dr hab. inż. Tadeusz Witkowski
Paweł Antczak
Wydział Inżynierii Produkcji Politechniki Warszawskiej
Arkadiusz Antczak
Wydział Elektrotechniki i Elektroniki Politechniki Łódzkiej

ZASTOSOWANIE PROCEDURY GRASP DO HARMONOGRAMOWANIA MAŁOSERYJNEJ PRODUKCJI WIELOASORTYMENTOWEJ

W pracy przeanalizowano jedno ogólne zadanie planowania małoseryjnej produkcji wieloasortymentowej, tak nazywany elastyczny problem planowania warsztatowego. Dla rozwiązania tej klasy zadań opracowano algorytm na podstawie procedury GRASP (Greedy Randomized Adaptive Search Procedure), a także przedstawiono schemat szybkiej oceny rozwiązań z sąsiedztwa Nowickiego-Smutnickiego.Przedstawiona analiza pracochłonności obliczeniowej poszczególnych etapów algorytmu. Przedstawiono realizację procedury GRASP dla tego zadania. Opisano wyżej przedstawione sąsiedztwo i przeprowadzono jego modyfikację dla rozwiązywanego zadania. Dla przedstawionego algorytmu opisano wyniki eksperymentu komputerowego.

USED GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE (GRASP) FOR FLEXIBLE JOB SHOP SCHEDULING PROBLEM

The one general task of flexible flexible job shop scheduling has been analysed. The algorithm based on the on procedure GRASP, has been prepared for the class of mentioned task, the schema of fast evaluation of solutions from the neighbourhood Nowickiego-Smutnickiego, ands also has benn presented. Next the realization of GRASP procedure for this task has been presented. The neighbourhood Nowickiego-Smutnickiego has been described and modified for resolving this task. The results of the computer experiment for the algorithm has been presented.

1. WPROWADZENIE

Systemy planowania i sterownia produkcją działają w obszarach związanych z przepływem materiałów i informacji w systemach wytwarzania. Realizują one proces planowania, to znaczy dobór środków do realizacji wyznaczonych zadań

produkcyjnych w zadanym horyzoncie czasowym i osiągnięcia postawionych celów, oraz proces sterowania czyli uruchamianie, nadzorowanie i zapewnienie realizacji zadań produkcyjnych. Są uważane za nadrzędne w obiegu i wymianie danych dla całego procesu wytwarzania. Przydatność narzędzi harmonogramowania dla analizy odpowiednich modeli produkcyjnych zależy od wielkości i charakteru produkcji. Podstawowe metody rozwiązania zadania harmonogramowania małoseryjnej produkcji wieloasortymentowej (flexible job schop scheduling problem) dzielimy na przybliżone i dokładne. Do tego typu zadań w praktyce wykorzystuje się właściwie tylko metody przybliżone. Wśród metod dokładnych wyróżniamy metody podziału i granic, metody rozwiązujące specjalne problemy w czasie wykładniczym czy też metody subgradientowe. Metod przybliżonych jest zdecydowanie więcej niż dokładnych, zwykle są one problemowo-zorientowane. Metody przybliżone dzieli się na algorytmy iteracyjne i algorytmy konstrukcyjne. Do grupy algorytmów konstrukcyjnych zaliczamy reguły priorytetów czy też heurystyki wykorzystujące pojecie „wąskiego gardła". Spośród algorytmów iteracyjnych jedną z dwóch grup tworzą metody sztucznej inteligencji, gdzie wyróżniamy metody spełnienia ograniczeń, sztuczne sieci neuronowe, systemy ekspertowe czy też poszukiwanie mrówkowe. Do drugiej grupy tj. poszukiwania lokalnego zaliczamy m. in.: poszukiwanie z zakazami (tabu serach), poszukiwanie progowe (np. symulowane wyżarzanie), algorytmy genetyczne czy też metody metaheurystyczne typu GRASP.

