Problemy równoległej optymalizacji dyskretnej

PROBLEMY RÓWNOLEGŁEJ OPTYMALIZACJI DYSKRETNEJ

Redaktorzy:

Leon Słomiński

Ignacy Kaliszewski

1. Wprowadzenie

Niniejsza książka prezentuje wyniki prac prowadzonych w Zakładzie Programowania Matematycznego Instytutu Badań Systemowych PAN w latach 1986-92, w ramach tematu badawczego "*Optymalizacja na strukturach kombinatorycznych z uwzględnieniem obliczeń równoległych*". Podjęcie tego tematu, którym kierował pierwszy z współredaktorów, poprzedziło roczne wspólne seminarium Zakładu Programowania Matematycznego IBS PAN oraz Pracowni Metod Numerycznych Instytutu Podstaw Informatyki PAN. Niektóre pomysły dyskutowane podczas seminarium znalazły swoje odbicie w treści tej książki.

Celem książki jest przybliżenie polskiemu Czytelnikowi problematyki obliczeń równoległych w zadaniach optymalizacji, a specjalnie w zadaniach optymalizacji dyskretnej oraz uzupełnienie, chociażby częściowe, luki jaka jest w tym zakresie zauważalna w krajowej literaturze. Z publikacji w języku polskim poświęconych w całości lub w części metodom i algorytmom równoległym optymalizacji dyskretnej można wymienić pozycje [1 ÷ 3]. Postawiony cel chcemy osiągnąć przez zaprezentowanie wyboru zadań optymalizacyjnych z zakresu naszych zainteresowań i przedstawienie, na ich przykładzie, problemów związanych z konstrukcją algorytmów równoległych. Problemy te ukazujemy na tle znacznie szerszego wachlarza zagadnień związanych z obliczeniami równoległymi.

Książka ma następujący układ. Rozdział drugi zawiera podstawowe pojęcia związane z obliczeniami równoległymi i z maszynami równoległymi. Przedstawiono w nim najważniejsze modele obliczeń równoległych, typowe architektury maszyn równoległych i ich charakterystyki, a także podstawowe wyniki teorii złożoności dla

algorytmów równoległych. Dopełnieniem tego rozdziału jest omówienie nowatorskiej realizacji sprzętowej - transputera, którego sieci wydają się szczególnie przydatne do rozwiązywania zadań optymalizacji dyskretnej za pomocą ogólnych i wąsko ukierunkowanych algorytmów równoległych. W rozdziale trzecim przedstawiono algorytmy równoległe dla rozwiązywania zadania wyznaczania dendrytu minimaksowego w grafie skierowanym z wagami na łukach. W celu pokazania związku, który zachodzi między złożonością algorytmu i wybranym modelem maszyny równoległej, użyto różnych modeli maszyny i różnych pierwowzorów algorytmów sekwencyjnych. Rozdział czwarty przedstawia algorytmy wyznaczania elementu maksymalnego w skończonym zbiorze wektorów oraz opis ich realizacji na sieci transputerów, wraz z wynikami testów numerycznych. W rozdziale piątym opisano asynchroniczny algorytm równoległy, do rozwiązywania algebraicznego zagadnienia k - przydziału, przedstawiono realizację tego algorytmu na sieci transputerów oraz wyniki eksperymentu numerycznego. Kolejny, szósty rozdział prezentuje język programowania równoległego MODEST. Jest to język ogólnego zastosowania zawierający mechanizmy do uruchamiania procesów i kontroli przebiegu obliczeń równoległych. Dwa dodatki zawierają: Dodatek 1 - Polsko - angielski słownik nazw i pojęć z zakresu obliczeń równoległych, Dodatek 2 - Listę artykułów i raportów ZPM IBS PAN, które ukazały się w latach 1986 - 1992 i które dotyczą problematyki tej książki.

Warszawa, styczeń 1993.

Leon Słomiński
Ignacy Kaliszewski

Literatura

1. Błażewicz J. (1988): Złożoność obliczeniowa problemów kombinatorycznych. WNT, Warszawa.
2. Słomiński L., Kaliszewski I. (1988): "Problemy obliczeń równoległych". *Prace IBS PAN*, 168, Warszawa.
3. Sysło M.M. (1985): "Maszyny i algorytmy równoległe". *Raport Instytutu Informatyki Uniwersytetu Wrocławskiego*, 154, Wrocław.

3. Algorytmy równoległe dla znajdowania minimaksowego dendrytu w ważonym grafie skierowanym.

3.1. Wstęp

Przedstawiamy trzy algorytmy równoległe rozwiązujące zadanie znalezienia minimaksowego dendrytu w grafie skierowanym o n wierzchołkach i m łukach, z wagami liczbowymi przypisanymi łukom. Za punkt wyjścia bierzemy dwa algorytmy sekwencyjne rozwiązujące to zadanie, każdy o najlepszej złożoności w swojej klasie - bezpośredni algorytm progowy o złożoności $O(m\log n)$ i algorytm pośredni, o złożoności $O(n^3)$, który otrzymujemy po niewielkiej modyfikacji algorytmu znajdowania macierzy wartości dróg minimaksowych (wąskich gardeł) między wszystkimi parami wierzchołków grafu.

Interesujące nas algorytmy równoległe otrzymujemy jako odpowiedniki algorytmów sekwencyjnych. Sekwencyjny algorytm progowy zostanie przedstawiony w dwóch odmianach modelu *P-RAM*: jako algorytm równoległy *SIMD-CREW* o złożoności $O(n\log^2 n)$, przy liczbie procesorów $O(m)$, i jako algorytm równoległy *SIMD-CRCW* o złożoności $O(n\log n)$, przy tej samej co poprzednio liczbie procesorów. Odpowiednikiem równoległym sekwencyjnego algorytmu pośredniego będzie algorytm *SIMD-CREW* o złożoności $O(\log^2 n)$, przy liczbie procesorów $O(n^3)$.

