PROBLEM HARMONOGRAMOWANIA ZADAŃ WIELOMASZYNOWYCH

Karolina BOROWICKA, Wojciech BOŻEJKO, Łukasz KACPRZAK, Mieczysław WODECKI

Streszczenie: W pracy rozpatrujemy problem harmonogramowania zadań wykonywanych na wielu identycznych maszynach z pewnym kryterium kosztowym, którym jest iloczyn czasu wykonywania zadań oraz liczby użytych maszyn. Rozwiązanie problemu sprowadza się do pewnego dwuwymiarowego problemu pakowania. Przedstawiamy algorytm symulowanego wyżarzania z różnymi wariantami strategii pakowania. Przeprowadzone eksperymenty obliczeniowe wykazały, że wyznaczone rozwiązania niewiele różnią się od górnych ograniczeń wartości funkcji celu.

Słowa kluczowe: równoległe maszyny, zadania wielomaszynowe, kryterium kosztowe.

1. Wprowadzenie

W zdecydowanej większości rozpatrywanych w literaturze problemów szeregowania zakłada się, że zadanie jest wykonywane na pojedynczej maszynie. Istnieją jednak takie rzeczywiste procesy produkcyjne, a w szczególności współczesne systemy komputerowe, w których wykonanie zadania wymaga jednocześnie więcej niż jednego procesora (maszyny). Mówimy wówczas o systemach z maszynami równoległymi i zadaniach wielomaszynowych (wieloprocesorowych). W literaturze takie problemy były już od dość dawna rozpatrywane, np. przeglądowa praca Drozdowskiego [4] oraz monografia [5], a także praca doktorska Kramera [7], które głównie dotyczą systemów komputerowych. Założenie dotyczące wielomaszynowości zadań może być realizowane na różne sposoby:

- zadnie może być wykonane na dowolnym podzbiorze wymaganej liczbie maszyn (np. Makuchowski [11]),
- ustalone są podzbiory maszyn (sposoby wykonania) spośród których należy dokonać wyboru (np. Nowicki [10]).

Pierwsze zastosowania wielomaszynowych modeli szeregowania zadań odnoszą się do przemysłu chemicznego (Bozoki i Richard [3]) oraz harmonogramowania projektów (Vizing [13]). Naturalnym odniesieniem dla tego typu zagadnień są badane od kilkudziesięciu lat wieloprocesorowe systemy komputerowe (Błażewicz i in. [1]). W przeważającej większości optymalizowano termin wykonania wszystkich zadań (C_{max}). Obecnie wiele centrów komputerowych oferuje wykonywanie obliczeń przy czym koszt wykonania usługi jest zależny, przede wszystkim, od czasu trwania obliczeń oraz liczby wymaganych procesorów. Istnieje więc potrzeba budowy oraz badania nowych, odpowiadających obecnej rzeczywistości modeli, a także analizy funkcji kryterialnych uwzględniających całokształt kosztów. Z podobnymi zagadnieniami spotykamy się także przy planowaniu przedsięwzięc budowlanych, gdzie jednoczesne korzystanie z wielu zasobów jest podstawą organizacji pracy.

Podstawowy problem szeregowania zadań wielomaszynowych, $P \mid size_j \mid C_{max}$ (gdzie

P jest symbolem wielomaszynowości, a $size_j$ - liczbą wymaganych maszyn) był badany głównie pod kątem algorytmów przybliżonych oraz analizy najgorszego przypadku (Lin [8], Lloyd [9]). Udowodniono bowiem, że już w przypadku dwóch maszyn, problem $P2 \mid size_j \mid C_{max}$ jest *NP-trudny*. Błądek i in. [2] rozpatrują problem z kryterium C_{max} zakładając dodatkowo, że do wykonania pewnych zadań należy przydzielić ustaloną liczbę sąsiednich maszyn. Z kolei obszerny rozdział w pracy Nowickiego [10] jest poświęcony problemowi gniazdowemu z równoległymi maszynami. Przedstawiono model grafowy oraz algorytm oparty ma metodzie przeszukiwania z tabu, a także wyniki przeprowadzonych eksperymentów obliczeniowych. Kryterium kosztowe (iloczyn długości uszeregowania i liczby użytych maszyn) jest rozpatrywane w pracy Rudek i in. [12]. Dodatkowo przyjęto założenie, że czas wykonywania zadania podlega efektowi uczenia się bądź starzenia, w zależności od momentu rozpoczęcia jego wykonywania.

Kryteria uwzględniające, oprócz czasu pracy maszyn, także inne parametry systemu niewątpliwie umożliwiają dokładniejsze wyznaczenie rzeczywistych kosztów realizowanego przedsięwzięcia. W pracy rozpatrujemy kryterium kosztowe zależne od czasu wykonywania zadań i liczby użytych maszym.