2. SFORMUŁOWANIE ZADANIA

Ogólnie istota danego zadania zawiera się w następującym. Istnieje pewien zbiór typów części, które należy wytworzyć w ilości określonej przez zamówienia produkcyjne. Dla wytworzenia każdego typu części należy wykonać w określonym technologicznym porządku szereg operacji za pomocą ograniczonego zbioru maszyn. Dla każdej operacji określa się czas niezbędny na jej wykonanie. Każda operacja technologiczna może być wykonana na jednej maszynie spośród grupy technologicznie zamiennych maszyn. Przed rozpoczęciem wykonania operacji maszynę należy wstępnie przezbroić, jednak jeżeli wykonywane są operacje tego samego typu, to przezbrajanie maszyny nie jest wymagane (czas przezbrajania dla poszczególnych operacji jest różny). Należy wybrać dla każdej operacji maszynę i czas jej rozpoczęcia w taki sposób, aby zamówienia były wykonane w wymaganej wielkości, z określonymi ograniczeniami i harmonogram spełniał wybrane kryterium optymalności . W danej pracy jako kryterium optymalności przyjęto kryterium Johnsona (maksymalny czas zakończenia operacji) [1].

Przedstawimy formalne sformułowanie rozwiązywanego zadania, które jest bardziej ogólnym przepadkiem zadania szeregowania dla skonstruowania harmonogramu (flexible job shop scheduling problem) i stanowi jego częściowy przypadek.

Może ono być opisane w następujący sposób. Określono zbiór maszyn M (moc zbioru M oznaczmy przez m), zbiór operacji O, elementami którego są poszczególne technologiczne operacje σ^i, $i = 1..n$, gdzie n – moc zbioru O.

Każdej operacji $\sigma^i \in O$ przyporządkowano podzbiór maszyn $M^i \in M$, które mogą je wykonywać. Zbiór O jest zbiorem częściowo uporządkowanym, tj. określono zbiór następstwa kolejności wykonania $C = \{\sigma^i \prec \sigma^j\}$, który określa kolejność wykonania operacji (« $\sigma^i \prec \sigma^j$ » oznacza, że operacja σ^i powinna być wykonana przed rozpoczęciem wykonania operacji σ^j). Do rozpoczęcia wykonania operacji na maszynie należy przeprowadzić na niej operację przezbrajania. Oprócz tego, wprowadza się klasy operacji tego samego typu k^j, $j = 1...K$, gdzie K ilość klas tego samego typu. Sens klas tego samego typu zawiera się w następującym: jeżeli σ^i i σ^j przynależy do tej samej klasy rodzaju operacji i wykonywane są na jednej maszynie, przy czym po wykonaniu operacji σ^i ta maszyna nie wykonuje operacji do rozpoczęcia wykonania operacji σ^j, to wstępne przezbrajanie maszyny dla wykonania operacji σ^j nie jest wymagane.

Oznaczmy przez $p(\sigma^i)$ czas, niezbędny do wykonania operacji σ^j, $t(\sigma^j)$ – czas niezbędny dla przezbrojenia maszyny przed wykonaniem operacji σ^i, $S(\sigma^i)$, $(F(\sigma^i))$ – czas rozpoczęcia (zakończenia) wykonania operacji σ^i, m^i – maszyna, wybrana z M^i dla wykonania operacji σ^i.

Zadania polega na tym, aby wybrać dla każdej operacji $\sigma^i \in O$ maszynę ze zbioru M^i ($i = 1...n$) i po tym określić kolejność wykonania operacji na maszynach z M, w ten sposób, aby określony harmonogram minimalizował sumaryczny czas wykonania prac (kryterium Johnsona). Zauważmy, że jeżeli wszystkie wielkości $t(\sigma^i)$ są równe zero dla $\sigma^i \in O$, to można określić ograniczenia porządku i podzbiory M^i w taki sposób, aby otrzymać klasyczne sformułowanie zadania planowania (*job shop scheduling* problem). Zadanie określenia harmonogramu wykonania technologicznych operacji rozpatrywane w danej pracy można sformułować w następujący sposób:

$$\min F \tag{1}$$

przy ograniczeniach:

$$F \geq F(\sigma^i), \forall \sigma^i \in O \tag{2}$$

$$F(\sigma^i) \leq S(\sigma^i), \forall \sigma^i \prec \sigma^j \tag{3}$$

$$S(\sigma^i) \geq t(\sigma^i), \forall \sigma^i \in O \tag{4}$$

$$F(\sigma^i) = S(\sigma^i) + p(\sigma^i), \forall \sigma^i \in O \tag{5}$$

$$F(\sigma^i) \leq S(\sigma^j) - t(\sigma^j) \vee F(\sigma^j) \leq S(\sigma^i) - p(\sigma^i),\ \forall \sigma^i, \sigma^j \in O,$$
$$\text{takich, że } m^i = m^j, (\sigma^i \cup \sigma^j) \notin k^l, l = 1...K \tag{6}$$