Po sformułowaniu problemu (pkt. 3.1) przedstawiamy schemat sekwencyjnego algorytmu progowego (pkt. 3.2), który następnie poddajemy zrównolegleniu na maszynie *SIMD-CREW* i *SIMD-CRCW*, (pkt. 3.3). W punkcie 3.4 przedstawiamy algorytm równoległy pośredni (algorytm znajdowania dróg minimaksowych), będący

także algorytmem *SIMD-CREW*, i podajemy krótką charakterystykę innych algorytmów dróg minimaksowych o identycznej lub lepszej złożoności. Podsumowanie i dyskusję niektórych aspektów złożoności obliczeń zawiera punkt 3.5.

3.2. Założenia i sformułowanie problemu

Dany jest skończony graf skierowany $G(V, A, \omega)$, gdzie: V jest zbiorem wierzchołków o mocy n, A jest zbiorem łuków o mocy m złożonym z uporządkowanych par wierzchołków $(v_i, v_j) = a_{ij} = a_k$, $v_i, v_j \in V$, $i, j = 1,\ldots,n$, $i \neq j$, $k = 1,\ldots,m$, (w ogólnym przypadku $m = n^2$); ω-jest funkcją, która przyporządkowuje każdemu łukowi nieujemną liczbę rzeczywistą - wagę, ω: $A \Rightarrow R^+$. Wartości funkcji ω, $(\omega_1,\ldots,\omega_m)$, tworzą zbiór Ω. Zakładamy, że graf $G(V, A, \omega)$ ma dokładnie jeden wierzchołek, $r \in V$, do którego nie prowadzi żaden łuk, i który jest połączony drogą skierowaną z każdym z pozostałych $(n-1)$ wierzchołków. Wierzchołek r jest nazywany korzeniem. W celu uproszczenia zapisu, przyjmujemy iż wierzchołki są ponumerowane liczbami od 1 do n, (korzeń ma zawsze numer 1) a graf jest zadany przez listy sąsiedztwa $L(i)$ dla wierzchołków i listę wag dla łuków.

Drzewem grafu $G(V, A, \omega)$ nazywa się podgraf złożony z podzbioru zbioru V (zawierającego korzeń) i odpowiednich łuków, taki że każdy wierzchołek podzbioru jest osiągalny wzdłuż drogi skierowanej o początku w korzeniu.

Dendrytem grafu $G(V, A, \omega)$ nazywa się drzewo o n wierzchołkach i o $(n-1)$ łukach (każdy wierzchołek grafu jest osiągalny wzdłuż jedynej drogi skierowanej o początku w korzeniu).

ZADANIE. W grafie $G(V, A, \omega)$ znaleźć dendryt minimaksowy, to znaczy taki podgraf $D^0 = D(V, A_{D^0})$, że:

$$\underset{a_k \in A_D^0}{k:\ max}\{\omega_k\} = \underset{D \in \mathcal{D}}{min} \underset{k:\ a_k \in A_D}{max}\{\omega_k\}, \tag{1}$$

gdzie: $D = D(V, A_D)$ jest dowolnym dendrytem, A_D jest zbiorem łuków dendrytu, a $\mathcal{D}$ jest zbiorem wszystkich dendrytów grafu.

Zadaniem związanym blisko z (1) jest zadanie następujące: W grafie $G(V, A, \omega)$ znaleźć dendryt minimalny, to znaczy taki podgraf $D' = D'(V, A_{D'})$, że:

$$\sum_{k:\ a_k \in A_{D'}} \omega_k = \underset{D \in \mathcal{D}}{min} \left(\sum_{k:\ a_k \in D} \omega_k \right). \tag{2}$$

Uwagi o związku między zadaniem (1) i zadaniem (2):

a) dendryt minimalny grafu skierowanego nie jest, w przypadku ogólnym, dendrytem minimaksowym (w przeciwieństwie do grafu nieskierowanego); b) każde drzewo dróg minimaksowych, które łączą korzeń z pozostałymi $(n - 1)$ wierzchołkami jest, z definicji, dendrytem minimaksowym, ale nie odwrotnie, [9].

Z uwagi a) wynika że, dla grafu skierowanego zadania (1) i (2) są zadaniami odrębnymi.

3.3. Sekwencyjny algorytm progowy

Sekwencyjne algorytmy dla rozwiązania zadania (1), wymienione we Wstępie, przedstawiono w pracach [3,7,9]. Algorytmy przedstawione w [3,9] są algorytmami progowymi. Algorytm opisany w [3] korzysta z metody przeglądu grafu w głąb, a algorytm z pracy [9] - z metody przeglądu grafu wszerz. Nakład obliczeń w obydwu przypadkach jest $O(m \log n)$. Algorytm opisany w [7] jest modyfikacją algorytmu dla znajdowania drzewa najkrótszych dróg z korzenia do pozostałych wierzchołków, polegającą na wykorzystaniu