2. Sformułowanie problemu

Problem harmonogramowania zadań wielomaszynowych można sformułować następująco.

Problem: dany jest zbiór zadań $J = \{1,2,\ldots,n\}$, które należy wykonać na identycznych maszynach (tj. o tych samych własnościach funkcjonalnych i jednakowych wydajnościach) ze zbioru $M = \{1,2,\ldots,m\}$. Zadanie $i \in J$ wymaga jednocześnie do wykonania $size_i$ maszyn przez okres $p_i > 0$. Muszą być przy tym spełnione następujące ograniczenia:

a) żadna maszyna nie może w dowolnym momencie wykonywać więcej niż jedno zadanie,
b) wykonywanie zadania nie może być przerwane,
c) każde zadanie jest wykonywane na wymaganej liczbie maszyn.

Zakładamy ponadto, że liczba maszyn $m \geq \sum_{i=1}^{n} size_i$.

Rozpatrywany w pracy problem sprowadza się do wyznaczenia, dla każdego zadania, podzbioru maszyn oraz momentów rozpoczęcia jego wykonywania (na każdej z maszyn) spełniających ograniczenia (*a*)-(*c*), aby zoptymalizować przyjęte kryterium.

Każde rozwiązanie dopuszczalne Θ (tj. spełniające ograniczenia (*a*)-(*c*)) może być reprezentowane przez parę $\Theta = (Q,S)$, przy czym:

- $Q = [Q_1, Q_2, ..., Q_n]$, gdzie Q_i $(Q_i \subseteq M)$ jest zbiorem maszyn (przydziałem) na których będzie wykonywane zadanie $i \in J$, $\mid Q_i \mid = size_i$,
- $S = (S_{1,1}, ..., S_{1,m}, S_{2,1}, ..., S_{2,m}, ..., S_{n,1}, ..., S_{n,m})$, gdzie element $S_{i,j}$ jest momentem rozpoczęcia wykonywania zadania i na maszynie j (jeżeli $j \notin Q_i$, wówczas przyjmujemy $S_{i,j} = -\infty$).

Oznaczmy przez Π zbiór wszystkich rozwiązań dopuszczalnych.

Dla dowolnego rozwiązania $\Theta \in \Pi$

$$C_{\max}(\Theta) = \max\{S_{i,j} + p_i : \ i \in J, j \in M\}$$

jest momentem zakończenia wykonywania wszystkich zadań, a

$$M_{\max} = \max\{j : j \in Q_i, i \in J\}$$

maksymalnym numerem maszyny spośród wszystkich przydzielonych do wykonywania zadań. Niech

$$F(\Theta) = C_{\max}(\Theta) \cdot M_{\max}(\Theta). \tag{1}$$

Rozpatrujemy problem harmonogramowania zadań (w skrócie oznaczany przez **MP**) polegający na wyznaczeniu rozwiązania $\Theta^* \in \Pi$ takiego, że

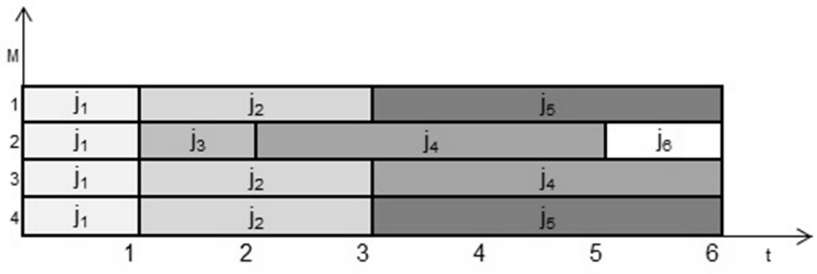
$$F(\Theta^*) = \min\{F(\Theta) : \ \Theta \in \Pi\}.$$

Przykład 1. Dany jest zbiór wielomaszynowych zadań $J = \{1,2,3,4,5,6\}$ do wykonania na $m = 4$ równoległych maszynach ze zbioru $M = \{1,2,3,4\}$. Parametry poszczególnych zadań są przedstawiono w tabeli 1.

Tab. 1. Parametry zadań.

zadane (j)	1	2	3	4	5	6
p_j	1	2	1	3	3	1
$size_j$	4	3	1	2	2	1

Na rysunku 1 przedstawiono diagram Gannta pewnego harmonogramu Θ, dla którego $M_{\max}(\Theta) = 4,\ C_{\max}(\Theta) = 6,$ a wartość funkcji celu $F(\Theta) = 24.$



Rys. 1. Harmonogram rozwiązania problemu szeregowania zadań wieloprocesorowych.