$$F(\sigma^i) \leq S(\sigma^j) \vee F(\sigma^j) \leq S(\sigma^i),\ \forall \sigma^i, \sigma^j \in O,$$
$$\text{takich, że } m^i = m^j, (\sigma^i \cup \sigma^j) \in k^l, l \in \{1...K\} \tag{7}$$

$$F(\sigma^i), S(\sigma^i) \geq 0 \wedge m^i \in M^i, \forall \sigma^i \in O \tag{8}$$

Ograniczenia 1-2 określają kryterium optymalizacji (kryterium Johnsona). Ograniczenia 3 określają ograniczenie kolejności zgodnie z wykonaniem operacji technologicznych. Ograniczenia 4 wymagają wykonania przezbrojenia maszyny przed rozpoczęciem wykonania operacji. Ograniczenia 5 określają relację między początkiem i

zakończeniem wykonania operacji. Ograniczenia 6,7 przedstawiają ograniczenia na zasoby (maszyna może jednocześnie wykonywać tylko jedną operację), one także uwzględniają czas na przezbrojenie maszyny. Ograniczenia 8 wymagają, aby czasy rozpoczęcia i zakończenia operacji były wartościami nieujemnym i żeby operacje były wykonywane na maszynach z danej grupy technologicznie zamiennych maszyn.

3. METAHEURYSTCZNA PROCEDURA GRASP

W pracy wykorzystano schemat procedury GRASP [2] dla opracowania algorytmu rozwiązania zadania (1)-(8).

Procedura GRASP składa się z dwóch podstawowych etapów: konstruowanie początkowego rozwiązania i przeszukiwania lokalnego. Na etapie konstruowania początkowego rozwiązania generowane jest dopuszczalne rozwiązanie dla zadania (1)-(8) i jego sąsiedztwo badane jest na etapie przeszukiwania lokalnego. Te etapy są powtarzane dopóki nie będzie spełnione kryterium zatrzymania. Najlepsze znalezione rozwiązanie spośród wszystkich iteracji zwracane jest jako wynik zastosowania procedury.

Na etapie konstruowania początkowego rozwiązania najpierw tworzone jest rozwiązanie spełniające ograniczenia zadania. Dla konstrukcji takiego rozwiązania wprowadza się pojęcie kandydata do wstawienia. Dla zadania (1)-(8) kandydatem do wstawienia jest para: operacja i numer maszyny, na którą ona pretenduje.

Oznaczmy *RCL* i *CL*, listę kandydatów do wstawienia, przy czym w liście *CL* będą przechowywani kandydaci już wstawieni do częściowego harmonogramu, a w liście *RCL* – kandydaci pretendujący do wstawienia do tego harmonogramu.

Rozwiązaniem zadania (1)-(8) jest uporządkowana lista *CL*, dla którego: $\forall \sigma^i \in O$ $\exists m \in M^i$ takie, że $(\sigma^i, m) \in CL$; jeżeli $(\sigma^i, m) \in CL$, to nie istnieje $(\sigma^j, m) \in CL$ takie, że $\sigma^i = \sigma^j$ i $m \neq k$. To jest jeżeli $(\sigma^j, m) \in CL$, to oznacza to, że operacja σ^i w harmonogramie, który określany jest listą *CL* będzie wykonywana na maszynie m ($m^i = m$). Dalej mówimy, że operacja σ^i wchodzi do listy *CL*, jeżeli istnieje para $(\sigma^i, m) \in CL$ przy pewnym m.

Oznaczmy $pos(\sigma^i)$ pozycję kandydata związanego z operacją σ^i, na liście *CL*. Przy obliczaniu $S(\sigma^i)$ i $F(\sigma^i)$ jeżeli $pos(\sigma^i) < pos(\sigma^j)$ i $m^i = m^j$, to operacja σ^i występuje przed operacją σ^j. Niech $F(CL) = \max[F(\sigma^i)]$, dla wszystkich σ^i dla których $(\sigma^i, m) \in CL$ przy określonym m. W ten sposób $F(CL)$ oznacza wartość kryterium Johnsona dla harmonogramu określonego listą CL.

