Andrzej GNATOWSKI
Politechnika Wrocławska

RÓWNOLEGŁY ALGORYTM DOKŁADNY DLA PROBLEMU PRZYDZIAŁU W WIELOMASZYNOWYM GNIEŹDZIE PRODUKCYJNYM

Streszczenie. Problem przydziału operacji do maszyn w gnieździe wielomaszynowym stanowi istotne w praktyce zagadnienie, w tym także jako element innych problemów optymalizacji. Ze względu na duży nakład obliczeń niezbędny w przypadku zastosowania istniejącego algorytmu dokładnego, w pracy zaproponowano jego równoległą odmianę, pozwalającą na wykorzystanie potencjału środowisk wieloprocesorowych.

PARALLEL EXACT ALGORITHM FOR THE ASSIGNMENT PROBLEM IN THE MULTI-MACHINE NEST

Summary. The problem of operations to machines assignment in a multi-machine nest is a relevant issue in practice, also as part of a wider optimization problem. Due to the large amount of calculations necessary for the use of the existing exact algorithm, in the paper a parallel variant of the algorithm is proposed, aiming for the multiprocessor environments.

1. Wprowadzenie

Gniazda produkcyjne z równoległymi maszynami stanowią istotny element wielu współczesnych systemów produkcyjnych. W systemach takich, poszczególne operacje mogą być wykonywane na różnych maszynach, umożliwiając tworzenie alternatywnych marszrut technologicznych. Użycie gniazd wielomaszynowych wiąże się jednak z dodatkową trudnością w planowaniu, ponieważ oprócz harmonogramu wykonywania kolejnych zadań, konieczne jest opracowanie przydziału operacji do maszyn. W celu rozwiązania tak postawionego problemu, stosować można algorytmy dwupoziomowe [1], w których algorytm pierwszego poziomu odpowiada za dobór kolejności wykonywanych zadań, a algorytm niższego poziomu — za przydział operacji do maszyn.

W niniejszej pracy rozpatrywany jest problem przydziału operacji do maszyn w gnieździe wielomaszynowym, w którym kolejność wykonywania operacji jest ustalona. Przed każdą operacją wykonywana jest konfiguracja (przezbrojenie), której czas trwania zależy od poprzedniej operacji wykonywanej na danej maszynie. Problem ten nazywany jest dalej problemem przydziału (AP, od ang. *Assignment Problem*). AP może być częścią problemu definiowanego dla bardziej złożonego systemu produkcyjnego, takiego jak elastyczny problem gniazdowy opisany w [6, 7], lub zastępować maszyny w klasycznym zagadnieniu planowania zadań, np. cyklicznym problemie gniazdowym badanym w [4]. Praktyczną implementacją AP może być na przykład ramię robota obsługujące

jedną z wielu maszyn. Optymalizacja procesów produkcyjnych w których wykorzystywane są roboty jest osobnym i interesującym problemem, w którym przypadki dwu- i trzy-maszynowe są często badane [8, 9, 10]. Problem cyklicznego AP w dwumaszynowym gnieździe produkcyjnym rozważano z kolei w [3], również jako element problemu przepływowego. W [2] zaproponowano dwupoziomowy algorytm, w którym algorytm dokładny wykorzystywany był do wyznaczania optymalnego przydziału, a przeszukiwanie z zabronieniami — do poszukiwania optymalnej kolejności wykonywania operacji.

Sekwencyjny algorytm dokładny dla AP zaproponowano w [5]. Niestety, pomimo wielomianowej złożoności obliczeniowej, jego czas wykonania dla instancji AP o dużej liczbie maszyn może być nieakceptowalny w praktyce. W niniejszej pracy przedstawiono równoległy algorytm dokładny, pozwalający wykorzystać środowiska wieloprocesorowe, a tym samym potencjalnie umożliwiający rozwiązywanie większych instancji.

2. Sformułowanie problemu

Problem przydziału (AP) zdefiniowany jest w następujący sposób. Dane jest gniazdo produkcyjne, składające się z m maszyn, numerowanych kolejno $1, 2, \ldots, m$, przy czym zbiór maszyn oznaczono przez $\mathcal{M}$. Na maszynach wykonywane są operacje ze zbioru $\mathcal{O} = \{1, 2, \ldots, o\}$. Każda operacja musi być wykonywana na dokładnie jednej maszynie, określanej przez przydział $P = (P(1), P(2), \ldots, P(o)) \in \mathcal{P}$, gdzie $P(i),\ i \in \mathcal{O}$, oznacza przydział operacji i do maszyny $P(i)$, a $\mathcal{P}$ to zbiór wszystkich możliwych przydziałów. Proces produkcyjny musi spełniać następujące ograniczenia:

1. Operacje muszą być wykonywane w kolejności $1 \rightarrow 2 \rightarrow \cdots \rightarrow o$, tj. jeżeli $i < j,\ i, j \in \mathcal{O}$, to termin rozpoczęcia operacji j nie może być wcześniejszy niż termin zakończenia operacji i.
2. Każda operacja $i \in \mathcal{O}$ musi być wykonywana nieprzerwanie przez p_i^a czasu na maszynie $a = P(i)$, określonej przez przydział P.
3. Na każdej maszynie $a \in \mathcal{M}$, pomiędzy każdymi kolejnymi operacjami $i,\ j \in \mathcal{O}$, musi zostać wykonane nieprzerywalne przezbrojenie, trwające $s_{i,j}^a$ czasu.
4. Jednocześnie może być wykonywana najwyżej jedna operacja albo przezbrojenie.