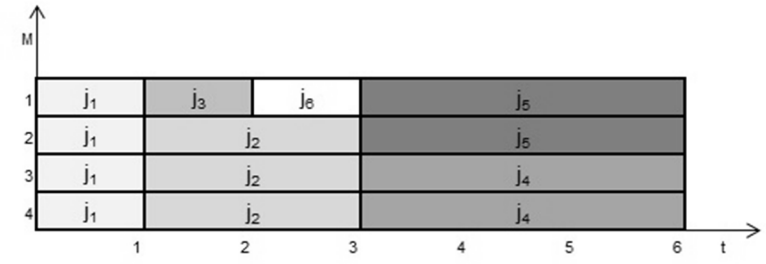
Maszyny $l,k \in M$ $(l \neq k)$ nazywamy *sąsiednimi*, jeżeli $k = l+1$ $(l = 1,2,...,m-1)$ lub $k = l-1$ $(l = 2,3,...,m)$. Wprowadzamy nowe ograniczenie:

(***c'***) każde zadanie jest wykonywane na wymaganej liczbie **sąsiednich** maszyn.

Problem harmonogramowania zadań z ograniczeniami (*a*), (*b*), (*c'*) oraz funkcją celu zdefiniowaną w (1) będziemy nazywali problemem prostokątnego harmonogramowania zadań wielomaszynowych i w skrócie oznaczali przez **MPP**. W tym przypadku funkcję celu oznaczamy przez $F^{\otimes}(\Theta)$.

Na rysunku 2 przedsytawiono rozwiązanie dla prostokątnego harmonogramowania

zadań z przykładu 1. W tym przypadku, podobnie jak w przykładzie 1, wartość funkcji celu $F^{\otimes}(\Theta) = 24$. Można łatwo udowodnić, że jeżeli dla pewnego przykładu danych, Θ^* jest rozwiązanie optymalnym problemu **MP**, a Ω^* rozwiązaniem optymalnym problemu **MPP**, to $F^{\otimes}(\Omega^*) \geq F(\Theta^*)$. Pewne dolne oraz górne oszacowania rozpatrywanego kryterium przedstawiono w pracy [6].



Rys. 2. Harmonogram rozwiązania problemu prostokątnego szeregowania zadań.

Rozwiązanie problemu prostokątnego harmonogramowania zadań wieloprocesorowych można łatwo sprowadzić do pewnego problemu dwuwymiarowego pakowania.

3. Dwuwymiarowy problem pakowania

Problem dwuwymiarowego pakowania, w rozważanym wariancie, polega na takim rozmieszczeniu dostępnych, dwuwymiarowych obiektów (prostokątów), aby zajmowany przez nie obszar posiadał minimalne pole powierzchni. Bardziej formalnie.

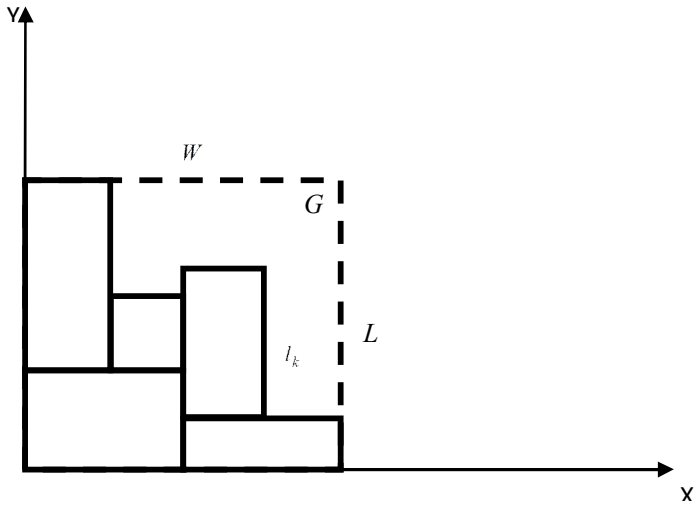
Problem: dany jest zbiór prostokątów $P = \{p_1, p_2, ..., p_n\}$ oraz typów $T = \{t_1, t_2, ..., t_n\}$. Każdy prostokąt $p_i \in P$ posiada określony typ u_i, definiujący jego wymiary, tj. $u_i = t_k$, dla $k \in \{1,2,...,m\}$, $i = 1,2,...,n$, $u \in U = \{u_1, u_2, ..., u_n\}$. Dowolny typ $t_k \in T$ definiuje para liczb $t_k = (l_k, w_k)$, gdzie l_k oznacza długość a w_k szerokość prostokąta.

Rozpatrywany problem polega na znalezieniu obszaru o minimalnym polu powierzchni, wewnątrz którego możliwe jest umieszczenie wszystkich dostępnych obiektów, tj minimalizacja iloczynu,

$$\Phi(G) = L \cdot W,$$

gdzie L jest długością, a W szerokością obszaru G.

