mgr inż. Paweł Antczak
mgr inż. Arkadiusz Antczak
dr hab. inż. Tadeusz Witkowski
Wydział Inżynierii Produkcji Politechniki Warszawskiej

ZASTOSOWANIE ALGORYTMU SYMULOWANEGO WYŻARZANIA DO OPTYMALIZACJI HARMONOGRAMOWANIA PRODUKCJI MAŁOSERYJNEJ

W pracy przedstawiono jedną z klas ogólnego problemu szeregowania zadań. Do rozwiązania tego problemu opracowano algorytm symulowanego wyżarzania. Eksperymenty były porównane z wynikami otrzymanymi przy użyciu innych algorytmów.

USED SIMULATED ANNEALING ALGORITHM FOR OPTIMIZATION JOB SHOP SCHEDULING PROBLEM

One class of problem, that presents general case of scheduling problem. To solve this class of problem an algorithm was created based on simulated annealing procedure. Experiments were compared with results obtained with used of other algorithms.

1. WPROWADZENIE

Zbadano jedną z klas zadania harmonogramowania, przedstawiające ogólny przypadek zadania harmonogramowania z określonym porządkiem wykonania operacji na grupach technologicznie zamiennych maszyn. Dla rozwiązania tej klasy zadań opracowano algorytm na podstawie algorytmu symulowanego wyżarzania. Przedstawiono sformułowanie rozpatrywanego zadania i wprowadzono umowne oznaczenia poszczególnych wielkości. Przedstawiono podstawowe etapy opracowanego algorytmu jego parametry. Przeprowadzono eksperymenty i opisano otrzymane wyniki porównując je z wynikami otrzymanymi za pomocą innych algorytmów przy rozwiązywaniu testowego zadania.

2. SFORMUŁOWANIE ZADANIA

Przedstawimy sformalizowane sformułowanie rozwiązywanego zadania, która jest bardziej ogólnym analogiem zadania harmonogramowania (flexible job shop scheduling problem). Matematycznie ona może być przedstawiona w następujący sposób.

Określono zbiór maszyn M (moc M oznaczymy przez m), zbiór operacji O, elementami którego są poszczególne operacje technologiczne σ^i, $i = 1..n$, gdzie n – moc zbioru O. Każdej operacji $\sigma^i \in O$ przypisano podzbiór maszyn $M^i \in M$,

które mogą je wykonywać. Zbiór operacji O jest częściowo uszeregowany. Oznaczmy to częściowe uszeregowanie symbolem π. Z punktu widzenia ograniczeń technologicznych dla operacji σ^i, $\sigma^i \in O$ relacja («$\sigma^i \pi \sigma^j$» oznacza, że operacja σ^i powinna być wykonana przed rozpoczęciem operacji σ^j). Na zbiorze O także określono relację równoważności j. Operacje σ^i, $\sigma^i \in O$, związane relacją równoważności j, będziemy nazywać równoważnymi. Relacja j pozwala podzielić zbiór O na klasy ekwiwalentne. Oznaczmy przez $K(\sigma^i)$ klasę elementów ze zbioru O operacji σ^i. Niech dla operacji σ^i: $p(\sigma^i)$ – czas, niezbędny dla wykonania operacji (jednakowy dla wszystkich maszyn), $t(\sigma^j)$ – czas, niezbędny dla przezbrojenia maszyny przed wykonaniem danej operacji.

Zadanie opracowania harmonogramu polega na tym, aby wybrać dla każdej operacji $\sigma^i \in O$ maszynę ze zbioru $M^i (i = 1..n)$ i następnie określić sekwencję wykonania operacji na maszynach ze zbioru M, w taki sposób, aby określony harmonogram minimalizował sumaryczny czas wykonania wszystkich prac (kryterium Johnsona).