Niech $C_{max}(P)$ oznacza najkrótszy czas konieczny do wykonania wszystkich operacji dla przydziału P. Problem przydziału sprowadza się do wyznaczenia przydziału $P^* \in \mathcal{P}$ minimalizującego $C_{max}(P)$.

3. Algorytm sekwencyjny

Algorytm sekwencyjny rozwiązywania AP, na którym bazuje przedstawiony dalej algorytm równoległy, pochodzi z [5]. Poniżej zostanie omówiona w skrócie jego zasada działania. Konstrukcja algorytmu opiera się o schemat programowania dynamicznego. Dla instancji AP, definiowany jest zbiór podproblemów $\mathcal{X}$. Dowolny podproblem $\vec{x} \in \mathcal{X}$ może być jednoznacznie opisany wektorem parametrów $\vec{x} = (x_1, x_2, \ldots, x_m) \in \mathcal{X}$, spełniającym warunki

$$\forall i \in \mathcal{M}\ (x_i \in \mathcal{O} \cup \{0\}), \tag{1}$$

$$\forall i, j \in \mathcal{M}\ \Big(x_i = x_j \Rightarrow (x_i = 0 \vee i = j)\Big). \tag{2}$$

Niech $\mathcal{P}^i$ oznacza zbiór wszystkich możliwych przydziałów pierwszych i operacji. Przydział $P \in \mathcal{P}^i$ jest rozwiązaniem dopuszczalnym dla podproblemu $\vec{x}$ wtedy i tylko wtedy, gdy dla każdego $a \in \mathcal{M}$ spełnione są następujące warunki:

1. Jeżeli $x_a = 0$, to na maszynie a nie może być wykonywana żadna operacja.
2. Jeżeli $x_a > 0$, to na maszynie a jako ostatnia jest wykonywana operacja x_a.

Rozwiązaniem optymalnym podproblemu $\vec{x}$ jest taki dopuszczalny przydział $P^* \in \mathcal{P}^i$, $i = \max_{a\in\mathcal{M}} x_a$, że wartość $C_{max}(P^*)$ jest najmniejsza. Najkrótszy czas wykonania wszystkich operacji dla $\vec{x}$ oznaczono przez $C_{max}(\vec{x}) = C_{max}(P^*)$.

Maszyna na której w podproblemie $\vec{x}$ wykonywana jest ostatnia operacja została oznaczona przez $\omega(\vec{x})$, natomiast przedostatnia aktywna maszynę przez

$$\psi(\vec{x}) = \begin{cases} 0 & \text{dla: } A(\vec{x}) < 2, \\ \max\Big\{a \in \mathcal{M} \setminus \{\omega(\vec{x})\} \ : \ \Big(x_a = \max\limits_{b\in\mathcal{M}\setminus\{\omega(\vec{x})\}} x_b\Big)\Big\} & \text{dla: } A(\vec{x}) \geqslant 2, \end{cases} \tag{3}$$

gdzie $A(\vec{x}) = |\{i \in \mathcal{M} : x_i \neq 0\}|$ jest liczbą maszyn na których wykonywane są operacje. Dla zwiększenia czytelności dalszych rozważań, w przypadku gdy $\omega(\vec{x}) = 0$ lub $\phi(\vec{x}) = 0$, przyjęto $x_0 := 0$. Za pomocą wprowadzonej notacji, zbiór $\mathcal{X}$ można podzielić na trzy rozłączne podzbiory

$$\mathcal{X}^{(1)} = \{\vec{x} \in \mathcal{X} \ : \ x_{\omega(\vec{x})} - x_{\psi(\vec{x})} = 0\} = \{(0, 0, \ldots, 0)\}, \tag{4}$$

$$\mathcal{X}^{(2)} = \{\vec{x} \in \mathcal{X} \ : \ x_{\omega(\vec{x})} - x_{\psi(\vec{x})} = 1\}, \tag{5}$$

$$\mathcal{X}^{(3)} = \{\vec{x} \in \mathcal{X} \ : \ x_{\omega(\vec{x})} - x_{\psi(\vec{x})} > 1\}. \tag{6}$$

Dla każdego z nich, w [2] wyprowadzono wzór na $C_{max}(\vec{x})$

$$C_{max}(\vec{x}) = \begin{cases} 0 & \text{gdy: } \vec{x} \in \mathcal{X}^{(1)}, \\ \min\limits_{i\in u(\vec{x})} \Big\{C_{max}\big(\mu(\vec{x}, i)\big) + p^{\omega(\vec{x})}_{x_{\omega(\vec{x})}} + s^{\omega(\vec{x})}_{i,x_{\omega(\vec{x})}}\Big\} & \text{gdy: } \vec{x} \in \mathcal{X}^{(2)}, \\ C_{max}\big(\mu(\vec{x}, x_{\psi(\vec{x})} + 1)\big) + \sum\limits_{i=x_{\psi(\vec{x})}+2}^{x_{\omega(\vec{x})}} \big(p^{\omega(\vec{x})}_i + s^{\omega(\vec{x})}_{i-1,i}\big) & \text{gdy: } \vec{x} \in \mathcal{X}^{(3)}, \end{cases} \tag{7}$$