Z każdym prostokątem, znajdującym się wewnątrz obszaru G, utożsamiona jest pozycja, na której został on umieszczony reprezentowana przez współrzędne w kartezjańskim układzie współrzędnych. Prostokąty należy umieścić wewnątrz obszaru G w taki sposób, aby na siebie nie nachodziły a ich krawędzie, reprezentujące odpowiednio ich długość i szerokość, położone były równoległe do krawędzi reprezentujących długość i szerokość obszaru G. Przykład ilustrujący wyżej opisane definicje i oznaczenia przedstawiono na rys. 3.



Rys. 3. Obszar G wraz z zapakowanymi prostokątami.

4. Strategie pakowania oraz algorytm

W tym rozdziale opisano metodę reprezentacji rozwiązania, przedstawiono schemat algorytm symulowanego wyżarzania, różne strategie pakowania, a także schemat działania algorytmu wraz z przyjętą strategią pakowania.

4.1. Rozwiązanie

Rozwiązanie reprezentowane jest jako chromosom, będący ciągiem (permutacją) prostokątów $p_i \in P$. Na jego podstawie, przy zastosowaniu strategii pakującej, ustalana jest rzeczywista postać rozwiązania (współrzędne prostokątów wewnątrz obszaru G, jego pole powierzchni oraz dokładność rozwiązania δ). Przykładowy chromosom dla $n = 7$ prostokątów przedstawiono rys. 4.

p_2	p_5	p_4	p_3	p_1	p_7	p_6

Rys. 4. Chromosom

Sekwencja prostokątów odzwierciedla kolejność, w jakiej będą one rozpatrywane przez odpowiednią strategię pakującą.

4.2. Algorytm symulowanego wyżarzania

Algorytm symulowanego wyżarzania (SA) został zastosowany do wyznaczania nowych rozwiązań (chromosomów), z których korzystamy umieszczając prostokąty wewnątrz obszaru stosując, w tym celu jedną z opisanych strategii pakowania.

Kluczowymi elementami algorytmu są: temperatura początkowa, temperatura końcowa, funkcja prawdopodobieństwa z którym akceptowane są rozwiązania gorsze od bieżącego oraz schemat schładzania (zmniejszania wartości) temperatury.

W każdym kroku algorytmu, rozpatrywane są rozwiązania tworzące sąsiedztwo bieżącego rozwiązania. Sąsiedztwo jest zbiorem elementów, generowanych poprzez niewielkie modyfikacje rozpatrywanego aktualnie rozwiązania. W przypadku, gdy rozwiązanie bieżące jest gorsze od najlepszego z sąsiedztwa, zostaje ono nim zastąpione. Z pewnym prawdopodobieństwem jest możliwe również zastąpienie aktualnego rozwiązania rozwiązaniem gorszym. Ma to na celu zapobieganie stagnacji i kierowaniu trajektorii poszukiwań w nowe obszary przestrzeni rozwiązań. Dobór temperatury, schematu chłodzenia i funkcji prawdopodobieństwa wpływa na częstość akceptowania rozwiązań gorszych od bieżącego, a więc na zdolność algorytmu do opuszczania minimów lokalnych jak i stabilność poszukiwań.

W zaimplementowanym algorytmie symulowanego wyżarzania sąsiedztwo generowane jest na dwa sposoby:

- poprzez losową zamianę miejscami wybranych pól chromosomu, oraz
- zapisaniu wybranego podciągu elementów chromosomu w odwrotnej kolejności.

4.3. Strategie pakowania

W konstrukcji algorytmu zastosowano cztery strategie umieszczania prostokątów wewnątrz tworzonego obszaru.

- minimalnego powiększania powierzchni obszaru,
- współczynnika zależności pole-wymiary,
- zasady ruletki,
- XYP.

Ogólny schemat działania strategii pakujących, w postaci pseudokodu, został przedstawiony na rys. 5. W każdej z zastosowanych strategii dążymy do umieszczenia prostokątów wewnątrz obszaru o możliwie najmniejszym polu powierzchni, rozpatrując je zgodnie z kolejnością ich występowania w chromosomie.

Pętla w linii 5 jest wykonywana aż do momentu umieszczenia wewnątrz obszaru wszystkich prostokątów. Kluczowym elementem jest lista pozycji, służąca do przetrzymywania aktualnie dostępnych współrzędnych rozważanych przy poszukiwaniu miejsca, w którym aktualnie rozpatrywany prostokąt p_i może być umieszczony.