Oznaczmy przez $S(\sigma^i)$ – czas rozpoczęcia wykonania operacji technologicznej σ^i, $F(\sigma^i) = S(\sigma^i) + p(\sigma^i)$ – czas zakończenia jej wykonania, m^i – maszyna wybrana ze zbioru M^i dla wykonania operacji σ^i. Wtedy matematyczne sformułowanie zadania opracowania harmonogramu dla wykonania operacji technologicznych, rozpatrywanych w danej pracy będzie mieć następującą postać:

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

przy ograniczeniach:

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

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

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

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

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

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

Ograniczenia (1),(2) określają kryterium optymalizacji (kryterium minimalizacji wykonania wszystkich zadań – kryterium Johnsona). Ograniczenia (3) określają ograniczenia kolejności zgodnie z technologicznym porządkiem wykonania. Ograniczenia (4) wymagają wykonania przezbrojenia maszyny przed rozpoczęciem wykonania operacji. Ograniczenia (5), (6) przedstawiają ograniczenia na zasoby (maszyna może wykonywać jednocześnie tylko jedna operację). Te ograniczenia także uwzględniają czas niezbędny na przezbrajanie maszyn przy wykonaniu operacji. Ograniczenia (7) wymagają, aby operacje były wykonywane na odpowiednich, wstępnie określonych maszynach technologicznie zamiennych z danej grupy.
W pracy [1] przedstawiono charakterystykę algorytmu symulowanego wyżarzania. Niżej przedstawiono zastosowanie tego algorytmu do harmonogramowania w przypadku istnienia grup technologicznie zamiennych maszyn.

3. PRZEDSTAWIENIE ROZWIĄZANIA

Rozwiązania zadania (1)-(7) można poszukiwać w postaci uporządkowanego zbioru F_{queue} par w postaci (σ^i, m), gdzie $\sigma^i \in O$, $m \in M^i$. Ten zbiór powinien posiadać następujące właściwości:

1) $\forall \sigma^i \in O$, $m \in M^i$ takich, że $(\sigma^i, m) \in F_{queue}$,

2) jeżeli $(\sigma^i, m) \in F_{queue}$, że nie istnieje $l \in M^i$ takich, że $l \neq m$ i $(\sigma^i, l) \in F_{queue}$.

Właściwość 1) wymaga, żeby każda operacja była przedstawiona w zbiorze F_{queue} w parze z numerem maszyny, która może ją wykonywać, a właściwość 2) – żeby każda operacja wchodziła w zestaw tylko jednej pary, tj. moc zbioru F_{queue} .równa jest n .

Określając zbiór F_{queue}, możemy określić wielkości $S(\sigma^i)$ dla $\sigma^i \in O$ i otrzymać w ten sposób dopuszczalne rozwiązanie zadania (1)-(7).

Niech $(\sigma^i, m^i) \in F_{queue}$, wtedy oznaczymy $A1(\sigma^i) = \max[F(\sigma^j), 0]$, $\sigma^j \pi \sigma^i$; $A2(\sigma^i) = \max[F(\sigma^j), 0]$, $\forall (\sigma^i, m^i) \in F_{queue}$ takich, że $(\sigma^j, m^i) = (\sigma^i, m^i) \vee \sigma^j \in K(\sigma^i)$; $A3(\sigma^i) = \max[F(\sigma^j) + t(\sigma^i), 0]$, $\forall (\sigma^i, m^i) \in F_{queue}$ takich, że $(\sigma^j, m^i) = (\sigma^i, m^i) \vee \sigma^j \notin K(\sigma^i)$.

Określając te zależności rekurencyjne można obliczyć $S(\sigma^i)$ dla wszystkich $\sigma^i \in O$: $S(\sigma^i) = \max[A1(\sigma^i), A2(\sigma^i), A3(\sigma^i), t(\sigma^i)]$. Optymalne rozwiązanie dla przyjętego kryterium optymalności wchodzi do zbioru wszystkich możliwych rozwiązań, które można określić w podobny sposób.

Niech $F(F_{queue}) = \max[F(\sigma^i)]$, dla wszystkich $\sigma^i \in O$, gdzie wielkości $F(\sigma^i)$ obliczane są na podstawie wzorów rekurencyjnych określonych wyżej na zbiorze F_{queue}. W ten sposób, $F(F_{queue})$ oznacza maksymalny czas zakończenia operacji wchodzących do zbioru O. Jeżeli operacje te wykonywać zgodnie z harmonogramem

przedstawionym przez zbiór F_{queue}, tj. wartość rozwiązania będzie równa F_{queue}. Aby taki harmonogram był dopuszczalny wystarczającym warunkiem jest spełnienie właściwości 1), 2).