gdzie

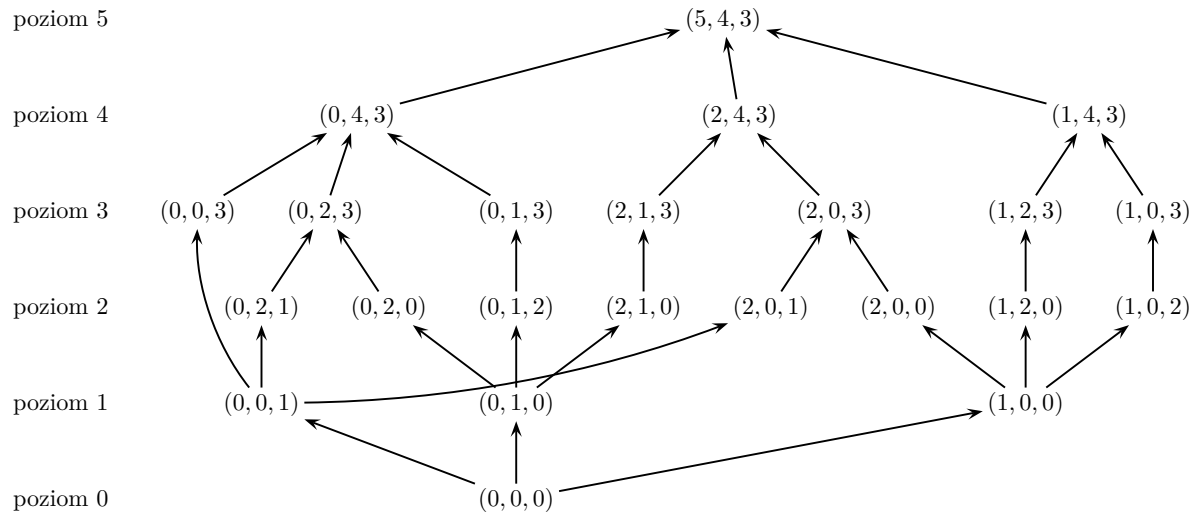
$$\mu(\vec{x}, i) = (x_1, x_2, \ldots, x_{\omega(\vec{x})-2}, x_{\omega(\vec{x})-1}, i, x_{\omega(\vec{x})+1}, x_{\omega(\vec{x})+2}, \ldots, x_m), \tag{8}$$

oraz

$$u(\vec{x}) = \{0\} \cup \big(\{1, 2, \ldots, x_{\omega(\vec{x})}\} \setminus \{x_1, x_2, \ldots, x_m\}\big). \tag{9}$$

Opisane wyżej rozważania pozwalają na wyznaczenie wartości $C_{max}(\vec{x})$ dla dowolnego podproblemu. Aby wyznaczyć optymalne rozwiązanie instancji AP, w [2] zaproponowano graf $\mathcal{B}(\vec{x}) = (V(\vec{x}), E(\vec{x}))$, skonstruowany w oparciu o równanie (7). Zbiór wierzchołków zdefiniowano jako $V(\vec{x}) = f_1(\vec{x})$,

$$f_1(\vec{x}) = \begin{cases} \{\emptyset\} & \text{gdy: } \vec{x} \in \mathcal{X}^{(1)}, \\ \bigcup\limits_{i\in u(\vec{x})} \Big(\{\mu(\vec{x}, i)\} \cup f_1\big(\mu(\vec{x}, i)\big)\Big) & \text{gdy: } \vec{x} \in \mathcal{X}^{(2)}, \\ \big\{\mu(\vec{x}, x_{\psi(\vec{x})} + 1)\big\} \cup f_1\big(\mu(\vec{x}, x_{\psi(\vec{x})} + 1)\big) & \text{gdy: } \vec{x} \in \mathcal{X}^{(3)}, \end{cases} \tag{10}$$



Rys. 1.. Graf $\mathcal{B}(\vec{x})$ dla $\vec{x} = (5, 4, 3)$

natomiast zbiór łuków jako $E(\vec{x}) = \left\{(\vec{i}, \vec{j}) : \vec{i}, \vec{j} \in V \wedge \vec{i} \in f_2(\vec{j})\right\}$,

$$f_2(\vec{x}) = \begin{cases} \{\emptyset\} & \text{gdy: } \vec{x} \in \mathcal{X}^{(1)}, \\ \bigcup\limits_{i \in u(\vec{x})} \{\mu(\vec{x}, i)\} & \text{gdy: } \vec{x} \in \mathcal{X}^{(2)}, \\ \left\{\mu(\vec{x}, x_{\psi(\vec{x})} + 1)\right\} & \text{gdy: } \vec{x} \in \mathcal{X}^{(3)}. \end{cases} \tag{11}$$

Przykład grafu $\mathcal{B}(\vec{x})$ dla $\vec{x} = (5, 4, 3)$ pokazano na rys. 1.