W kroku pierwszym (i:= 0) na liście pozycji umieszczana jest para (0,0) będąca początek układu współrzędnych (linia 9). Jeżeli wstawienie rozważanego prostokąta jest możliwe, pozycja *j* zostaje usunięta z listy (Aktualizacja), a na jej miejsce dodawane są dwie nowe, powstałe w wyniku umieszczenia prostokąta p_i. Pozycje te mają współrzędne: $(x_j + w_i, y_j)$, $(x_j, y_j + w_i)$ i są nazywane punktami referencyjnymi bądź skrajnymi (ang. *referencepoints*, *extremepoints*). Następnie, gdy pozostały jeszcze pewne nieumieszczone prostokąty, to są one rozpatrywane zgodnie z kolejnością występowania w chromosomie. Wcześniej jednak sortuje się zaktualizowaną listę dostępnych pozycji, sprawdzając jednocześnie możliwości wstawienia. Po rozpatrzeniu wszystkich prostokątów następuje zakończenie działania procedury.

Metoda sortowania listy aktualnych pozycji zależy od stosowanej strategii (linia 12). Każda ze strategii sortuje aktualnie dostępne pozycje według innego kryterium. Jako

właściwa pozycja, dla rozważanego prostokąta jest wybierana pierwsza, umożliwiająca wstawienie prostokąta, bez kolizji z już tam wcześniej umieszczonymi.

```
1 Obliczenia wstępne:
2   n := liczba dostępnych prostokątów;
3   p_i := prostokąt rozważany w kroku i.
4
5 For ( i := 0; i < n; i := i+1 )
6  {
7   If ( i = 0 )
8   {
9      lista_pozycji.dodaj(0,0);
10  }
11
12  Dla prostokąta p_i sortuj lista_pozycji; j:=0;
13  wstawienie = false;
14
15  While (wstawienie = false)
16  {
17      If ( p_i pasuje na pozycję j)
18      {
19         Aktualizacja;
20         wstawienie := true;
21      }
22      Else
23      {
24         j := j+1;
25      }
26
27  }
```

Rys. 5. Ogólny schemat strategii umieszczania prostokątów wewnątrz obszaru

Zastosowanie strategii *minimalnego obszaru* powoduje wyznaczenie, dla każdej aktualnie dostępnej pozycji *j*, potencjalnego pola powierzchni tymczasowego obszaru powstałego na skutek wstawienia rozważanego prostokąta p_i na pozycję *j* w chromosomie. Następnie, pozycje generujące upakowanie o mniejszym polu powierzchni tymczasowego obszaru przesuwane są na początek listy pozycji.

Strategia *pole-wymiary* jest rozwinięciem strategii minimalnego obszaru. Obliczana jest wartość współczynnika uwzględniającego pole powierzchni tymczasowego obszaru i różnicę długości jego boków. Zastosowanie opisanego współczynnika ma na celu tworzenie obszarów o bardziej równomiernym kształcie. Premiowane są pozycje o jak najmniejszej wartości współczynnika pole-wymiary.

Kryterium według którego sortowane są dostępne pozycje, przy wykorzystaniu strategii *ruletki*, wybierane jest z pewnym prawdopodobieństwem. Możliwe jest sortowanie pozycji według mniejszych wartości współrzędnych *x*, *y* oraz różnicy *x*-*y*. Wraz z kolejnymi iteracjami wzrasta prawdopodobieństwo sortowania według kryterium $x - y$. Dzięki temu,

wraz z powiększaniem się obszaru G, częściej są wykorzystane pozycje dostępne w jego wnętrzu.

Zastosowanie strategii *XYP* powoduje sortowanie dostępnych pozycji według mniejszych wartości współrzędnych x, y oraz dodatkowo różnicy $x - y$ z identycznym prawdopodobieństwem wyboru każdego z nich.

Po wyznaczeniu pozycji położenia wszystkich prostokątów, obliczane jest pole powierzchni $\Phi(G) = L \cdot W$ obszary G, gdzie W jest szerokością, a L wysokością obszaru. *Dokładnością rozwiązania* jest iloraz

$$\delta = \frac{R}{\Phi(G)},$$

gdzie R jest sumą pól powierzchni wszystkich prostokątów.

4.4. Schemat działania algorytmu wraz z przyjętą strategią

Ogólny schemat algorytmu symulowanego wyżarzania wraz z dowolną strategią pakowania przedstawiono na rysunku 6.