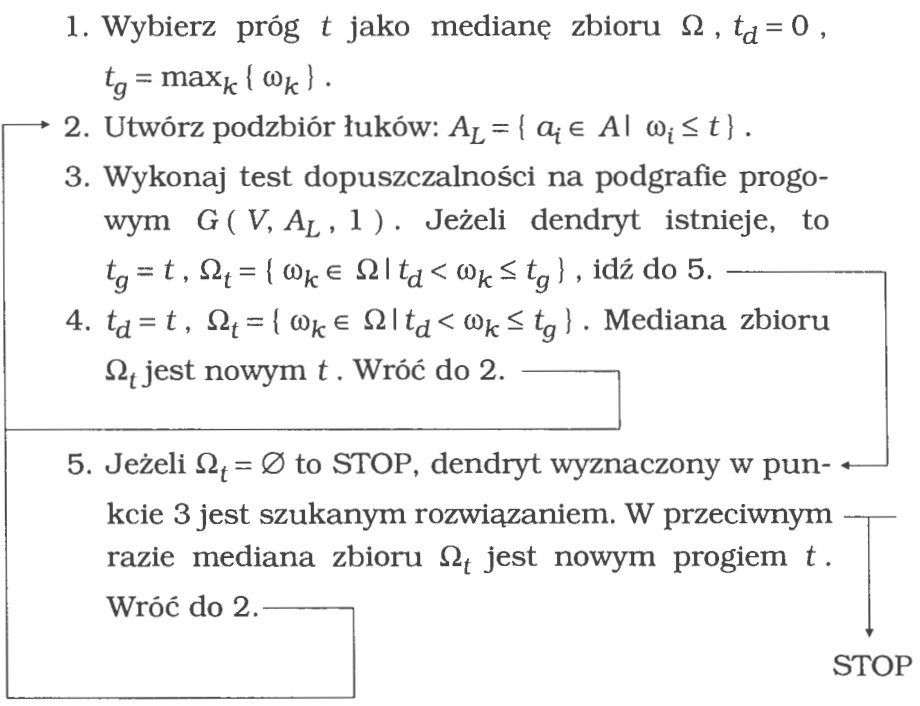
struktury danych o nazwie stóg Fibonacciego. Nakład obliczeń w tym przypadku jest $O(\min\{m + n\log n, m\log^* n\})$, gdzie: $\log^* x = \min\{i \mid \log^{(i)} x \le 1\}$.

Szczegółowy opis algorytmu progowego można znaleźć w cytowanych wyżej pracach. Tutaj ograniczymy się do schematycznej prezentacji jego podstawowych elementów: metody wyboru progu oraz testu dopuszczalności. Zadanie (1) można rozwiązać w ten sposób, że dla każdego elementu ze zbioru Ω, traktowanego jako próg t, tworzy się graf progowy: $G(V, A_L, 1)$, gdzie $A_L = \{a_k \in A \mid \omega_k \le t\}$ i gdzie łukom należącym do A_L przypisuje się wagę jednostkową. Każdy graf progowy poddawany jest testowi dopuszczalności, który polega na znajdowaniu w tym grafie dendrytu skierowanego. Najmniejsza wartość progu zapewniająca pozytywny wynik testu jest wartością optymalną funkcji celu zadania (1), a odpowiedni dendryt jest szukanym dendrytem minimaksowym. Ponieważ zadanie znalezienia dendrytu w grafie skierowanym, przy zastosowaniu metod przeglądu grafu (w głąb, wszerz), ma złożoność $O(m)$, [1], to złożoność całego algorytmu jest $O(m^2)$. Złożoność tę można obniżyć do $O(m\log n)$ przez bardziej oszczędny sposób wyboru progu. Sposób ten polega na wyborze elementu środkowego w uporządkowanym zbiorze Ω i w jego uporządkowanych podzbiorach. Uporządkowanie m-elementowego zbioru ma złożoność $O(m\log m)$, wybranie w nim elementu środkowego ma nakład $O(1)$. Próg można wybierać także w inny sposób - przez znajdowanie mediany zbioru. Złożoność w tym przypadku jest $O(m)$, [1].

Niech t będzie aktualnym progiem. Jeżeli w grafie progowym $G(V, A_L, 1)$ istnieje dendryt, to następnym progiem staje się mediana podzbioru $\Omega_t = \{\omega_k \in \Omega \mid t_d < \omega_k \le t_g\}$, gdzie: $t_d = 0$, $t_g = t$. To postępowanie powtarzamy aż do ustalenia progu, np. t_1, dla

którego dendrytu nie ma. W tym przypadku $t_d := t_1$, a $t_g = t_+$, gdzie t_+ jest ostatnią wartością progu, dla której dendryt istniał. Łatwo zauważyć, że opisane postępowanie kończy się po $O(\log n)$ krokach.

Sekwencyjny algorytm progowy można przedstawić poniższym schematem:



Złożoność obliczeń sekwencyjnego algorytmu progowego jest $O(m \log n)$ gdyż:

- znalezienie mediany m-elementowego zbioru łuków (kroki 1,4,5), przez jego uporządkowanie, wymaga $O(m \log n)$ obliczeń,
- wykonanie testu dopuszczalności kosztuje $O(m)$ obliczeń (krok 3),

- test dopuszczalności ma nakład obliczeń $O(\log n)$,
- inne działania (aktualizacja list sąsiedztwa i grafu progowego) wymagają $O(m)$ obliczeń.

3.4. Algorytmy równoległe

Algorytmy równoległe D1 i D2 są odpowiednikami algorytmu progowego opisanego w poprzednim punkcie. Algorytm D3 jest prostą modyfikacją równoległego algorytmu mnożenia macierzy.

3.4.1. Algorytm progowy D1 (*SIMD-CREW*)

Równoległy algorytm progowy D1 jest algorytmem *SIMD-CREW*. W tym modelu obliczeń realizowane są procedury: wyboru progu, budowy grafu progowego i wykonania testu na istnienie dendrytu. Złożoność obliczeniowa tego algorytmu jest $O(n\log^2 n)$ przy użyciu $O(m)$ procesorów.

