jest wpływ poszczególnych parametrów algorytmu TSAB na wyniki. Dla każdego z 4 parametrów P_1 do P_4 rozważano 6 różnych wartości, oznaczonych symbolami W_1 do W_6, przy czym wartość W_i jest różna dla różnych parametrów. Ponadto wartości każdego parametru są rosnące, tj. zachodzi zależność:

$$W_1 < W_2 < W_3 < W_4 < W_5 < W_6. \tag{2}$$

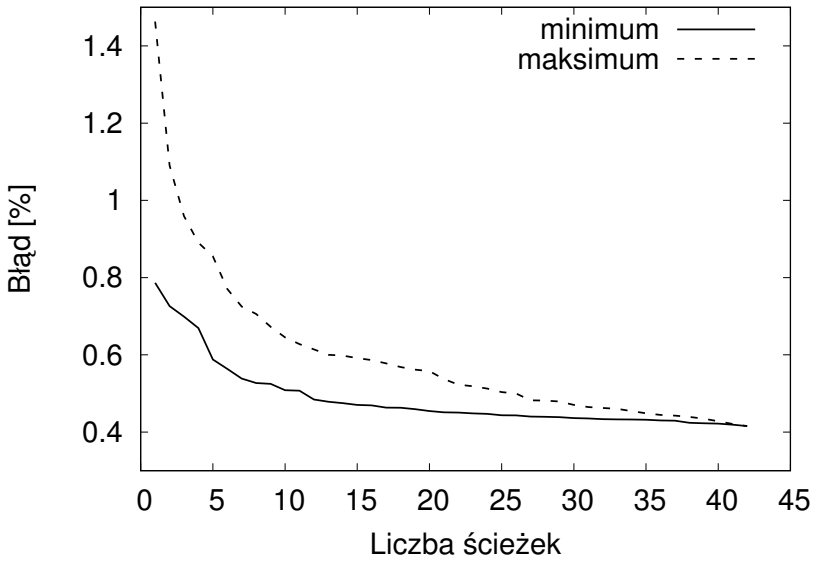
Przykładowo, dla P_3 wartości W_1 do W_6 wynoszą kolejno 3000, 5000, 8000, 10000, 15000 oraz 20000.

Procentowy udział poszczególnych wartości parametrów P_1 do P_4 umieszczony został w tabeli 4. W przypadku parametru P_4 każda z rozważanych wartości miała istotny udział w tworzeniu końcowego zestawu parametrów. Widoczna jest dla tego parametru zależność rosnąca, a następnie malejąca (z maksimum w pobliżu W_4). Z kolei dla parametrów P_1 i P_3 widoczny jest wyraźny wzrost udziału w miarę wzrostu wartości parametru. Jednakże mniejsze wartości parametru wciąż mają znaczenie. Najciekawszy jest parametr P_2, dla którego w końcowych 42 zestawach parametrów znalazła się tylko jedna, największa, wartość. To czy funkcja udziału parametrów P_1, P_2 i P_3 w zależności od wartości jest rosnąca poza rozpatrywanym przedziałem pozostaje kwestią otwartą.

Aby udowodnić korzyść z brania pod uwagę różnych (nie tylko wysokich) wartości parametrów przeprowadzono test. Z końcowych 42 zestawów parametrów usunięto te zestawy, w których znalazły się mniejsze wartości parametrów (niewytłuszczone w tabeli 4). W rezultacie średni błąd algorytmu P-TSAB wzrósł z 0.42% do 0.53% (oznacza to wzrost błędu o ponad 25%).

Ostatnią badaną kwestią był wpływ liczby wątków oraz liczby zestawów parametrów (równoległych ścieżek) algorytmu na wyniki. Poza etapami uruchomienia i zbierania wyników (które mogą być zrealizowane w pesymistycznym czasie $O(\log n)$ przy użyciu n procesorów) algorytm P-TSAB jest w pełni równoległy (współczynnik zrównoleglenia bliski 1). Oznacza to, że dwukrotne zmniejszenie liczby równolegle działających wątków (z 42 na 21) przy zachowaniu liczby ścieżek algorytmu (42), spowoduje dwukrotne spowolnienie algorytmu. Z kolei zastosowanie większej liczby wątków niż liczba ścieżek nie spowoduje dodatkowego przyspieszenia. Ponadto, jakość wyników algorymtu jest niezależna od liczby użytych wątków.