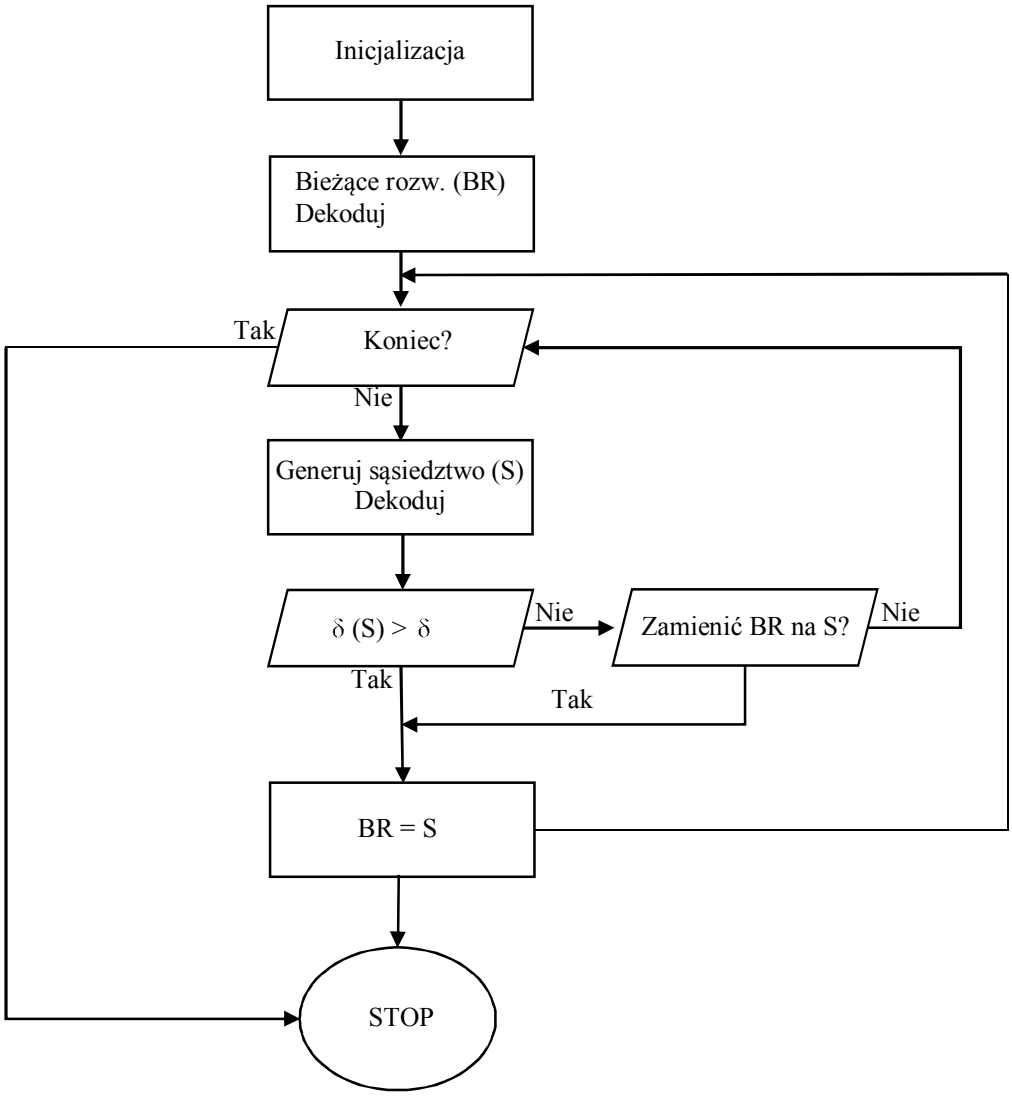
W kroku pierwszym (Inicjalizacja) są wczytywane dane oraz inicjowane podstawowe zmienne. Wyznaczane jest pierwsze rozwiązanie - Bieżące rozw. (BR), a następnie jest ono dekodowane (tj. na jego podstawie są wyznaczane pozycje każdego z prostokątów wewnątrz obszaru G). Po dekodowaniu obliczana jest dokładność rozwiązania. Następne kroki algorytmu są powtarzane aż do momentu, gdy jest spełniony warunek stopu. Kolejno, generowane jest sąsiedztwo rozwiązania bieżącego, a następnie każde rozwiązanie z sąsiedztwa poddawane jest dekodowaniu. Jeżeli w sąsiedztwie znajduje się lepsze rozwiązanie (δ (S)> δ (BR)), zastępuje ono aktualne (tj. BR=S). W przypadku, gdy żadne rozwiązanie z sąsiedztwa nie ma mniejszej dokładności, wówczas zgodnie z funkcją prawdopodobieństwa jest akceptowane jedno z sąsiednich rozwiązań.

5. Eksperymenty obliczeniowe

Obliczenia wykonano na komputerze wyposażonym w 6-rdzeniowy procesor Intel Core i7 CPU X980 (3.33GHz) i system operacyjny Linux Ubuntu 12.04.5 LTS.

Na potrzeby eksperymentów obliczeniowych wygenerowane zostały dane testowe o różnym stopniu heterogeniczności. Liczba możliwych typów prostokątów wynosiła od trzech do dwudziestu. Wraz ze wzrostem liczby typów danych wejściowych zmniejszana była liczba, dostępnych w każdym typie, prostokątów. Dla każdej instancji danych testowych uruchamiano czterokrotnie algorytm z różnymi strategiami pakowania.