Równoległy wybór progu i utworzenie grafu $G(V, A_L, 1)$. Jeżeli próg jest wybierany jako element środkowy uporządkowanego zbioru Ω, to złożoność obliczeniowa wyboru jest $O(\log n)$. Składa się na nią $O(\log n)$ operacji na jednokrotne uporządkowanie zbioru łuków i $O(\log n)$-krotny wybór progu kosztem $O(1)$ działań dla każdego wyboru. Jednokrotny wybór progu, jako mediany zbioru Ω, kosztuje $O((\log\log n)^2)$ obliczeń i musi on być powtórzony $O(\log n)$ razy. Do porządkowania zbioru Ω można zastosować równoległy algorytm scalający opisany w [4]. Do wyboru mediany można użyć algorytmu podanego w [5]. W obydwu przypadkach algorytmy równoległe są algorytmami *SIMD-CREW* i potrzebna liczba procesorów, do ich wykonania jest $O(m)$. Utworzenie podgrafu $G(V, A_L, 1)$ przez aktualizację list sąsiedztwa $L_t(i)$ i listy wag łuków można wykonać nakładem $O(1)$ operacji, przy użyciu $O(m)$ procesorów.

Test na istnienie dendrytu - znajdowanie dendrytu w grafie progowym metodą równoległego przeglądu grafu skierowanego wszerz.

Procedurę tę opiszemy bardziej szczegółowo.

WEJŚCIE

Listy sąsiedztwa wierzchołków $L_t(i)$, $i \in V$, korygowane wraz ze zmianą progu (przed rozpoczęciem pracy algorytmu $t = \infty$ i lista wag przypisanych łukom.

WYJŚCIE

DENDRYT

n-wymiarowa tablica, gdzie $DENDRYT[i] = j$, $i, j \in V$, $i \neq j$, oznacza, że w dendrycie, wierzchołek j jest wierzchołkiem bezpośrednio poprzedzającym - ojcem - wierzchołka i. Korzeń nie ma ojca. Przyjmujemy na stałe, że $DENDRYT[1] = \infty$.

OPT

wartość funkcji celu dla optymalnego dendrytu.

PRZEJŚCIOWE STRUKTURY DANYCH

OZ [*i*]

n-wymiarowa tablica, gdzie:
$OZ[i] = 1$ oznacza, że wierzchołek i jest otwarty i oczekuje na przegląd.
$OZ[i] = 2$ oznacza, że wierzchołek i jest w trakcie przeglądania (aktywny).
$OZ[i] = 0$ oznacza, że wierzchołek i jest zamknięty, (po przeglądnięciu).

Przed rozpoczęciem przeglądu wszystkie wierzchołki są otwarte, $OZ[i] = 1$, $i \in V$. Aktywnym może się stać tylko wierzchołek otwarty. Zamknięcie wszystkich wierzchołków oznacza zakończenie przeglądu i znalezienie dendrytu. Zakończenie przeglądu, przy chociażby jednym niezamkniętym wierzchołku, oznacza, że podgraf nie ma dendrytu.

$PRETENDENT_j[i]$

$j \in V$: n-wymiarowa tablica, gdzie $PRETENDENT_j[i] = 1$ oznacza, że wierzchołek o numerze i może być, w dendrycie, ojcem dla wierzchołka j.
Dla korzenia tablica *PRETENDENT* nie jest definiowana, a dla $i = j$ $PRETENDENT_i[i] = \infty$.

$$STAN = \begin{cases} 0, & \text{jeżeli } \min\limits_{i \neq 1} \{ OZ[i] \} < 2, \\ 1, & \text{w przeciwnym razie.} \end{cases}$$

Algorytm *D1*

INICJALIZACJA

Wartości początkowe tablic i zmiennych:

PARALLEL

$STAN = 0$,
$OZ[i] = 1$, $i \in V \setminus \{1\}$, $OZ[1] = 2$,
$PRETENDENT_j[i] = 0$, $i, j \in V \setminus \{1\}$, $i \neq j$,
$DENDRYT[i] = 0$, $i \in V \setminus \{1\}$,
$OPT = \infty$.

END PARALLEL

ITERACJA 1-sza.

PARALLEL

Procesory $P(1, j)$, $j \in L_t(1)$, dla których $\omega_{1j} \leq t$ wykonują:

$OZ[1] := 0$,

$OZ[j] := 2$,

$DENDRYT[j] := 1$

END PARALLEL

ITERACJA *l*-**ta,** ($l > 1$)

Jeżeli $STAN = 0$ i $\min_{i \neq 1}\{DENDRYT[i]\} = 0$, to próg t należy zwiększyć.

Jeżeli $STAN = 0$ i $\min_{i \neq 1}\{DENDRYT[i]\} > 0$ to, pod warunkiem że podzbiór $\Omega_t \neq \varnothing$, wartość progu t należy zmniejszyć.