Bardziej skomplikowana jest sytuacja, w której redukcji ulega zarówno liczba wątków jak i liczba ścieżek algorytmu. Rozważmy uruchomienie algorytmu P-TSAB z dwukrotną redukcją liczby wątków i ścieżek (z 42 do 21). Czas działania algorytmu po redukcji będzie zbliżony do czasu działania przed redukcją. Trudność sprawia fakt, że taka redukcja wymaga usunięcia 21 zestawów parametrów, a to można zrobić na wiele sposobów (w tym przypadku jest to $\binom{42}{21} = 5.38 \times 10^{11}$ różnych sposobów).



Rys. 1. Oszacowanie minimalnego i maksymalnego błędu algorytmu P-TSAB w zależności od liczby równoległych ścieżek k

W celu oszacowania błędu algorytmu dla różnej liczby ścieżek przeprowadzono następujący test. Dla danej liczby ścieżek k uruchomiono algorytm P-TSAB dwukrotnie. W pierwszym uruchomieniu wykorzystano k najlepszych (oszacowanie minimum) zestawów parametrów z oryginalnych 42 zestawów (zestaw parametrów jest tym lepszy im mniejszy jest średni błąd algorytmu TSAB przy użyciu tego zestawu parametrów). W drugim uruchomieniu wykorzystano k najgorszych zestawów (oszacowanie maksimum). Wyniki testu zostały przedstawione na rysunku 1.

W przypadku optymistycznym redukcja liczby ścieżek do $k = 30$ spowoduje wzrost błędu dopiero o 5%. W przypadku pesymistycznym wzrost błędu wynosi już 18%. Dla $k = 21$ błąd wzrasta o odpowiednio 9% i 29%. Z przeprowadzonego testu można wywnioskować, że zmniejszenie liczby ścieżek spowoduje wzrost błędu algorytmu, a dla $k < 34$ wzrost ten będzie znaczący.

7. Podsumowanie

W pracy przedstawiono pewną ideę zrównoleglenia algorytmu TSAB dla problemu gniazdowego. Badania TSAB oraz P-TSAB przeprowadzono pod kątem zastosowania ich w komputerowych systemach wspomagania harmonogramowania produkcji, ograniczając czas ich działania do max 60 sekund. Badania testowe przeprowadzone na dużej liczbie literaturowych przykładów jednoznacznie pokazały, że algorytm TSAB w sensie idei nadal należy do czołówki najefektywniejszych algorytmów dla problemu gniazdowego. Zastosowanie przetwarzania równoległego umożliwiło poprawę jakości generowanych rozwiązań oraz w dużym stopniu pozwoliło na uniezależnienie parametrów algorytmu od konkretnej instancji problemu. Opracowany algorytm P-TSAB ze względu na swoją skuteczność i czas działania może być stosowany w komputerowych systemach wspomagania harmonogramowania produkcji oraz jako procedura pomocnicza przy szacowaniu czasu cyklu w systemach wytwarzania cyklicznego.

Podziękowania

Niniejsza praca powstała w wyniku realizacji projektu badawczego OPUS o numerze DEC 2017/25/B/ST7/02181 finansowanego ze środków Narodowego Centrum Nauki.