Istota etapu konstrukcji początkowego rozwiązania początkowego polega na tym, aby sformułować taka listę kandydatów CL, która przedstawiałaby dopuszczalne rozwiązanie zadania (1)-(8). Na początku etapu konstrukcji rozwiązania lista CL jest pusta, a do listy RCL wprowadzane są warianty wykonania operacji na maszynach,

które mogą być wykonane z uwzględnieniem ograniczeń technologicznych. Następnie dla każdego kandydata $(\sigma^i, m) \in RCL$ obliczana jest wartość funkcji «zachłannej», określonej wartością $F(CL \cup (\sigma^j, m))$.

Oznaczmy $\overline{F}$ ($\underline{F}$) maksimum (minimum) wartości $F(CL \cup (\sigma^j, m))$ dla wszystkich $(\sigma^j, m) \in RCL$. Dalej z listy RCL wybiera się losowo jednego z kandydatów $(\sigma^j, m) \in RCL$, dla którego

$\underline{F} \le F(CL \cup (\sigma^j, m)) \le \underline{F} + \alpha(\overline{F} - \underline{F})$, gdzie α parametr, określający losowość generacji rozwiązań ($\alpha \in [0,1]$). Przy α=1 będzie generowane rozwiązanie absolutnie losowe, a przy α = 0 – rozwiązania otrzymywane za pomocą algorytmu «zachłannego». Jeżeli wybrany jest kandydat $(\sigma^i, m) \in RCL$, to listy CL i RCL są aktualizowane w następujący sposób: $CL = CL \cup (\sigma^i, m)$; $RCL = \cup(\sigma^j, m)$, gdzie $m \in M^j$, a σ^j takie, że dla wszystkich $\sigma^k \prec \sigma^j$ obowiązkowo $(\sigma^k, p) \in CL$ przy określonym p, $(\sigma^j, l) \notin CL$ ale nie dla każdego $l \in M^j$. Nastepnie wybiera się nowego kandydata do listy CL w sposób przedstawiony powyżej. Etap konstrukcji rozwiązania początkowego kończy się kiedy lista RCL jest pusta.

Teraz rozpatrzmy dokładniej, jak jest realizowane obliczanie wielkości $F(CL)$. Dla każdej operacji σ^i określamy zbiór $JS(\sigma^i)$ zawierający operacje σ^j takie, że $\sigma^j \prec \sigma^i$ i nie istnieje operacji σ^l, takiej, że $\sigma^j \prec \sigma^l \prec \sigma^i$. Oznaczymy przez $MP(\sigma^i)$ operację, dla której $m^{MP(\sigma^i)} = m^i \wedge pos(MP(\sigma^i)) < pos(\sigma^i)$ i nie istnieje operacji σ^l, takiej, że $pos(MP(\sigma^i)) \prec pos(\sigma^l) \prec pos(\sigma^i) \wedge m^l = m^i$, jeżeli taka operacja nie istnieje, to $MP(\sigma^i) = \varnothing$. Zbiór *JP* dla każdej operacji pozostaje stałym, a *MP* określa się przy wstawieniu operacji do listy CL. Nazwiemy drogą P w rozwiązaniu, określonym przez listę CL, dowolny uszeregowany zbiór operacji $\sigma^{k_1}, \sigma^{k_2}, ...\sigma^{k_l}$, gdzie l – moc P, $k_i \in \{1...n\}$, przy czym dla σ^{k_i} i $\sigma^{k_{i+1}}$ albo $\sigma^{k_i} \in JP(\sigma^{k_{i+1}})$ albo $\sigma^{k_i} = MP(\sigma^{k_{i+1}})$. Długość drogi $d(P,l) = \{\sigma^{k_1}, \sigma^{k_2}, ...\sigma^{k_l}\}$ określimy następującą zależnością rekurencyjną:

$d(P,1) = p(\sigma^{k_1})$, jeżeli $MP(\sigma^{k_1}) \neq \varnothing$ i $MP(\sigma^{k_1}) \cup \sigma^{k_i} \in k^l$, $l \in \{1...K\}$,

w przeciwnym przypadku $d(P,1) = p(\sigma^{k_1}) + t(\sigma^{k_1})$;

$d(P,i) = d(P,i-1) + t(\sigma^{k_i}) + p(\sigma^{k_i})$, jeżeli $i > 1$, $MP(\sigma^{k_i}) = (\sigma^{k_{i-1}})$ i $MP(\sigma^{k_i}) \cup \sigma^{k_i} \notin k^l$, $\{l = 1...K\}$,

w przeciwnym przypadku $d(P,i) = d(P,i-1) + p(\sigma^{k_i})$.

Wtedy $F(CL)$ będzie równe maksymalnej długości drogi dla CL. Dla $\sigma^i \in O$, które wchodzą do CL czasy zakończenia $F(\sigma^i)$ równają się maksymalnej długości drogi, w której operacja σ^i jest ostatnią ($S(\sigma^i) = F(\sigma^i) - p(\sigma^i)$). Jak widać, przy dodaniu nowego elementu do listy CL, należy ocenić tylko te drogi, które zawierają ostatnią dodaną operację.

Załóżmy, że chcemy otrzymać wielkość $F(CL \cup \sigma^i)$ i dla wszystkich operacji wchodzących do listy CL znane są czasy zakończenia (maksymalna długość drogi, w której ta operacja jest ostatnią). Wtedy dla dowolnej drogi $(...,\sigma^j,\sigma^i)$, składającej się z l elementów, wielkość $d(P, l-1)$ już jest określona i jej długość można obliczyć. A więc jeżeli weźmiemy maksymalną wartość spośród długości takich dróg $P = \{...,\sigma^j,\sigma^i\}$, gdzie albo $\sigma^j \in JP(\sigma^i)$, albo $\sigma^j = MP(\sigma^i)$, albo $P = \{\sigma^i\}$, to otrzymamy maksymalną długość drogi, w której operacja σ^i jest ostatnią ($F(\sigma^i)$). Wtedy $F(CL \cup \sigma^i) = \max[F(CL), F(\sigma^i)]$.

Rozwiązania które generowane są na etapie konstrukcji rozwiązania początkowego, oczywiście nie są lokalnie optymalnymi w stosunku do danego sąsiedztwa, dlatego na ogół stosuje się przeszukiwanie lokalne, aby ulepszyć rozwiązanie początkowe. Przeszukiwanie lokalne kolejno zamienia rozwiązanie bieżące na rozwiązanie lepsze z jego sąsiedztwa (lepsze rozwiązanie rozumiemy w sensie kryterium Johnsona). Etap przeszukiwania lokalnego kończy się kiedy w sąsiedztwie nie zostało znalezione lepsze rozwiązanie.

4. PRZESZUKIWANIE LOKALNE

Na etapie przeszukiwania lokalnego było wykorzystane sąsiedztwo przedstawione w [3,8], które wykorzystuje pojęcie drogi krytycznej. Przyporządkowujemy każdej operacji $\sigma^i \in O$ zbiór $JS(\sigma^i)$, składający się z operacji σ^j takich, że $\sigma^i \in JP(\sigma^j)$; i operację $MS(\sigma^i)$, taką że $MP(MS(\sigma^i))$, jeżeli oczywiście taka operacja istnieje.

Niech rozwiązanie zadania (1)-(8) określone jest przez zbiór CL, jak to opisano wyżej. Dla każdej operacji $\sigma^i \in O$ określimy wartość $tail(\sigma^i)$, która równa się maksymalnej długości spośród dróg $P = \{\sigma^j,...\}$, gdzie $\sigma^j \in JP(\sigma^i)$, albo $\sigma^j = MP(\sigma^i)$. Jeżeli takie drogi nie istnieją, to $tail(\sigma^i) = 0$.

Jak widać z określenia, wartość $tail(\sigma^i)$, pokazuje ile czasu potrzebne jest na zakończenie wszystkich operacji, które nie mogą rozpocząć się przed zakończeniem wykonania operacji σ^i zgodnie z ograniczeniami technologicznymi i porządkiem wykonania operacji.