W przeciwnym razie:

PARALLEL

Procesory $P(i, j)$ takie, że: $i \in V_c$ i $\omega_{ij} \leq t$, gdzie V_c jest podzbiorem wszystkich wierzchołków aktywnych wykonują:

$PRETENDENT_j[i] := 1$

END PARALLEL

PARALLEL

Procesory $P(i_0, j)$, $i_0 = min\{i \in V \mid PRETENDENT_j[i] = 1\}$, dla których $OZ[j] = 1$, wykonują:

$OZ[j] := 2$,

$OZ[i_0] := 0$,

$DENDRYT[j] := i_0$

END PARALLEL

PARALLEL

Jeżeli $\min_{i \neq 1} \{OZ[\,i\,]\} < 2$, to

$STAN = 1$

END PARALLEL

END ITERACJA *l*-**ta**

END Algorytm *D1*

Liczba iteracji dla jednej wartości progu jest $O(n)$. Nakład obliczeń jednej iteracji jest $O(\log n)$, gdyż taką złożoność mają działania na tablicach: $PRETENDENT_j$, OZ i $DENDRYT$ wykonane za pomocą algorytmu równoległego stosującego łączenie elementów w pary (algorytm drzewiasty) opisanego w rozdziale 2.6. Stąd złożoność obliczeniowa jednej iteracji jest $O(n\log n)$, a złożoność całego algorytmu D1 - $O(n\log^2 n)$, przy użyciu $O(m)$ procesorów.

3.4.2. Algorytm progowy D2 (*SIMD-CRCW*)

W tym modelu jest możliwy jednoczesny odczyt/zapis z/do tej samej komórki pamięci przez więcej niż jeden procesor. Przyjmiemy konwencję zapisu odpowiadającą modelowi *P-RAM-CRCW*(d) (patrz: [8] i rozdz. 2.3): w przypadku konfliktu dostępu do pamięci zapisu dokona jeden, dowolny, procesor. Algorytm D2 otrzymamy z algorytmu D1 przez rezygnację z tablicy $PRETENDENT_j$, $j \in V$. Spośród kilku kandydatów do ojcostwa dla danego wierzchołka zapisu dokona jeden, dowolny, procesor bezpośrednio w odpowiednim miejscu tablicy *DENDRYT*. Zmiana ta dotyczy *l*-tej iteracji, która przyjmuje postać:

Algorytm *D2*

ITERACJA *l*-**ta,** ($l > 1$)

Jeżeli $STAN = 0$ i $\min_{i \neq 1} \{ DENDRYT[i] \} = 0$, to próg t należy zwiększyć.

Jeżeli $STAN = 0$ i $\min_{i \neq 1} \{ DENDRYT[i] \} > 0$ to, pod warunkiem że podzbiór $\Omega_t \neq \emptyset$, wartość progu t należy zmniejszyć.

W przeciwnym razie:

PARALLEL

Procesory $P(i,j)$ takie, że: $i \in V_c$, $j \in V_1$, $\omega_{ij} \leq t$, gdzie V_c jest podzbiorem wszystkich wierzchołków aktywnych, a V_1 jest podzbiorem wierzchołków niezamkniętych, podejmują próbę zapisu do komórek $DENDRYT[j]$, uwieńczoną dla każdej komórki sukcesem pewnego procesora, np. dla j_0 może to być $P(i_0, j_0)$, poza tym próba jednoczesnego zapisu dotyczy komórek tablicy *OZ*:

$DENDRYT[j_0] = i_0$

$OZ[j] := 2$,

$OZ[i] := 0$,

END PARALLEL

PARALLEL

Jeżeli $\min_{i \neq 1} \{ OZ[i] \} < 2$, to

$STAN = 1$

END PARALLEL

END ITERACJA *l*-**ta.**

END ALgorytm *D2*

Złożoność obliczeniowa algorytmu D2 jest $O(n\log n)$, gdyż złożoność l-tej iteracji jest w tym przypadku $O(1)$, a liczba iteracji nie ulega zmianie i jest $O(n)$. Złożoność pojedynczej iteracji ulega zmniejszeniu w wyniku zastosowania mocniejszego modelu maszyny równoległej - *SIMD -CRCW*. W tym modelu złożoność obliczeń związanych ze sprawdzaniem wartości zmiennej *STAN* i warunków dla tablicy *DENDRYT* jest $O(1)$, przy $O(m)$ procesorach (patrz: algorytm *MIN*1 dla modelu *SIMD -CRCW* - rozdz. 2.4).

3.4.3. Algorytm D3 (*SIMD -CREW*)

Algorytm D3 otrzymamy przez wykorzystanie zależności (rozdz. 3.2), która utożsamia każde drzewo skierowane dróg minimaksowych, od korzenia do pozostałych $(n-1)$ wierzchołków, z pewnym skierowanym dendrytem minimaksowym. Dla potrzeb bieżącego paragrafu przyjmiemy, że graf $G(V, A, \omega)$ jest zadany przez macierz wag W przypisanych łukom, której elementy definiujemy następująco:

$$w_{ij} = \omega_{ij} \quad \text{dla } (i,j) \in A,$$

$$w_{ij} = +\infty \quad \text{dla } (i,j) \notin A.$$

Chcemy obliczyć macierz wartości minimaksowych $W^* = [\, w_{ij}^{\,*} \,]$, gdzie $w_{ij}^{\,*}$ jest najmniejszą wagą (wąskim gardłem) na drodze z $i \in V$ do $j \in V$, wśród wszystkich możliwych dróg łączących tę parę wierzchołków, i macierz samych dróg minimaksowych (zawierającą interesujący nas dendryt minimaksowy) $S^* = [\, s_{ij}^{\,*} \,]$, gdzie $s_{ij}^{\,*}$ jest numerem następnego wierzchołka po i, na drodze z i do j.