LITERATURA

1. Asadzadeh, L.: A local search genetic algorithm for the job shop scheduling problem with intelligent agents. Computers & Industrial Engineering 85 (2015), 376–383.
2. Cheng, T. C. E., Peng, B., and Lü, Z.: A hybrid evolutionary algorithm to solve the job shop scheduling problem. Annals of Operations Research 242, 2 (2016), 223–237.
3. Engin, O., Ceran, G., and Yilmaz, M. K.: An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. Applied Soft Computing 11, 3 (2011), 3056–3065.
4. Fernando, G. J., and C., R. M. G.: An extended Akers graphical method with a biased random-key genetic algorithm for job-shop scheduling. International Transactions in Operational Research 21, 2 (2014), 215–246.
5. Gao, L., Li, X., Wen, X., Lu, C., and Wen, F.: A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. Computers & Industrial Engineering 88 (2015), 417–429.
6. González, M. A., Vela, C. R., González-Rodríguez, I., an Varela, R.: Lateness minimization with tabu search for job shop scheduling problem with sequence dependent setup times. Journal of Intelligent Manufacturing 24, 4 (2013), 741–754.
7. Gröflin, H., and Klinkert, A.: A new neighborhood and tabu search for the blocking job shop. Discrete Applied Mathematics 157, 17 (2009), 3643–3655.
8. Huang, D. W., and Lin, J.: Scaling populations of a genetic algorithm for job shop scheduling problems using MapReduce. In IEEE Second International Conference on Cloud Computing Technology and Science (2010), pp. 780–785.
9. Huang, K.-L., and Liao, C.-J.: Ant colony optimon combined with taboo search for the job shop scheduling problem. Computers & Operations Research 35, 4 (2008), 1030–1046.
10. Kuhpfahl, J., and Bierwirth, C.: A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. Computers & Operations Research 66 (2016), 44–57.
11. Kurdi, M.: An effective new island model genetic algorithm for job shop scheduling problem. Computers & Operations Research 67 (2016), 132–142.
12. Lawrence, S.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement). Graduate School of Industrial Administration, Carnegie-Mellon University, 1984.

13. Nowicki, E., and Smutnicki, C.: A fast taboo search algorithm for the job shop problem. Management Science 42, 6 (1996), 797–813.
14. Nowicki, E., and Smutnicki, C.: An advanced tabu search algorithm for the job shop problem. Journal of Scheduling 8, 2 (Apr 2005), 145–159.
15. Pardalos, P. M., and Shylo, O. V.: An algorithm for the job shop scheduling problem based on global equilibrium search techniques. Computational Management Science 3, 4 (Sep 2006), 331–348.
16. Pardalos, P. M., Shylo, O. V., and Vazacopoulos, A.: Solving job shop scheduling problems utilizing the properties of backbone and "big valley". Computational Optimization and Applications 47, 1 (Sep 2010), 61–76.
17. Pempera, J., and Smutnicki, C.: Open shop cyclic scheduling. European Journal of Operational Research 269, 2 (2018), 773–781.
18. Peng, B., Lü, Z., and Cheng, T.: A tabu search/path relinking algorithm to solve the job shop scheduling problem. Computers & Operations Research 53 (2015), 154–164.
19. Smutnicki, C.: Cykliczny problem gniazdowy. In: Zaawansowane modele i algorytmy optymalizacji w systemach cyklicznych, W. Bożejko, J. Pempera, C. Smutnicki, and M. Wodecki, Eds. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2017, ch. 6, pp. 105–160.
20. Sundar, S., Suganthan, P. N., Jin, C. T., Xiang, C. T., and Soon, C. C.: A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint. Soft Computing 21, 5 (Mar 2017), 1193–1202.
21. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 2 (1993), 278–285.
22. van Hoorn, J. J.: The current state of bounds on benchmark instances of the job-shop scheduling problem. Journal of Scheduling 21, 1 (Feb 2018), 127–128.
23. Watson, J.-P., Howe, A. E., and Whitley, L. D.: Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. Computers & Operations Research 33, 9 (2006), 2623–2644.
24. Yang, J., Sun, L., Lee, H. P., Qian, Y., and Liang, Y.: Clonal selection based memetic algorithm for job shop scheduling problems. Journal of Bionic Engineering 5, 2 (2008), 111–119.
25. Zhang, C. Y., Li, P., Rao, Y., and Guan, Z.: A very fast TS/SA algorithm for the job shop scheduling problem. Computers & Operations Research 35, 1 (2008), 282–294. Part Special Issue: Applications of OR in Finance.