Dla każdej operacji $\sigma^i \in O$ określimy zbiór $JP(\sigma^i)$, zawierający operacje $\sigma^j \in O$ takie, że $\sigma^j \text{ p } \sigma^i$ i nie istnieje operacja $\sigma^l \in O$, taka, że $\sigma^j \text{ p } \sigma^l \text{ p } \sigma^i$. Oznaczymy przez $MP(\sigma^i)$ operację, dla której $(\sigma^i, m^i), (MP(\sigma^i), m^i) \in F_{queue}$, $(MP(\sigma^i), m^i) = (\sigma^i, m^i)$ oraz nie istnieje operacja $\sigma^l \in O$, taka, że $(\sigma^l, m^i) \in F_{queue}$ i $(MP(\sigma^j), m^i) = (\sigma^l, m^i) = (\sigma^i, m^i)$. Zbiór *JP* dla każdej operacji pozostaje niezmienionym, a *MP* zależy od konkretnego rozwiązania określonego przez zbiór F_{queue}. Dalej oznaczmy przez $JP(\sigma^i)$ zbiór operacji $\sigma^l \in O$, dla których $\sigma^i \in JP(\sigma^l)$ oraz $MS(\sigma^i)$ – operację dla której $MP(MS(\sigma^i)) = \sigma^i$.

Drogą P w rozwiązaniu F_{queue} będziemy nazywać sekwencję operacji $\{\sigma^{k_1}, \sigma^{k_2}, ..., \sigma^{k_l}\}$ taką, że dla wszystkich $\sigma^{k_i}, \sigma^{k_{i+1}} \in O$ albo $\sigma^{k_i} \in JP(\sigma^{k_{i+1}})$, albo $s^{k_i} = MP(s^{k_{i+1}})$, gdzie $i \in \{1,2,...,l-1\}$.

Dla drogi o maksymalnej długości P_{MAX} w rozwiązaniu F_{queue} spełniona jest równość $d(P_{MAX}) = F(F_{queue})$ i dla każdej operacji $\sigma^i \in O$, $S(\sigma^i)$ jest równe długości maksymalnej drogi spośród wszystkich dróg o postaci $P = \{..., \sigma^l\}$, gdzie albo $\sigma^l \in JP(\sigma^i)$, albo $\sigma^l = MP(\sigma^i)$.

Dla każdej operacji $\sigma^i \in O$ określimy wielkość $tail(\sigma^i)$, która równa się maksymalnej długości spośród dróg $P = \{\sigma^l, ...\}$, gdzie albo $\sigma^l \in JS(\sigma^i)$, albo $\sigma^l = MS(\sigma^i)$. Jeżeli takie drogi nie istnieją, to $tail(\sigma^i) = 0$. Operacja $\sigma^i \in O$ nazywana jest krytyczną w rozwiązaniu F_{queue}, jeżeli należy ona do drogi o maksymalnej długości tj. krytyczna droga – jest to droga o maksymalnej długości. Blokiem nazywamy część drogi krytycznej $\{\sigma^{k_1}, \sigma^{k_{i+1}}, ..., \sigma^{k_{i+c}}\}$, takiej, że $m^{k_i} = m^{k_{i+1}} = ..m^{k_{i+c}}$, σ^{k_i} – pierwsza operacja bloku, $\sigma^{k_{i+c}}$ – ostatnia operacja. Blok, który nie jest podzbiorem żadnego innego bloku nazywany jest blokiem krytycznym.

4. PRZESZUKIWANIE LOKALNE

Istotę metody wyżarzania symulowanego, podobnie jak wielu innych algorytmów wykorzystujących przeszukiwanie lokalne, stanowi operacja zamiany rozwiązania na

rozwiązanie z tego sąsiedztwa. Pod sąsiedztwem zwykle rozumiemy zbiór rozwiązań otrzymanych za pomocą określonych elementarnych przekształceń nad rozwiązaniem wyjściowym. Większość sąsiedztw zaproponowanych dla zadań harmonogramowania, w których jako kryterium optymalności występuje kryterium Johnsona, wykorzystuje określenie drogi krytycznej.