Macierz W^* otrzymujemy w wyniku ciągu przekształceń macierzy $W^{(l)} = [\, w_{ij}^{(l)} \,]$, gdzie:

$$w_{ij}^{(0)} = w_{ij},$$

$$w_{ij}^{(l)} = min\{\, w_{ij}^{(l-1)}, min\{\, w_{il}^{(l-1)}, w_{lj}^{(l-1)} \}\},$$

$l = 0, 1, \ldots, n,$

przy czym $w_{ij}^{(l)}$ jest wartością wąskiego gardła dla drogi z wierzchołka i do wierzchołka j, która może przechodzić jedynie przez wierzchołki należące do zbioru $\{ 1, \ldots, l \} \subseteq V$.

Macierz dróg S^* powstaje następująco:

$$s_{ij}^{(0)} = \begin{cases} j, \text{ jeżeli } & w_{ij} \neq \infty \\ 0, \text{ jeżeli } & w_{ij} = \infty \end{cases}.$$

Jeżeli dla kroku o numerze l mamy: $min\{\, w_{il}^{(l)}, w_{lj}^{(l)} \} \leq w_{ij}^{(l)}$, to:

$$s_{ij}^{(l)} := s_{il}^{(l)}.$$

Algorytm sekwencyjny D3S wykonujący przekształcenia dane wzorami (4), (5) i (6) ma postać:

INICJALIZACJA

Ustanowienie wartości początkowych macierzy W i S.

Algorytm *D3S*

for $l = 1$ **to** n **do**
for $i = 1$ **to** n **do**

if $w_{il} \neq \infty$ **then**

for $j = 1$ **to** n **do**

$w_{ij} := min\{ w_{ij}, min\{w_{il}, w_{lj}\}\}$.

if $min\{ w_{il}, w_{lj}\} < w_{ij}$ **then**

$s_{ij} := s_{il}$.

END Algorytm *D3S*

Złożoność obliczeniowa algorytmu D3S jest $O(n^3)$, o czym decydują trzy pętle **for**.

Algorytm równoległy D3 (*SIMD-CREW*) jest prostą konsekwencją algorytmu D3S i przy założeniu, że liczba procesorów jest $O(n^3)$, można go przedstawić następująco (pomijamy inicjalizację):

Algorytm *D3R*

for $l = 1$ **to** $\lceil \log n \rceil$ **do**

PARALLEL1

Procesory P_{ikj}, wykonują:

$c_{ikj}^{(l)} = min\{ w_{ik}^{(l)}, w_{kj}^{(l)} \}$

END PARALLEL1

PARALLEL2

Procesory P_{ikj} i P_{ik} wykonują:

$$w_{ij}^{(l+1)} = min\left\{ \min_{k} (c_{ikj}^{(l)}) , w_{ij}^{(l)} \right\}$$

END PARALLEL2

PARALLEL3

Procesory P_{ikj} i P_{ik} wykonują:

if $\min_k (c_{ikj}^{(l)}) < w_{ij}^{(l)}$ **then**

$s_{ij}^{(l)} := s_{ik_0}^{(l)}, \ (\min_k (c_{ikj}^{(l)}) = c_{ik_0 j}^{(l)})$

END PARALLEL3

END Algorytm *D3R*

Złożoność tego algorytmu jest $O(\log^2 n)$, gdyż każda z trzech instrukcji **PARALLELi,** $(i = 1, 2, 3)$, wymaga $O(\log n)$ obliczeń przy liczbie procesorów $O(n^3)$ lub $O(n^2)$ (wyznaczanie elementu minimalnego metodą łączenia w pary). Liczba iteracji pętli **for** ma następujące uzasadnienie. Każdy element $w_{ij}^{(l)} \neq \infty$ w macierzy $W^{(l)}$ świadczy, że w grafie istnieje droga, z wierzchołka i do wierzchołka j, o długości nie większej niż $2^{(l)}$. Ponieważ w naszym przypadku droga ta może się składać z co najwyżej $n-1$ łuków, stąd maksymalna wartość dla l jest $\lceil \log n \rceil$.

Z istoty algorytmu D3, która polega na wykonaniu ciągu mnożeń macierzy przez siebie z zastąpieniem operacji + i * operacjami *min* i *max* (patrz: wzór (4)), wynika że można do niego sprowadzić każdy algorytm wszystkich najkrótszych dróg będący modyfikacją algorytmu mnożenia macierzy. Dla przykładu, algorytm *SIMD-hipersześcian*, [6], ma złożoność $O(\log^2 n)$ przy liczbie procesorów $O(n^3)$.

Zauważmy, że algorytm D3 będący algorytmem *SIMD-CREW* można zrealizować jako algorytm *SIMD-CRCW*, obniżając złożoność

do $O(\log n)$, kosztem zwiększenia liczby procesorów do $O(n^4)$ (instrukcje **PARALLEL1** i **PARALLEL2**).

Zadanie (1) można rozwiązać także, modyfikując algorytm *SIMD-PDB* (*SIMD-podwójne drzewo binarne*), [2], znajdowania najkrótszych dróg z korzenia do pozostałych wierzchołków. Złożoność obliczeń tego algorytmu jest $O(n\log^2 n)$.

3.5. Uwagi końcowe

Podsumowując rozważania tego rozdziału zauważamy, że:

- zadanie znalezienia dendrytu minimaksowego w skierowanym grafie ważonym należy do zadań klasy NC, gdyż algorytm D3 ma złożoność obliczeniową $O(\log^k n)$ przy liczbie procesorów $O(n^l)$;
- metoda przeglądania grafu wszerz nie prowadzi do algorytmu równoległego o wielomianowo-logarytmicznym czasie obliczeń, i należy przypuszczać, że ta metoda nie ma większego znaczenia dla budowy teoriografowych algorytmów równoległych o niskim nakładzie obliczeń;

W Tabeli 3.1 podajemy wartości: przyśpieszenia, efektywności wykorzystania procesorów i kosztu, dla algorytmów D1÷D3. Wartości te odbiegają od wartości idealnych: 1 - dla e_{WP}, $O(m)$ - dla kosztu i p - dla przyśpieszenia.