Aby móc wykorzystać graf $\mathcal{B}$ do rozwiązania instancji AP, należy wprowadzić m operacji o zerowych czasach przezbrojeń oraz wykonania, wykonywanych po operacjach ze zbioru $\mathcal{O}$. Wtedy, rozwiązując podproblem $(o + m + 1, o + m, \ldots, o + 1)$, możliwe jest uzyskanie rozwiązanie instancji AP. Dla uproszczenia notacji, w dalszej części pracy w zapisie $\mathcal{B}(\vec{x})$, $V(\vec{x})$, $E(\vec{x})$, pomijane będzie „$(\vec{x})$", jeżeli $\vec{x} = (o + m + 1, o + m, \ldots, o + 1)$.

W grafie $\mathcal{B}$ wyróżniono ścieżkę $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_{|\kappa|})$, gdzie

$$\kappa_i = \begin{cases} (n + m, n + m - 1, \ldots, n + 1) & \text{gdy: } i = 1, \\ \max\left\{\vec{x} \in f_2(\kappa_{i-1}) : \left(\vec{x} = \min\limits_{\vec{y} \in f_2(\kappa_{i-1})} \left\{C_{max}(\vec{y})\right\}\right)\right\} & \text{gdy: } i \in \{2, 3, \ldots, |\kappa|\}, \end{cases} \tag{12}$$

z pewnym porządkiem definiowanym dla zbioru podproblemów (dla zastosowania operacji „max") opartym na przykład o równ. 13. Korzystając z algorytmu 1, ścieżka κ może posłużyć do wyznaczenia rozwiązania optymalnego instancji AP.

Algorytm 1: Budowa optymalnego przydziału na podstawie ścieżki κ

```
1 while i = 1, 2, ..., |κ| do
2 |   x⃗ ← κ_i;
3 |   for i = x_ψ(x⃗) + 1, x_ψ(x⃗) + 2, ..., x_ω(x⃗) do
4 |   |   if i ⩽ o then P(i) ← ω(x⃗);
5 return P
```

4. Algorytm równoległy

Analizując strukturę grafu $\mathcal{B}$ łatwo zauważyć, że podczas rozwiązywania instancji AP za pomocą równania (7), możliwe jest równoległe wyznaczanie $C_{max}(\vec{x})$ dla podproblemów cechujących się tą samą wartością funkcji ω (ta sama maszyna jest w nich aktywna jako ostatnia). Opierając się na tym spostrzeżeniu, zdefiniowano pojęcie *poziomu*. Podproblem $\vec{x} \in \mathcal{X}$ należy do poziomu l wtedy i tylko wtedy, gdy $\vec{x} \in \{\vec{y} \in \mathcal{X} : \omega(\vec{y}) = l\} = \mathcal{X}_l$. Przynależność wierzchołków grafu $\mathcal{B}((5,4,3))$ do poziomów 0–5 pokazano na rys. 1.

W zaproponowano w tej pracy algorytmie, w pierwszej kolejności budowany jest graf $\mathcal{B}$, oraz ustalane jest do których poziomów należą jego wierzchołki. Następnie, równolegle rozwiązywane są podproblemy składające się na kolejne poziomy tego grafu.

4.1. Indeksowanie podproblemów

Komórki pamięci komputera (oraz pamięci maszyny PRAM, wykorzystywanej dalej do analizy złożoności obliczeniowej algorytmu równoległego) są adresowane przez kolejne liczby całkowite. Tym samym warta rozważenia jest szybka metoda jednoznacznego przekształcenia wektora $\vec{x} \in \mathcal{X}$ dowolnego podproblemu, w liczbę całkowitą z możliwie wąskiego przedziału — i odwrotnie. W tym celu zaproponowano funkcję

$$\gamma(\vec{x}) = \sum_{i=1}^{m} x_i (o+m+1)^{i-1}, \tag{13}$$

oraz odwrotną do niej

$$\gamma^{-1}(i) = \vec{x} \;\Rightarrow\; \forall i \in \mathcal{M} \;\left(x_j = \left\lfloor i/(o+m+1)^{i-1} \right\rfloor \mod (o+m+1)\right). \tag{14}$$

4.2. Równoległa budowa grafu $\mathcal{B}$

Pseudokod równoległej metody budowy grafu $\mathcal{B}$ pokazano w algorytmie 2. W pierwszej kolejności, zapisywane są w pamięci wektory $\vec{x}$ odpowiadające indeksom $i = 0, 1, \ldots, (o+m+1)^m$. Jeżeli tak powstały wektor $\vec{x}$ spełnia warunki z równań (1)–(2), to jest oznaczany jako podproblem oraz obliczane i zapisywane są dla niego wartości $\omega(\vec{x})$ oraz $\psi(\vec{x})$. Następnie, zbiór podproblemów $\mathcal{X}$ jest dzielony na podzbiory podproblemów o takich samych poziomach, które są zapisywane w tablicach $\mathtt{X}_l$. W tym celu, do każdego z poziomów $l \in \{0, 1, \ldots, m+o+1\}$, przydzielane jest $(o+m+1)^m$ procesorów. Procesory te kopiują do tablicy $\mathtt{X}_l$ swój indeks tylko wtedy, gdy odpowiadający mu podproblem należy do danego poziomu. Następnie, z tablic usuwane są pola do których nie przypisano wartości. W kolejnym kroku algorytmu, kolejno dla każdego z poziomów $l = o+m+1, o+m, \ldots, 1$, rozważane są równolegle podproblemy odpowiadające indeksom z tablicy $\mathtt{X}_l$. Jeżeli podproblem $\vec{x} = \gamma^{-1}(i)$ należy do V, to w tablicy $\mathtt{D}[i]$ zapisywane są indeksy podproblemów których rozwiązania są niezbędne do wyznaczenia wartości $C_{max}(\vec{x})$ z równania (7) (bez odwoływania się do rekurencji). Dodatkowo, podproblemy te są oznaczane jako należące do V. Ostatecznie, z tablic $\mathtt{X}_l$ usuwane są indeksy podproblemów które nie są wierzchołkami grafu $\mathcal{B}$, tak by