Różne rodzaje sąsiedztw opisano w [1]. Sąsiedztwo zaproponowane w [7], składa się z rozwiązań, otrzymanych z rozwiązania wyjściowego poprzez zmianę porządku uszeregowania dwóch pierwszych lub dwóch ostatnich operacji w blokach krytycznych. Oznaczmy $N^{NS}(x)$ zbiór rozwiązań z tego sąsiedztwa dla rozwiązania x. Główną wadą tego sąsiedztwa dla zadania (1)-(7) jest to, że ono nie uwzględnia możliwości wykonania operacji na kilku technologicznie zamiennych maszynach. Dlatego też, stosując lokalne przeszukiwanie, wykorzystujące to sąsiedztwo, dla zadania (1)-(7) wybór maszyn do wykonania operacji pozostaje niezmiennym. Aby usunąć tę wadę rozszerzymy to sąsiedztwo, dodając do niego rozwiązania, otrzymane w wyniku przeniesienia operacji krytycznej na inną maszynę. Ale jeżeli podejmiemy decyzję o przeniesieniu operacji σ^i na inną maszynę, którą oznaczymy przez m^i, to powinniśmy także określić porządek, w jakim będą wykonywane operacje na nowej maszynie m^i po takiej zamianie. Nawet, jeśli zachowamy porządek wykonania operacji na nowej maszynie, a operację σ^i będziemy wstawiać między te operacje, to liczba rozwiązań w tym sąsiedztwie będzie bardzo duża. Dlatego dla nowego sąsiedztwa należy ograniczyć liczbę wariantów poprzez dokładny i efektywny wybór miejsca wstawienia σ^i do uszeregowania wykonania operacji na nowej maszynie.

Niech rozwiązanie x będzie określone poprzez uporządkowany zbiór F_{queue}. Zmieniamy kolejność elementów $(\sigma^i, m^i) \in F_{queue}$ tak, że dla dowolnych par $(\sigma^i, m^i)\ (\sigma^j, m^j) \in F_{queue}$ przy $S(\sigma^i) < S(\sigma^j)$ spełniony jest warunek $(\sigma^i, m^i) = (\sigma^j, m^j)$. Wtedy rozwiązanie odpowiadające F_{queue} po takiej zmianie kolejności nie zmienia się. Oznaczymy przez $N^{NEW}(x)$ zbiór rozwiązań, które odpowiadają zbiorowi uporządkowanych zbiorów, otrzymanych z F_{queue} poprzez zamianę pary (σ^j, m^j) na parę (σ^j, l^j), gdzie σ^j – operacja krytyczna, a $l^j \in M^j$ oraz $l^j \neq m^j$, a uszeregowanie pozostaje poprzednim. W takim przypadku wszystkie rozwiązania przynależące do $N^{NEW}(x)$ będą dopuszczalnymi, z powodu wstępnie przeprowadzonemu przeszeregowaniu F_{queue}. W ten sposób określamy sąsiedztwo dla rozwiązania x, które w odróżnieniu od sąsiedztwa $N^{NS}(x)$ nie posiada wad przedstawionych wyżej.

Niech rozwiązanie $y \in N^{NEW}(x)$ otrzymuje się ze zbioru CL w wyniku zamiany (σ^j, m^j) na (σ^j, l^j). Oznaczymy w rozwiązaniu x: $\sigma^{OMP} = MP(\sigma^j)$,

$\sigma^{OMS} = MS(\sigma^j)$, a w rozwiązaniu y: $\sigma^{NMP} = MP(\sigma^j)$, $\sigma^{NMS} = MS(\sigma^j)$. Przy przejściu od rozwiązania x do rozwiązania y należy koniecznie uaktualnić wartości MS oraz MP dla tych operacji:

$MS(\sigma^{OMP}) = \sigma^{OMS}$, $MP(\sigma^{OMS}) = \sigma^{OMP}$, $MS(\sigma^{NMP}) = \sigma^j$,

$MP(\sigma^{NMS}) = \sigma^j$, $MS(\sigma^j) = \sigma^{NMS}$, $MP(\sigma^j) = \sigma^{NMP}$.