Najniższą złożoność obliczeniową ma algorytm D3, ale stosunkowo najlepszymi wartościami parametrów charakteryzuje się algorytm D2.

Tabela 3.1. Parametry algorytmów D1÷D3

	$s = t_{sz} / t_r$	$e_{WP} = s / p$	$KOSZT = t_r * p$
D1 *SIMD-CREW*	$n / \log n$	$1 / (n \log n)$	$n^3 \log^2 n$
D2 *SIMD-CRCW*	n	$1/n$	$n^3 \log n$
D3 *SIMD-mnożenie macierzy*	$n^3 / \log^2 n$	$1 / \log^2 n$	$n^3 \log^2 n$

Literatura

1. Aho A.V., Hopcroft J.H., Ullman J.D. (1974): The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Co.
2. Bentley J.L., Kung H.T. (1979): "A Tree Machine for Searching Problems". *Proceedings of 1979 International Conference on Parallel Processing*, 257-266.
3. Camerini P.M. (1978): "The Minimax Spanning Tree Problem and Extensions". *Information Processing Letters*, 20, 10-14.
4. Cole R. (1986): "Parallel Merge Sort". *Foundations of Computer Sciences.*
5. Cole R., Yap C.K. (1985): "A Parallel Median Algorithm". *Information Processing Letters*, 20, 137-140.
6. Dekel E., Nassimi D., Sahni S. (1981): "Parallel Matrix and Graph Algorithms". *SIAM Journal on Computing*, 10, 657-675.
7. Gabow H.N., Tarjan R.E. (1988): "Algorithms for two bottleneck optimization problems". *Journal of Algorithms*, 9, 411-417.
8. Kučera L. (1982): "Parallel Computation and Conflicts in Memory Access". *Information Processing Letters*, 14, 93-96.

9. Słomiński L. (1988): "Algorytmy równoległe znajdowania minimaksowych rozgałęzień". *Zeszyty Naukowe Politechniki Śląskiej. Seria: Automatyka*, z. 94, 287-301.

Dodatek 2

RAPORTY I PUBLIKACJE ZAKŁADU PROGRAMOWANIA MATEMATYCZNEGO IBS PAN DOTYCZĄCE RÓWNOLEGŁEJ OPTYMALIZACJI DYSKRETNEJ (1986-92)

I. Publikacje

1. Bertocchi M., Brandolini L., Słomiński L., Sobczyńska J. (1992): "A Monte-Carlo Approach for 0-1 Programming Problems". *Computing* 48, 259-274.
2. Dudziński K. (1986): "Wybrane równoległe algorytmy kombinatoryczne". *Roczniki PTM Seria III, Matematyka Stosowana XXVIII*, 91-123.
3. Kaliszewski I., Wojtowicz M. (1991) "Modest. A language for parallel programming". *Prace IPI PAN*, 691, Warszawa.
4. Słomiński L., Kaliszewski I. (1988): "Problemy obliczeń równoległych". *Prace IBS PAN*, 168, Warszawa.
5. Słomiński L. (1988): "Algorytmy równoległe znajdowania minimaksowych rozgałęzień". *Zeszyty Naukowe Politechniki Śląskiej, ser. Automatyka* 94, 287-301.
6. Słomiński L. (1988): "Obliczenia równoległe i optymalizacja". W: Materiały Konferencji Naukowej: Problemy współczesnej radiolokacji, WAT, Warszawa, 124-130.
7. Słomiński L. (1989): "Parallel Algorithms for Optimization". W: Proceedings of the 3rd Polish-Finish Symp. on Methodology and Applications of Decision Support Systems, red. R. Kulikowski, IBS PAN, Warszawa, 219-233.
8. Słomiński L. (1990): "Implementacje algorytmów kombinatorycznych na symulowanych sieciach wielomikroprocesorowych".

Zeszyty Naukowe Politechniki Śląskiej, ser. Automatyka, 100, 287-300.

9. Słomiński L. (1990): "Parallel Computing for Flexible Manufactoring Systems". W: Decision Making Models for Management and Manufacturing, red. R. Kulikowski, Omnitech Press, Warszawa, 215-229.
10. Słomiński L. (1990): "Komputery równoległe dla rozwiązywania zadań optymalizacji dyskretnej i sztucznej inteligencji". Materiały konferencji "*Sztuczna inteligencja w zarządzaniu*", Warszawa, listopad 1989, PTBiOS i IBS PAN, red. A.Straszak, Z. Nahorski, C. Iwański, Warszawa, 201-210.

II. Raporty (latami, wg alfabetu)

1986

1. Dudziński K.: "Obliczenia równoległe: Wybrane algorytmy kombinatoryczne". ZPM 20/86, IBS PAN, Warszawa.
2. Grygiel G.: "Obliczenia równoległe: Algorytm równoległy dla zadań przydziału". ZPM 24/86, IBS PAN, Warszawa.
3. Kaliszewski I.: "Obliczenia równoległe: Modele obliczeń w języku Ameba-Pascal". ZPM 23/86, IBS PAN, Warszawa.
4. Kaliszewski I., Słomiński L.: "Obliczenia równoległe: przegląd zagadnień". ZPM 17/86, IBS PAN, Warszawa.
5. Majchrzak J.: "Obliczenia równoległe: uwarunkowania i prognozy". ZPM 21/86, IBS PAN, Warszawa.
6. Słomiński L.: "Obliczenia równoległe: algorytm równoległy wyznaczania dendrytu minimaksowego w grafie skierowanym". ZPM 19/86, IBS PAN, Warszawa.