$$\forall l \in \{0, 1, \ldots, o+m+1\}\; \forall i \in \{1, 2, \ldots, |\mathtt{X}_l|\}\; (\mathtt{X}_l[i] \in \mathcal{X}_l \cap V).$$

Własność 1. *Niech dana będzie instancja AP o m-maszynach i o-operacjach. Budowa grafu $\mathcal{B}$ dla tej instancji na $((o+m+1)^m+1)\max\{m^2,(o+m)\}$ procesorowej maszynie*

Algorytm 2: Równoległa metoda budowy grafu $\mathcal{B}$

wejście: Instancja AP
wyjście: Graf $\mathcal{B}$; $\gamma^{-1}, \omega, \psi$ dla wierzchołków grafu oraz ich indeksów

1 **parfor** $i = 0, 1, \ldots, (o+m+1)^m$ **do** // $p = (o+m+1)^m + 1$
2 wyznacz wektor $\vec{x} = \gamma^{-1}(i)$, odpowiadający indeksowi i;
3 **if** $\vec{x} \in \mathcal{X}$ **then**
4 oblicz i zapisz wartość $\omega(\vec{x})$ oraz $\psi(\vec{x})$;
5 **parfor** $l = 1, 2, \ldots, o+m$ **do** // $p = o+m$
6 **parfor** $i = 0, 1, \ldots, (o+m+1)^m$ **do** // $p = (o+m+1)^m + 1$
7 **if** $\omega(\gamma(i)) = l$ **then**
8 $\mathtt{X}_l[i] = i$;
9 usuń z tablicy $\mathtt{X}_l$ niewykorzystane pola;
10 zapisz, że $(o+m+1, o+m, \ldots, o+1)$ należy do V i $\mathtt{X}_{o+m+1}$;
11 **for** $l = o+m+1, o+m, \ldots, 1$ **do**
12 **parfor** $i = \mathtt{X}_l[1], \mathtt{X}_l[2], \ldots, \mathtt{X}_l[|\mathtt{X}_l|]$ **do** // $p = |\mathtt{X}^l| \leqslant (o+m+1)^m + 1$
13 weź $\vec{x} = \gamma(i)$, odpowiadające indeksowi i;
14 **if** $\vec{x} \in V$ **then**
15 usuń z tablicy $\mathtt{D}[i]$ niewykorzystane pola;
16 **if** $\vec{x} \in \mathcal{X}^{(2)}$ **then**
17 **parfor** $j = 0, 1, \ldots, l$ **do** // $p = o+m+1$
18 $\mathtt{U}[j] \leftarrow 1$;
19 **parfor** $j = 1, 2, \ldots, m$ **do** // $p = m$
20 $\mathtt{U}[x_j] \leftarrow 0$;
21 $\mathtt{U}[0] \leftarrow 1$;
22 **parfor** $j = 0, 1, \ldots, l$ **do** // $p = o+m+1$
23 **if** $\mathtt{U}[j] = 1$ **then**
24 wyznacz indeks k podproblemu $\mu(\vec{x}, j)$;
25 zapisz k jako konieczne do rozwiązania i w $\mathtt{D}[i]$;
26 **if** $\vec{x} \in \mathcal{X}^{(3)}$ **then**
27 wyznacz indeks k podproblemu $\mu(\vec{x}, \psi(\vec{x}) + 1)$;
28 zapisz k jako konieczne do rozwiązania i w $\mathtt{D}[i]$;
29 **parfor** $l = 1, 2, \ldots, o+m$ **do** // $p = o+m$
30 Usuń z $\mathtt{X}_l$ podproblemy $\vec{x} \notin V$;

CREW PRAM może zostać wykonana przy użyciu algorytmu 2 w czasie $O(m \log(o+m) + (o+m)\log m)$.

Dowód. Dowód polega na analizie złożoności obliczeniowej algorytmu 2. Wartości funkcji γ^{-1} z linii 2 można wyznaczyć w czasie $O(1)$ przy użyciu m procesorów. Sprawdzenie spełnienia warunków z równań (1)–(2) w linii 3 to z kolei $O(\log m)$ czasu przy wykorzystaniu m^2 procesorów. Wyznaczenie ostatniej i przedostatniej operacji podproblemu może zostać wykonane w $O(\log m)$ czasu na m procesorach.