Następnie określimy wartości $S(\sigma^k)$ oraz $tail(\sigma^k)$ dla wszystkich operacji σ^k wchodzących do uszeregowania F_{queue} po zamianie. W tym celu należy znaleźć nowe wartości $S(\sigma^j)$ dla wszystkich operacji σ^k takich, że $(\sigma^k, m^k) = (\sigma^j, m^j)$, a dla pozostałych operacji one się nie zmieniają. Nowe wartości $tail(\sigma^k)$ należy określić tylko dla operacji σ^k takich, że mamy albo $(\sigma^k, m^k) = (\sigma^j, m^j)$, albo $(\sigma^k, m^k) = (\sigma^{OMS}, m^{OMS})$, albo $(\sigma^k, m^k) = (\sigma^{NMS}, m^{NMS})$.

Najbardziej pracochłonnym etapem lokalnego przeszukiwania jest ocena rozwiązań z danego sąsiedztwa. Jeżeli należy wybrać z sąsiedztwa tylko rozwiązanie, które jest lepsze od pewnego wyjściowego, to większość przeprowadzonych obliczeń będzie nieefektywna. Dlatego też bardzo użytecznym jest określenie właściwej wartości dolnej granicy rozwiązań z sąsiedztwa. W pracy do realizacji lokalnego przeszukiwania w sąsiedztwie $N^{NEW}()$ wykorzystano schemat szybkiej oceny rozwiązań przedstawiony w [12].

5. ALGORYTM SYMULOWANEGO WYŻARZANIA

W pracy dla rozwiązania zadania harmonogramowania przedstawiono algorytm wykorzystujący metodę symulowanego wyżarzania [13, 14]. Niżej przedstawiono ogólny schemat tego algorytmu do opracowania harmonogramu.

Parametry: K – liczba zmian temperatury
L - liczba iteracji ze stałą temperaturą
T - temperatura początkowa
r – parametr zmniejszenia temperatury
freq - częstość lokalnego przeszukiwania w sąsiedztwie N^{NEW}

Określić wejściowe parametry algorytmu
$k = 1$
while $k <= K$ **do**
for *iter* = 0 do L **do**
losowo wybrać rozwiązanie $y \in N^{NS}(x)$
if $f(y) < f(x)$ **then**
przejście do rozwiązania y, tj. $x = y$
else
przejście do rozwiązania y z prawdopodobieństwem
exp [- $\{f(y) - f(x)\}$/ T]

```
    end if
  if iter % freq = 0 then
        { % - reszta od dzielenia}
     improvement = true
     while improvement do
       if istnieje rozwiązanie y ∈ N^NEW(x) takie, że
f(y) < f(x) then
      x = y
      improvement = true
      else
       improvement = false
      end if
     end while
  end if
 end for
 k = k + 1
 T = T * r
end while
```

Rys 1. Metoda symulowanego wyżarzania do opracowania harmonogramu

6. PARAMETRY ALGORYTMU SYMULOWANEGO WYŻARZANIA

W procedurze symulowanego wyżarzania występuje wiele parametrów, których wartości muszą być ustalone. Brak jest jednak ogólnych metod ich doboru. Przyjmuje się, że początkowa temperatura T_0 powinna być wystarczająco wysoka, tak aby zagwarantować odpowiednio dużą wartość prawdopodobieństwa p_o zaakceptowania ruchów, które nie prowadzą do poprawy rozwiązania. Można przyjąć wartość p_o = 0,9, chociaż niektórzy autorzy proponują tylko wartość 0,4.

Duży wpływ na czas przetwarzania mają współczynnik schładzania r oraz liczba iteracji ze stała temperaturą. Często przyjmuje się r = 0,95 oraz L =16.

W [14] proponuje się, aby temperatura T_k była ustalana na podstawie następującej funkcji:

$$T_k = T_0 e^{-ck},$$

gdzie c –stała dodatnia, k- numer kroku algorytmu.

W eksperymentach komputerowych ważnym zagadnieniem jest określenie warunków zatrzymania algorytmu:

1). Procedura jest zatrzymywana, jeżeli F(H) nie uległa poprawie o przynajmniej p_1 % po K_1 kolejnych seriach L kroków;

2). Procedura jest zatrzymywana, jeżeli liczba możliwych do zaakceptowania ruchów jest mniejsza niż p_2 % po K_2 kolejnych seriach L kroków.

Ponadto zaleca się, aby kontrolować prawidłowość wyboru parametrów przez upewnienie się, że algorytm nie został zatrzymany zbyt wcześnie, to znaczy w fazie, gdy wartość F(H) ciągle jeszcze spada.