Operacja $\sigma^i \in O$ jest krytyczną, jeżeli $F(\sigma^i) + tail(\sigma^i) = F(CL)$. To są te operacje, które nie posiadają rezerwy czasowej (luzu czasowego). Droga krytyczna jest to zbiór operacji krytycznych, które nie są wykonywane jednocześnie. Zgodnie z tym określeniem może istnieć kilka krytycznych dróg. Przy realizacji przeszukiwania lokalnego wykorzystywana była jedna droga krytyczna.

Blokiem krytycznym CB nazywa się zbiór operacji krytycznych $\sigma^i \in O$, które wykonuje się na jednej maszynie bez przerw (czas niezbędny na przezbrajanie maszyny nie rozpatruje się jako przerwę). W ten sposób zbiór CB można uporządkować względem czasów początku wykonania operacji. Operacja z najmniejszym (największym) czasem rozpoczęcia wykonania nazywa się pierwszą (ostatnią) operacją bloku krytycznego.

Sąsiedztwo przedstawione w [3], składa się z rozwiązań otrzymanych z rozwiązania początkowego (wyjściowego) na drodze zmiany porządku dwóch pierwszych lub dwóch ostatnich operacji w uporządkowanych zbiorach bloków krytycznych. W realizacji tego sąsiedztwa dla modeli *job shop scheduling* wyklucza się rozpatrzenie rozwiązania otrzymane w wyniku zmiany porządku dwóch pierwszych operacji z bloku krytycznego z najmniejszym czasem rozpoczęcia wykonania pierwszej operacji wchodzącej w ten blok, pod warunkiem, że ten blok obejmuje więcej niż dwie operacje. Wykluczane są też rozwiązania otrzymane w wyniku zmiany porządku dwóch ostatnich operacji z bloku krytycznego z największym czasem rozpoczęcia wykonania pierwszej operacji wchodzącej do niego pod warunkiem, że ten blok obejmuje więcej niż dwie operacje [3]. Jest to związane z tym, że te rozwiązania te nigdy nie są lepsze od początkowego, ale w związku z koniecznością wstępnego przezbrajania maszyn na wykonanie operacji, dla zadania (1)-(8) te rozwiązania mogą być lepsze od początkowego i dlatego uwzględniane były w realizacji przeszukiwania lokalnego. Oprócz tego to sąsiedztwo dla zadania (1)-(8) traci jedną ważną właściwość w porównaniu z zadaniem *job shop scheduling*: rozwiązania z jego sąsiedztwa nie zawsze są dopuszczalnymi. Dlatego zawsze przeprowadzane było sprawdzenie rozwiązania ze względu na dopuszczalność rozwiązania.

5. SCHEMAT SZYBKIEJ OCENY ROZWIĄZAŃ

W [4] przedstawiono schemat pozwalający szybko otrzymywać dolną granicę rozwiązań sąsiedztwa [3].

Przedstawimy jak ten schemat jest realizowany dla modelu *job shop scheduling*. Niech rozwiązanie sąsiedztwa początkowego otrzymuje się przy zmianie porządku operacji σ^i i σ^j ze zbioru O, przy czym $S(\sigma^i) < S(\sigma^j)$. Dolna granica LB tego rozwiązania określa się w następujący sposób: $LB = \max[delta + gaj(\sigma^i), \max[delta, rbj(\sigma^j)] +$

$+ p(\sigma^i) + \max[gaj(\sigma^i), qam(\sigma^j)]]$,

gdzie: $rbj(\sigma^j) = \max(F(\sigma^k))$, maksimum bierze się względem wszystkich σ^k takim, że $\sigma^k \in JP(\sigma^j)$, a jeżeli lista $JP(\sigma^j)$ jest pusta, to $rbj(\sigma^j) = 0$;

$rbj(\sigma^i) = \max(F(\sigma^k))$, maksimum bierze się względem wszystkich σ^k takim, że $\sigma^k \in JP(\sigma^i)$, a jeżeli lista $JP(\sigma^i)$ jest pusta, to $rbj(\sigma^i) = 0$;

$rbm(\sigma^i) = F(MP(\sigma^i))$, a jeżeli brak operacji $MP(\sigma^i)$, to $rmb(\sigma^i) = 0$;

$delta = \max(rbj(\sigma^j), rbm(\sigma^i)) + p(\sigma^j)$;