Stąd, złożoność czasowa pętli z linii 1–4 to $O(\log m)$, na $m^2((o+m+1)^m+1)$ procesorach. Złożoność czasowa usunięcia wybranych elementów tablicy $\mathtt{X}_l$, o rozmiarze $|\mathtt{X}_l| = (o+m+1)^m+1$, za pomocą metody opartej o obliczanie sum prefiksowych, wynosi $O(\log((o+m+1)^m+1)) = O(m\log(o+m))$, przy użyciu $(o+m+1)^m+1$ procesorów. Analogicznie, operacja z linii 15 może zostać wykonana w czasie $O(\log m)$ na m procesorach. Zakładając dostępność na maszynie PRAM potęgowania jako pojedynczej instrukcji, operacje z linii 24 oraz 27 mogą być z kolei wykonane w czasie stałym na jednym procesorze, korzystając ze wzoru

$$k = i + (j - x_{\omega(\vec{x})}) \cdot (o+m+1)^{\omega(\vec{x})-1}.$$

Ostatecznie, algorytm może zostać wykonany w czasie

$$\begin{aligned} O(\log m + m\log(o+m) + (o+m+1)\log m + m\log(o+m)) = \\ O(m\log(o+m) + (o+m)\log m), \end{aligned} \tag{15}$$

na CREW PRAM wyposażonej w

$$\begin{aligned} \max\left\{m^2((o+m+1)^m+1), (o+m)((o+m+1)^m+1)\right\} = \\ ((o+m+1)^m+1)\max\left\{m^2, (o+m)\right\} \end{aligned} \tag{16}$$

procesorów. □

4.3. Rozwiązywanie AP

Dysponując grafem $\mathcal{B}$ (zapisanych jako tablice $\mathtt{X}_l, \mathtt{D}$), wyznaczanie rozwiązania AP sprowadza się do sukcesywnego, równoległego, rozwiązywania podproblemów należących do kolejnych poziomów grafu. Proces ten został pokazany w algorytmie 3. Wartości C_{max} dla podproblemów są zapisywane w tablicy $\mathtt{C}$. W pierwszej kolejności, inicjowana jest wartość $\mathtt{C}[0] = C_{max}((0,0,\ldots,0)) = 0$ oraz wyznaczane są sumy prefiksowe służące do przyspieszenia wykonywania sum z linii 16. Następnie, kolejno dla każdego z poziomów $l = 1, 2, \ldots, o+m$, wyznaczane są równolegle za pomocą równania (7) wartości $C_{max}(\vec{x})$ dla podproblemów $\mathcal{X}_l \cap V$ należących do bieżącego poziomu l i grafu $\mathcal{B}$. Jeżeli rozważany podproblem $\vec{x}$ należy do zbioru $\mathcal{X}^{(2)}$, to wartość $C_{max}(\vec{x})$ jest uzależniona od rozwiązań podproblemów zapisanych w $\mathtt{D}[\gamma(\vec{x})]$. Ostatnim krokiem algorytmu jest wyznaczenie optymalnego przydziału P na podstawie zawartości tablicy $\mathtt{A}$ (podobnie jak w alg. 1).

Własność 2. *Niech dana będzie instancja AP o m-maszynach i o-operacjach oraz zbudowany dla niej za pomocą algorytmu 3 graf $\mathcal{B}$. Wyznaczenie rozwiązania optymalnego dla tej instancji na $(o+m)^2$ procesorowej maszynie CREW PRAM może odbyć się w czasie $O((o+m)\log m)$.*

Dowód. Dowód polega na analizie złożoności obliczeniowej algorytmu 3. Wyznaczenie sum prefiksowych $o+m$-elementowych może być wykonane w czasie $O(\log(m+o))$ na $o+m$ procesorach. Tym samym cały proces obliczania sum prefiksowych zajmuje $O(\log(m+o))$ na $(o+m)^2$ procesorach. Operacja z linii 6 może zostać wykonana w stałym czasie, opierając się na obliczeniach wykonanych uprzednio przez algorytm 3. Podobnie, sprawdzenie do którego z podzbiorów należy $\vec{x}$ może odbyć się w stałym czasie, dzięki znajomości $\omega(\vec{x})$ oraz $\psi(\vec{x})$. Pobranie numeru

Algorytm 3: Równoległa metoda rozwiązywania AP

wejście: Instancja AP; graf $\mathcal{B}$ zbudowany za pomocą alg. 2
wyjście: Optymalny przydział P