Przyjęto ponadto następujące założenia. W każdej iteracji algorytmu SW generowana była określona liczba rozwiązań sąsiadujących z bieżącym rozwiązaniem. Rozwiązania gorsze od aktualnego, były akceptowane z prawdopodobieństwem $e^{-\Delta/T}$, gdzie Δ jest różnicą pola obszaru rozwiązania bieżącego i rozwiązania z sąsiedztwa. Sąsiedztwo generowane było z użyciem obu operatorów, z częstszym stosowaniem operatora odwracającego kolejność elementów w permutacji (chromosomie). Warunkiem stopu była maksymalna liczba iteracji algorytmu symulowanego wyżarzania, równa 1000. Eksperymenty obliczeniowe przeprowadzone zostały wielokrotnie, dla każdej instancji problemu.



Rys. 6. Schemat działania algorytmu SA wraz ze strategią pakowania

Wyniki badań przedstawione zostały w tabelach (2-5). Każda tabela zawiera uśrednione wyniki obliczeń przeprowadzonych z zastosowaniem jednej ze strategii pakowania. Każdy wiersz tabeli zawiera wyniki uzyskanym dla jednej instalacji danych testowych: liczbę typów prostokątów (*Typy*), dokładność wyznaczonego rozwiązania (*Dokładność*), pole powierzchni obszaru zawierającego rozważane prostokąty (*Pole obszaru*), długości krótszego i dłuższego boku pola obszaru (*Krótszy, Dłuższy bok*), stosunek dłuższego boku do krótszego (*Ratio*) oraz czas obliczeń w minutach (*Czas*).

Tab. 2. Strategia minimalnego obszaru

Typy	*Dokładność*	*Pole obszaru*	*Krótszy bok*	*Dłuższy bok*	*Ratio*	*Czas*
3	0,96	470363	195	2487	12,74	47,9
5	0,92	644772	219	3081	14,10	32,8
8	0,90	726975	341	2757	8,08	101,3
10	0,90	610817	227	2704	11,90	141,3
12	0,88	657578	166	4135	24,94	158,2
15	0,90	676805	261	2744	10,53	86,2
20	0,86	869917	366	2828	7,73	137,9

Najwyższą dokładność rozwiązań, dla wszystkich instancji problemu, otrzymano przy zastosowaniu strategii minimalnego obszaru (tab. 2). Jedynie dla danych zawierających dziesięć typów prostokątów nie odnotowano najkrótszego czasu obliczeń. Wartości *Ratio* były zdecydowanie większe od wartości otrzymanych przy zastosowaniu pozostałych strategii pakowania (dłuższy bok był od 7,73 do 24,94 krotnie większy od krótszego).

Tab. 3. Strategia współczynnika pole-wymiary

Typy	*Dokładność*	*Pole obszaru*	*Krótszy bok*	*Dłuższy bok*	*Ratio*	*Czas*
3	0,92	486734	687	709	1,03	112,7
5	0,89	661716	808	819	1,01	94,2
8	0,87	749650	858	874	1,02	218,7
10	0,86	641743	797	805	1,01	172,1
12	0,85	680492	815	835	1,03	206,0
15	0,86	707148	836	846	1,01	153,1
20	0,85	883282	938	941	1,00	301,8

Stosując strategię współczynnika pola-wymiaru (tab. 3) otrzymano rozwiązania o nieznacznie gorszej dokładności od rozwiązań przedstawionych w tabeli 2. Średni czas obliczeń jest mniejszy tylko w jednym przypadku (3 typów). Zastosowanie tej strategia dało znacznie mniejszy współczynnik *Ratio* w sześciu przypadkach (tj. dla 3, 5, 10, 12, 15 i 20 typów) oraz dwukrotnie o takiej samej wartości jak w przypadku strategii ruletki. Wartości *Ratio* bliskie 1.00 oznacza, że kształt wyznaczonego obszaru jest zbliżony do kwadratu.

Dokładność rozwiązań wyznaczonych przy zastosowaniu strategii ruletki (tab. 4) była mniejsza dla trzech przypadkach danych wejściowych, tj. dla instancji z liczbą typów prostokątów równą 8, 10 lub 20. Z kolei czas obliczeń był najkrótszy tylko w jednym przypadku (10 typów). Wyznaczony współczynnik *Ratio* miał najmniejszą wartość dla trzech przypadków z liczbą typów: 8, 10 i 12.