$qaj(\sigma^j) = \max(tail(\sigma^k) + p(\sigma^k))$, maksimum bierze się względem wszystkich σ^k takim, że $\sigma^k \in JS(\sigma^j)$, a jeżeli lista JS jest pusta, to $qaj(\sigma^j) = 0$;

$qaj(\sigma^i) \max(tail(\sigma^k) + p(\sigma^k))$, maksimum bierze się względem wszystkich σ^k takim, że $\sigma^k \in JS(\sigma^i)$, a jeżeli lista JS jest pusta, to $qaj(\sigma^i) = 0$;

$qam(\sigma^j) = tail(MS(\sigma^j)) + p(MS(\sigma^j))$, a jeżeli brak operacji $MS(\sigma^j)$, to $qam(\sigma^j) = 0$.

Jednak ta granica nie uwzględnia czasu na przezbrajanie maszyn. Dlatego, aby ta granica była prawidłową dla zadania (1)-(8) należy wprowadzić następujące poprawki do obliczeń:

1) $rbm(\sigma^i) = F(MP(\sigma^i))$, jeżeli istnieje operacja $MP(\sigma^i)$ taka, że $MP(\sigma^i) \cup \sigma^j \in k^l, l \in \{1...K\}$; $rbm(\sigma^i) = F(MP(\sigma^i)) + t(\sigma^j)$, jeżeli $MP(\sigma^i)$ występuje, ale $MP(\sigma^i) \cup \sigma^j \notin k^l$, $l = 1...K$; $rmb(\sigma^i) = 0$, jeżeli operacja $MP(\sigma^i)$ nie występuje;

2) $delta = \max(rbj(\sigma^j), rbm(\sigma^i)) + p(\sigma^j)$, jeżeli $\sigma^i \cup \sigma^j \in k^l$, $l \in 1...K$, w przeciwnym przypadku $delta = \max(rbj(\sigma^j), rbm(\sigma^i)) + p(\sigma^j) + t(\sigma^i)$;

3) Analogicznie do poprzednich poprawek należy uwzględnić przezbrajanie maszyn przy obliczaniu wielkości $qaj(\sigma^j)$, $qaj(\sigma^i)$ i $qam(\sigma^j)$.

6. EKSPERYMENT KOMPUTEROWY

Dla rozwiązania zadania (1)-(8) opracowano oprogramowanie dla algorytmu, opartego na procedurze GRASP. Eksperymenty komputerowo przeprowadzono dla danych przedstawionych w [1]. Liczba operacji dla tych danych - 160, liczba maszyn – 26. Wszystkie eksperymenty przeprowadzono w wykorzystaniem komputera z procesorem Pentium 733MHz i 256MB pamięci operacyjnej.

Najbardziej pracochłonnym etapem algorytmu dla rozwiązania zadania okazał się etap konstrukcji rozwiązania w ramach procedury GRASP. Na etap konstrukcji przypadło 98% czasu pracy algorytmu i 2% na etap przeszukiwania lokalnego. Dlatego nie patrząc na to, że schemat szybkiej oceny rozwiązania dla sąsiedztwa [3] zaadaptowany dla

zadania (1)-(8) pozwolił przyspieszyć lokalne przeszukiwanie średnio 10 razy, to istotnego przyspieszenia przebiegu algorytmu nie udało się otrzymać. Czas zużyty na jeden etap konstrukcji rozwiązania wynosił średnio 55 sekund. Jeśli chodzi o efektywność lokalnego przeszukiwania, to średnie polepszenie rozwiązań na etapie konstrukcji rozwiązań wyniosło średnio 1,85%.
Parametr α, który wykorzystuje się do wyboru kandydatów do wstawienia do harmonogramu na etapie konstrukcji rozwiązania wybierany był dla każdego etapu konstrukcji losowo z przedziału [0,1]. Najlepsze rozwiązanie znaleziono po 100 iteracjach rowne 26 938,1min.
Przy stałym $\alpha = 0$ (algorytm «zachłanny») najlepsze rozwiązanie dla przebiegu szeregowo-równoległego otrzymano po 100 iteracjach procedury GRASP .
Dla rozwiązania tego zadania stosowany był algorytm genetyczny, najlepsze znalezione rozwiązanie – 32115 min., i algorytm SZEZA, najlepsze znalezione rozwiązanie – 35222 min 5,6]. Wyniki z zastosowaniem GRASP do szeregowego przebiegu produkcji przedstawiono w innym referacie tego opracowania.