W przedstawionym algorytmie sąsiedztwo $N^{NS}(g)$ wykorzystuje się dla realizacji cyklu temperatury algorytmu symulowanego wyżarzania. Korzystne przejścia z tego

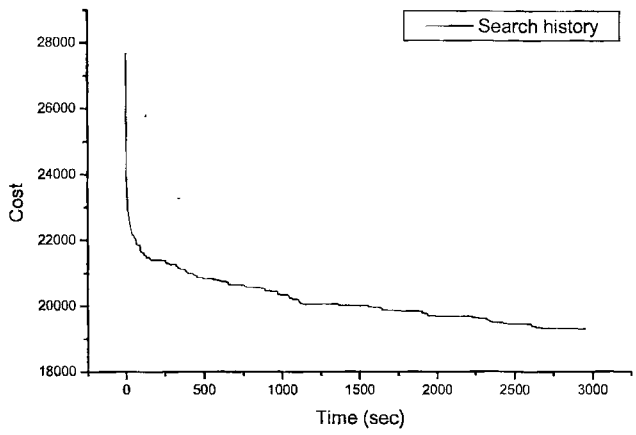
sąsiedztwa są od razu stosowane do rozwiązania, a niekorzystne stosowane są z prawdopodobieństwem zależnym od bieżącej temperatury i stopnia pogorszenia rozwiązania. Każda liczba *freq* iteracji przeprowadza lokalne przeszukiwanie w sąsiedztwie $N^{NEW}(g)$, co pozwala istotnie polepszyć otrzymywane rozwiązania.

Wykonane badania testowe [5, 8] dla różnych wariantów algorytmu symulowanego wyżarzania z różnymi otoczeniami i z kryterium minimalnego czasu obróbki, przy różnych metodach strojenia algorytmu, wykazały umiarkowany optymizm w ocenach jakość rozwiązania / czas obliczeń. Z tego też względu kolejne prace badawcze zmierzały w kierunku algorytmów hybrydowych, otrzymanych głównie poprzez wbudowanie dodatkowej procedury przeszukiwania lokalnego (pełny przegląd subotoczenia) poprzedzającej wybór końcowego rozwiązania. Podobne podejście zastosowano w pracy.

7. EKSPERYMENT KOMPUTEROWY

Dla rozwiązania zadania (1)-(7) opracowano oprogramowanie realizujące algorytm wykorzystujący metodę symulowanego wyżarzania, szczegóły którego przedstawiono wyżej. Eksperymenty obliczeniowe przeprowadzone dla danych przedstawionych w [9] pokazały dużą efektywność tego algorytmu. Liczba operacji dla tych danych wynosi 160, a liczba maszyn 26. Eksperymenty komputerowe były przeprowadzone z wykorzystaniem komputera z procesorem Pentium 733 MHz z 256 MB pamięci operacyjnej. Były wykorzystane następujące parametry algorytmu: K=40, L=100, T=100, r=0.9, *freq*=10. Jako początkowe rozwiązanie przyjmowano najlepsze rozwiązanie znalezione na wszystkich iteracjach. Na rys. 1 przedstawiono wyniki rozwiązania testowego zadania z [9]. Najlepsze znalezione rozwiązanie – 19284.7 minut. Dokładne obliczenia tego samego zadania dla innych algorytmów, m.in. sieci neuronowych, algorytmów genetycznych, algorytmów losowych zaprezentowano w [10, 11, 12]. W pracy dla ilustracji przedstawiono wyniki (tab. 1) otrzymane za pomocą procedury GRASP [2, 12]. Jak widać z tabl.1 najlepszą wartość kryterium F (H) przy stosunkowo małej liczbie iteracji (równej 80) otrzymano w przypadku algorytmu GRASP dla 40 iteracji w jednej z prób. Prowadzone badania dla tej samej struktury danych dla szeregowo-równoległego przebiegu produkcji z wykorzystaniem m.in. algorytmu hybrydowego typu GRASP+TS (metoda tabu) pokazują, że można uzyskać wartość kryterium F (H) o kilka a nawet o kilkanaście procent lepszą w porównaniu z „czystymi" algorytmami GRASP.