Tab. 4. Strategia ruletki

Typy	*Dokładność*	*Pole obszaru*	*Krótszy bok*	*Dłuższy bok*	*Ratio*	*Czas*
3	0,23	1961091	1364	1439	1,06	85,6
5	0,24	2472506	1548	1597	1,03	72,9
8	0,10	6312636	2495	2530	1,01	120,3
10	0,16	3394074	1828	1855	1,01	114,1
12	0,19	3032269	1718	1764	1,03	179,0
15	0,22	2826178	1627	1740	1,07	136,3
20	0,15	5101250	2238	2279	1,02	245,2

Zastosowanie strategii XYP (tab. 5) dało w pięciu przypadkach (3, 5, 12, 15 i 20 typów) rozwiązania o znacznie gorszej dokładności. Zdecydowanie najdłuższe były także czasy obliczeń oraz największe wartości współczynnika *Ratio*.

Tab. 5. Strategia XYP

Typy	*Dokładność*	*Pole obszaru*	*Krótszy bok*	*Dłuższy bok*	*Ratio*	*Czas*
3	0,14	3128115	1143	2759	2,41	112,1
5	0,21	2850932	1405	2042	1,45	95,2
8	0,16	4047605	1560	2597	1,66	263,1
10	0,19	2932445	1360	2166	1,59	219,8
12	0,18	3227463	1457	2226	1,53	260,3
15	0,20	3041758	1435	2134	1,49	196,3
20	0,15	5006677	1903	2632	1,38	378,3

6. Wnioski

W pracy rozpatrywano problem harmonogramowania zadań wielomaszynowych na równoległych (identycznych) maszynach z minimalizacją iloczynu liczby użytych maszyn i długości czasu wykonywania wszystkich zadań. Wyznaczenie rozwiązania można sprowadzić do pewnego problemu dwuwymiarowego pakowania prostokątów z minimalizacją pola powierzchni zajętego obszaru. Przedstawiono algorytm symulowanego wyżarzania z zastosowaniem różnych strategii pakowania, tj. umieszczania prostokątów wewnątrz obszaru. Przeprowadzone eksperymenty obliczeniowe wykazały, że wyznaczone pola powierzchni (rozwiązania problemu harmonogramowania zadań) są tylko nieznacznie większe od sumy pól wszystkich prostokątów (tj. od dolnego ograniczenia).

Literatura

1. Błażewicz J., Drabowski M., Węglarz J., Scheduling multiprocessor tasks to minimize schedule length, IEEE Transaction, 1986, 389-393.

2. Błądek I., Drozdowski M., Guinand F., Schepler X., On contiguous and non-contiguous parallel task scheduling, Journal of Scheduling, 2015.
3. Bozoki, G., Richard, J.-P., A branch-and-bound algorithm for the continous-process job-shop scheduling problem, AIIE Transactions 2/3, 1970, 246-252.
4. Drozdowski M., Scheduling multiprocessor tasks an overview, European Journal of Operational Research 94, 1996, 215-230.
5. Drozdowski M., Selected problems of scheduling tasks in multiprocessor computer systems, Poznan University of Technology Press, Series: Monographs No. 321, 1997.
6. Janiak A., Lichtenstein M., Mokrzysz K., Szeregowanie zadań wieloprocesorowych z kryterium minimalizacji powierzchni uszeregowania, Automatyzacja Procesów Dyskretnych, 2012.
7. Kramer A., Scheduling multiprocessor tasks on dedicated processors. PhD Thesis, Fachbereich Mathematik/Informatik, Univeristat Osnabruck, 1995.
8. Lin J., Chen S., Scheduling algorithm for nonpreemptive multiprocessor tasks, Computers and Mathematics with Applications 28/4, 1994, 85-92.
9. Lloyd E., Concurrent task system, Operations Research 29/1, 1981, 189-201.
10. Nowicki E., Metoda tabu w problemach szeregowania zadań produkcyjnych, Oficyna wydawnicza Politechniki Wrocławskiej, Wrocław, 1999.
11. Makuchowski M., Problemy gniazdowe z operacjami wielomaszynowymi. Własności i algorytmy, praca doktorska, Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, Raport serii: PREPRINTY nr 37/2004.
12. Rudek R., Agnieszka Rudek A., Kozik A., The solution algorithms for the multiprocessor scheduling with workspan criterion, Expert Systems with Applications 40, 2013, 2799–2806.
13. Vizing, V., About schedules observing deadlines, Kibernetika 1, 1981, 128-135.

Mgr inż. Karolina BOROWICKA
Prof. nadzw. dr hab. Wojciech BOŻEJKO
Mgr inż. Łukasz KACPRZAK
Katedra Automatyki, Mechatroniki i Systemów Sterowania
Politechnika Wrocławska
ul. Janiszewskiego 11/17, 50-372 Wrocław
e-mail: wojciech.bozejko@edu.pwr.wroc.pl
lukasz.kacprzak@edu.pwr.wroc.pl

Prof. nadzw. dr hab. Mieczysław WODECKI
Instytut Informatyki Uniwersytetu Wrocławskiego
ul. Joliot-Curie 15, 50-383 Wrocław
e-mail: mwd@ii.uni.wroc.pl