6. WNIOSKI

Dla ogólnego typu zadania elastycznego planowania produkcji małoseryjnej (*Flexible Job Shop scheduling*) - wiele maszyn dla jednej operacji, przedstawiono algorytm bazujący na procedurze GRASP. Przeanalizowano modyfikację sąsiedztwa Nowickiego-Smutnickiego [3] dla danego zadania. Eksperyment obliczeniowy pokazał, że najbardziej pracochłonną częścią algorytmu jest etap konstrukcji rozwiązania w odróżnieniu od klasycznego zadania harmonogramowania typu *Job-Shop Scheduling* (jedna maszyna dla danej operacji).
Przedstawiono schemat szybkiej oceny rozwiązania dla sąsiedztwa Nowickiego-Smutnickiego, który pozwolił przyspieszyć przeszukiwanie lokalne około 10 razy. Efektywność realizowanego przeszukiwania lokalnego okazała się nie bardzo wysoka. Dlatego dla danego zadania należy skonstruować bardziej optymalne sąsiedztwo, które zawierało by rozwiązania, w których operacje realizowane są na różnych maszynach w porównaniu z początkowym rozwiązaniem. Jednak, głównymi wadami realizowanego algorytmu jest niezależność poszczególnych iteracji i wysoka pracochłonność konstrukcji rozwiązania. Pierwszą wadę można usunąć, wprowadzając do algorytmu schemat intensyfikacji poszukiwania rozwiązania w obszarze najlepszego rozwiązania, tak jak to zrealizowano w [3], lub stosując procedurę path-relinking [7]. W celu usunięcia drugiej wady, przy rozwiązywaniu sformułowanego zadania można zastosować inne metody np. typu TABU, która nie wymaga stałej generacji nowych rozwiązań. Ale metoda Tabu Search także nie wykorzystuje informacji otrzymanej w procesie przeszukiwania, oprócz mechanizmu w postaci zakazanych ruchów. W [8] przedstawiono algorytm, który wykorzystuje procedurę path-relinking dla wdrożenia intensyfikacji przeszukiwania w sąsiedztwie najlepszych rozwiązań. To podejście może być bardzo obiecujące dla rozwiązania danego zadania. Jednak dla jego wykorzystania należy opracować sąsiedztwo, uwzględniające rozwiązania w których przydział operacji do maszyn różni się od rozwiązania początkowego.

LITERATURA

[1]. *Witkowski T.*, Decyzje w zarządzaniu przedsiębiorstwem, WNT, Warszawa 2000.

[2] *Binato S., Hery W.J., Loewenstern D.M., Resende M.G.* A greedy Randomized Adaptive Search Procedure For Job Shop Scheduling, AT&T Labs Technical Report 2000, pp.1-119.

[3]. *Nowicki E., Smutnicki C.* A Fast Taboo Search Algorithm for the Job Shop Problem, Manag. Sci. – 1996. – № 42(6). – P. 797–813.

[4]. *Taillard E.* Parallel taboo search techniques for the job shop scheduling problem. ORSA J. on Computing 6, 108-117.

[5]. *Witkowski T., Antczak A., Antczak P.* Random and Evolution Algorithms of Tasks Scheduling and the Production Scheduling. Proceedings of International Joint Conference on Fuzzy Systems –FUZZ –IEEE 2004, Budapest 2004, vol. 2, pp.727-732.

[6]. *Witkowski T., Antczak P., Strojny G.* The Implementation of Neural Network for the Optimization of the Production Scheduling. Proceedings of III International Conference on Advances in Production Scheduling – APE 2004, Warsaw 2004, part I, pp.121-130.

[7]. *R.M.Aiex, S.Binato, M.G.C.Resende.* Parallel GRASP with path-relinking for job shop scheduling. Parallel Computing, 29:393-430, 2003.

[8]. *Nowicki E., Smutnicki C.* Some new tools to solve the job shop problem. Wydawnictwo Politechniki Wrocławskiej, Wrocław 2002.