Przy obliczeniach początkowe rozwiązanie było generowane za pomocą algorytmu zachłannego. Na rys. 2 zwraca uwagę szybkie polepszenie funkcji kryterialnej $F(H)$ w pierwszej fazie eksperymentów komputerowych (w naszym przypadku podczas pierwszych 200 sekund). Dalszy przebieg obliczeń pokazuje stopniowe, lecz nieznaczne polepszenie wartości funkcji kryterialnej.



Rys. 2. Wartości $F(H)$ w zależności od czasu obliczeń dla algorytmu SA

Tabela 1. Wartości $F(H)$ w zależności od liczby iteracji dla algorytmu GRASP

iter.	Wartości kryterium $F(H)$ dla poszczególnych prób					Średni czas $F(H)$ (T/5 prób)
	1 próba	2 próba	3 próba	4 próba	5 próba	
80	33904,1	33991,2	33887,4	33631,0	33654,4	33813,6
70	33989,3	33975,3	34081,5	33707,2	**33633,2**	33877,3
60	33647,9	33849,0	**33850,5**	**33490,5**	34031,7	**33773,9**
50	33644,2	33976,3	33865,8	33843,8	33768,9	33819,8
40	**33576,9**	34085,4	33949,0	33803,8	33714,0	33825,8
30	33951,5	33827,4	34293,0	34014,4	34097,3	34036,7
20	34128,7	**33729,9**	34076,4	33967,1	33967,1	33973,8
10	34080,1	34262,8	34388,1	34066,4	34087,1	34176,9
9	34211,8	34209,4	34306,6	34108,2	33861,5	34139,5
8	34319,4	33859,2	34345,9	33657,2	34280,8	34092,5
7	34164,6	34543,5	34190,7	34185,9	34386,4	34294,2
6	33965,0	33813,6	34450,5	34352,8	34731,0	34262,6
5	34315,9	33998,3	34416,5	34433,4	34477,6	34328,3
4	34100,3	34347,0	34746,5	33898,1	34234,3	34265,2
3	34893,3	34554,8	34512,9	34741,7	34700,8	34680,7
2	35065,7	34204,5	34416,3	34811,8	34717,2	34643,1
1	34713,3	34865,2	35059,9	34637,2	34306,3	34716,4

LITERATURA

[1] Antczak A., Antczak P., Witkowski T., Algorytm symulowanego wyżarzania – efektywna metoda optymalizacji harmonogramów produkcji, Automation 2006, PIAP, Warszawa 2006.

[2] Binato S. i in., *A Greedy Randomized Adaptive Search Procedure For Job Shop Scheduling,* AT&T Labs Technical Report, 2000 pp.1-19.

[3] Feo T. i in., A GRASP For A Difficult Single Machine Scheduling Problem. *Computers Ops. Res.*, Vol. 18, No 8, 1990, pp. 635-643.

[4] Fleurent C., Glover F., Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory, *INFORMS Journal on Computing,* 11, 1999, pp. 198-204.

[5] Janiak A., *Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobó,*. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.

[6] Michalewicz Z., *Algorytmy genetyczne,* WNT, Warszawa 1996.

[7] Nowicki E., Smutnicki Cz., A Fast Taboo Search Algorithm for the Job Shop Problem, *Management Science*, vol. 42, No 6, 1996, pp. 797-813.

[8] Smutnicki Cz., *Algorytmy szeregowania,* Akademicka Oficyna Wydawnicza EXIT, Warszawa 2000.

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

[10] Witkowski T. i in., *Use Constraint Satisfaction Adaptive Neural Network For Job-Shop Scheduling,* Automation 2004, PIAP Warszawa 2004, s. 277-284.

[11] Witkowski T., Antczak P., Antczak A., 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.

[12] Witkowski T., Antczak A., Antczak P., Zastosowanie procedury GRASP do harmonogramowania małoseryjnej produkcji wieloasortymentowej, Automation 2005, PIAP, Warszawa 2005, s. 116 - 125.

[13] Van Laarhoven P., Aarts E, and Lenstra J., Job shop scheduling by simulated annealing, Operations Research, 40 (1992), pp. 113-125.

[14] Yamada T., Rosen B. E., Nakano R., A A simulated approach to job-shop scheduling using critical block transition operators. Proceedings of the 1st IEEE Conference on Evolutionary Computation, Florida 1994, pp. 4687-4692.