1 **parfor** $l = 1, 2, \ldots, o + m$ **do** // $p = o + m$
2 wyznacz sumy prefiksowe p_i^l oraz $s_{i,i+1}^l$ dla $i \in \{1, 2, \ldots, o + m\}$;
3 $\mathtt{C}[0] \leftarrow 0$;
4 **for** $l = 1, 2, \ldots, o + m$ **do**
5 **parfor** $i \in \mathtt{X}^l$ **do** // $p = |\mathtt{X}^l| \leqslant (o + m + 1)^m + 1$
6 weź $\vec{x}$ odpowiadające indeksowi i;
7 **if** $\vec{x} \in \mathcal{X}^{(2)}$ **then**
8 **parfor** $j \in \mathtt{D}[i]$ **do** // $p = |\mathtt{D}[i]| \leqslant m$
9 weź maszynę różniącą i i j i zapisz w $\mathtt{M}[j][1]$;
10 $\mathtt{M}[j][2] = \mathtt{C}[j] + p_{x_{\omega(\vec{x})}}^{\omega(\vec{x})} + s_{\mathtt{M}[j][1],x_{\omega(\vec{x})}}^{\omega(\vec{x})}$;
11 $k \leftarrow \arg \min_{j \in \{0,1,\ldots,|\mathtt{D}[i]|\}} \{\mathtt{M}[j][2]\}$;
12 $\mathtt{A}[i] \leftarrow \mathtt{M}[k][1]$;
13 $\mathtt{C}[i] \leftarrow \mathtt{M}[\mathtt{A}[i]][2]$;
14 **if** $\vec{x} \in \mathcal{X}^{(3)}$ **then**
15 $\mathtt{A}[i] \leftarrow \mathtt{D}[i]$;
16 $\mathtt{C}[i] \leftarrow \mathtt{C}[\mathtt{A}[i]] + \sum_{k=x_{\psi(\vec{x})}+2}^{x_{\omega(\vec{x})}} \left(p_i^{\omega(\vec{x})} + s_{k-1,k}^{\omega(\vec{x})}\right)$;
17 wyznacz indeks i podproblemu $(o + m + 1, o + m, \ldots, o + 1)$;
18 **while** $i \neq 0$ **do**
19 weź $\vec{x}$ odpowiadający i;
20 **parfor** $i = x_{\psi(\vec{x})} + 1, x_{\psi(\vec{x})} + 2, \ldots, x_{\omega(\vec{x})}$ **do** // $p \leqslant m + o + 1$
21 **if** $i \leqslant o$ **then**
22 $P(i) \leftarrow \omega(\vec{x})$;
23 $i \leftarrow \mathtt{A}[i]$;
24 **return** P

maszyny z linii 9 może być wykonane w czasie stałym, pod warunkiem dostępności operacji log na maszynie PRAM, korzystając ze wzoru $\mathtt{M}[j][1] = \lfloor \log_{o+m+n} |i - j| \rfloor$. Jeżeli logarytm nie należy do instrukcji procesora PRAM, operację z linii 9 można przenieść do algorytmu 2, lub wyznaczyć równolegle w czasie stałym za pomocą m procesorów. Operacje z linii 11 można wykonać w czasie $O(\log |\mathtt{D}[i]|) = O(\log m)$ na m procesorach. Dzięki wcześniejszemu obliczeniu sum prefiksowych, operacja z linii 16 może zostać wykonana w czasie stałym.

Pętla z linii 18 odzwierciedla odwiedzanie kolejnych wierzchołków ścieżki κ, łączącej wierzchołki $(o + m + 1, o + m, \ldots, o + 1)$ oraz $(0, 0, \ldots, 0)$. Stąd nie może składać się z więcej niż $m + o$ iteracji. Tym samym, wyznaczenie optymalnego przydziału P na podstawie tablicy $\mathtt{A}$ może zostać wykonane w czasie $O(m + o)$ na

$m + o + 1$ procesorach. Ostatecznie, złożoność czasowa algorytmu 2 to

$$O(\log(m+o) + (o+m)\log m + m + o) = O((o+m)\log m), \tag{17}$$

zakładając wykonanie na

$$\max\left\{(o+m)^2, (o+m)m, o+m\right\} = (o+m)^2 \tag{18}$$

procesorowej CREW PRAM. □

5. Eksperymenty obliczeniowe

W celu eksperymentalnej ewaluacji zaproponowanego algorytmu równoległego, dokonano jego implementacji w języku C++. Eksperymenty przeprowadzono na komputerze pod kontrolą systemu Windows 10 Education, wyposażonym w sześciordzeniowy procesor Intel Core i7-4930K CPU @3,4GHz, 32GB RAM. Instancje testowe o parametrach $o \in \{200, 100, 50, 20, 10\}$ i $m = 3$ oraz $o = \{30, 20, 10\}$ i $m \in \{3, 4\}$ zostały wygenerowane losowo. Testy przeprowadzano korzystając z $p = 2, 3, \ldots, 12$ wątków. Mierzono: $T_B(p)$ — czas budowy grafu $\mathcal{B}$ za pomocą alg. 2, oraz $T_{AP}(p)$ — czas rozwiązywania problemu za pomocą alg. 3. Następnie obliczono przyspieszenia względne

$$S_{TOTAL}(p) = \frac{T_B(1) + T_{AP}(1)}{T_B(p) + T_{AP}(p)}, \qquad S_{AP}(p) = \frac{T_{AP}(1)}{T_{AP}(p)}.$$

Uzyskane rezultaty przedstawiono w tabeli 1. Dla najmniejszej zbadanych instancji problemu ($o = 20,\ m = 3$) narzut równoległej wersji algorytmu budowy grafu $\mathcal{B}$ powodował występowanie przyspieszenia niższego niż 1. Dla każdej z pozostałych instancji, przyspieszenie było większe od jedności i osiągało ponad 4 dla $m = 3$ oraz $o \geqslant 50$. W prawie każdym z przypadków przyspieszenie dla alg. 2 było niższe niż alg. 3, tj. $S_{TOTAL}(p) < S_{AP}(p)$. Może to być rezultatem dużej liczby żądań alokacji pamięci przez alg. 2, realizowanej przez system sekwencyjnie.

Tabela 1. Uzyskane przyspieszenia dla instancji AP z m maszynowymi gniazdami

	$m = 3$								$m = 5$					
	$S_{TOTAL}(p)$				$S_{AP}(p)$				$S_{TOTAL}(p)$			$S_{AP}(p)$		
p \ o	200	100	50	20	200	100	50	20	30	20	10	30	20	10
2	1,68	1,85	1,95	0,94	1,82	1,89	1,84	1,75	1,65	1,48	1,42	1,65	1,70	2,12
3	2,11	1,97	1,92	0,94	2,36	2,52	1,86	2,32	2,08	1,76	1,52	2,13	2,07	2,23
4	2,43	2,46	1,95	0,87	2,81	3,29	2,29	3,45	2,30	1,89	1,32	2,40	2,24	3,84
5	2,56	3,00	2,36	0,62	3,04	2,85	3,09	2,84	2,70	2,08	1,81	3,09	2,72	2,87
6	2,71	3,65	2,27	0,60	3,26	4,05	2,92	2,87	2,69	2,21	1,62	3,01	2,73	3,63
7	2,95	3,29	2,43	0,51	3,63	4,83	3,56	2,72	2,74	2,21	1,76	3,08	2,93	3,19
8	3,15	3,32	2,87	0,43	3,93	4,13	3,99	2,56	2,83	2,44	1,64	3,07	3,07	4,14
9	3,38	3,02	2,53	0,35	4,21	4,21	3,69	2,04	3,18	2,50	1,58	3,08	3,25	2,91
10	3,73	3,29	2,70	0,28	4,61	4,78	4,24	2,47	3,20	2,52	1,58	3,64	3,45	2,94
11	3,82	3,70	2,30	0,35	4,85	4,77	3,66	2,65	3,28	2,55	1,47	3,48	3,70	4,03
12	3,69	3,58	2,24	0,32	4,53	4,13	3,64	2,28	2,97	2,46	1,53	3,31	2,98	3,23

6. Wnioski i uwagi

W pracy zaprezentowano równoległy algorytm dla AP w wielomaszynowym gnieździe. Ze względu na swój dwuetapowy charakter, szczególnie dobrze nadaje się gdy konieczne jest wielokrotne rozwiązywanie AP dla instancji o tym samym rozmiarze. Zastosowanie takie jest istotne praktycznie, jako że w procesie produkcyjnym często obok wyznaczania optymalnego przydziału operacji do maszyn, optymalizowana jest również kolejność ich wykonywania. W eksperymentach obliczeniowych uzyskano przyspieszenia do 4,85 na sześciordzeniowym procesorze, z perspektywą skalowalności algorytmu do większej liczby procesorów dla instancji o nietrywialnym rozmiarze.

Praca została sfinansowana ze środków dla rozwoju młodych naukowców.

LITERATURA

1. Aringhieri R., Landa P., Soriano P., Tànfani E., Testi A.: A two level metaheuristic for the operating room scheduling and assignment problem. Computers and Operations Research, 54, 2015, str. 21–34.
2. Bożejko W., Gnatowski A., Idzikowski R., Wodecki M.: Cyclic flow shop scheduling problem with two-machine cells. Archives of Control Sciences, 27(2), 2017, str. 151–168.
3. Bożejko W., Gnatowski A., Klempous R., Affenzeller M., Beham A.: Cyclic scheduling of a robotic cell. W: 7th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2016—Proceedings, 2016, str. 379–384.
4. Bożejko W., Gnatowski A., Niżyński T., Wodecki M.: Tabu Search Algorithm with Neural Tabu Mechanism for the Cyclic Job Shop Problem. W: Lecture Notes in Computer Science 9119, Springer International Publishing, Cham, 2016, str. 409–418.
5. Bożejko W., Gnatowski A., Wodecki M.: Wielomianowy algorytm przydziału dla wielomaszynowego gniazda produkcyjnego. W: DMMS 2017 & ZTS XX Conference Proceedings, Zakopane, September 26-30, 2017, ed. Tadeusz Sawik, Agencja Reklamowo-Wydawnicza „Ostoja", Cianowice, 2017, str. 229–237.
6. Bożejko W., Hejducki Z., Uchroński M., Wodecki M.: Solving the Flexible Job Shop Problem on Multi-GPU. Procedia Comput. Sci., 9, 2012, str. 2020–2023.
7. Bożejko W., Pempera J., Wodecki M.: Parallel Simulated Annealing Algorithm for Cyclic Flexible Job Shop Scheduling Problem. W: Lecture Notes in Artificial Intelligence 9120, Springer International Publishing, 2015, str. 603—612.
8. Gultekin H., Akturk M.S., Karasan O.E.: Cyclic scheduling of a 2-machine robotic cell with tooling constraints. Eur. J. Oper. Res., 174(2), 2006, str. 777–796.
9. Gultekin H., Akturk M.S., Karasan O.E.: Scheduling in a three-machine robotic flexible manufacturing cell. Comput. Oper. Res., 34(8), 2007, str. 2463–2477.
10. Majumder A., Laha D.: A new cuckoo search algorithm for 2-machine robotic cell scheduling problem with sequence-dependent setup times. Swarm Evol. Comput., 28, 2016, str. 131–143.