7. Waluk B.: "Obliczenia równoległe: Zastosowanie niektórych metod dekompozycji macierzy współczynników do rozwiązywania układów równań". ZPM 22/86, IBS PAN, Warszawa.

1987

1. Grygiel G.: "Zadanie przydziału: algorytm równoległy i eksperymentalna ocena liczby iteracji". ZPM 34/87, IBS PAN, Warszawa.
2. Kaliszewski I.: "Parallel counterparts of algorithms for determining minimal elements of discrete sets". ZPM 30/87, IBS PAN, Warszawa.
3. Kaliszewski I., Wojtowicz M.: "Język Modest". ZPM 31/87, IBS PAN, Warszawa.
4. Kulczycka D.: "Równoległe algorytmy b-skojarzeń w drzewach". ZPM 23/87, IBS PAN, Warszawa.
5. Majchrzak J., Skawiński A.: "Sieci komputerowe - przegląd literatury i ocena możliwości przetwarzania równoległego". ZPM 17/87, IBS PAN, Warszawa.
6. Majchrzak J., Skawiński A.: "Pakiet PPLANEUP do organizacji przetwarzania równoległego w sieci D-Link komputerów IBM PC/XT/AT". ZPM 18/87, IBS PAN, Warszawa.
7. Majchrzak J., Skawiński A.: "Propozycja bezgradientowej metody równoległej optymalizacji nieliniowej bez ograniczeń". ZPM 19/87, IBS PAN, Warszawa.
8. Słomiński L.: "Narzędzia optymalizacji równoległej na maszynach sekwencyjnych". ZPM 13/87, IBS PAN, Warszawa.
9. Słomiński L.: "Problemy równoległych algorytmów dla przepływu maksymalnego w sieci". ZPM 14/87, IBS PAN, Warszawa.

1988

1. Gondek H., Słomiński L.: "Implementacja na symulowanej maszynie równoległej typu dwudrzewo algorytmu Dijkstry wyznaczania drzewa dróg najkrótszych w digrafie". ZPM 37/88, IBS PAN, Warszawa.
2. Grygiel G., Kulczycka D.: "Ocena systemu symulacyjnego JUPITER-86". ZPM 24/88, IBS PAN, Warszawa.
3. Grygiel G.: "Symulacja algorytmu równoległego dla zadania przydziału przy pomocy pakietu JUPITER-86". ZPM 44/88, IBS PAN, Warszawa.
4. Kaliszewski I., Kulczycka D., Słomiński L.: "Równoległy algorytm Prima-Dijkstry wyznaczania dendrytu minimalnego: implementacja na symulowanej strukturze typu dwudrzewo". ZPM 5/88, IBS PAN, Warszawa.
5. Kaliszewski I., Słomiński L.: "Problemy obliczeń równoległych". ZPM 38/88, IBS PAN, Warszawa.
6. Kulczycka D.: "Algorytmy równoległe b-skojarzeń w drzewach: implementacja w systemie symulacyjnym Jupiter". ZPM 6/88, IBS PAN, Warszawa.
7. Majchrzak J.: "Pakiet PPLANEUP wersja 2 dla wspomagania przetwarzania równoległego w sieciach D-Link komputerów IBM PC/XT/AT". ZPM 41/88, IBS PAN, Warszawa.
8. Słomiński L.: "Algorytmy równoległe znajdowania minimaksowych rozgałęzień". ZPM 1/88, IBS PAN, Warszawa.
9. Słomiński L.: "Obliczenia równoległe i optymalizacja". ZPM 4/88, IBS PAN, Warszawa.

1989

1. Gondek H., Słomiński L.: "Symulowane odmiany struktury dwudrzewa dla algorytmów najkrótszych dróg w digrafie". ZPM 10/89, IBS PAN, Warszawa.

2. Gondek H.: "Równoległy algorytm mnożenia macierzy. Implementacja i eksperyment obliczeniowy na symulowanej strukturze typu mesh i hypercube o n^2 procesorach". ZPM 11/89, IBS PAN, Warszawa.
3. Grygiel G.: "Równoległość na poziomie podprocedur na przykładzie zadania przydziału". ZPM 13/89, IBS PAN, Warszawa.
4. Grygiel G.: "Program AZP rozwiązujący algebraiczne zagadnienie przydziału - opis funkcjonalny". ZPM 18/89, IBS PAN, Warszawa.
5. Kaliszewski I.: "VM1 - A parallel algorithms to determine minimal elements of discrete sets". ZPM 16/89, IBS PAN, Warszawa.
6. Kulczycka D.: "Algorytmy równoległe w sieci drzew ortogonalnych - realizacja w systemie symulacyjnym JUPITER-86". ZPM 12/89, IBS PAN, Warszawa.
7. Majchrzak J.: "O programowaniu dla transputerów INMOS T800". ZPM 14/89, IBS PAN, Warszawa.
8. Słomiński L.: "Parallel Algorithms for Optimization". ZPM 1/89, IBS PAN, Warszawa.
9. Słomiński L.: "Komputery równoległe dla rozwiązywania zadań optymalizacji i sztucznej inteligencji". ZPM 3/89, IBS PAN, Warszawa.
10. Słomiński L., Sobczyńska J.: "Programy i dane w języku Fortran 77 dla wybranych zadań dyskretnych, z uwzględnieniem potrzeb wektoryzacji". ZPM 4/89, IBS PAN, Warszawa.
11. Sobczyńska J.: "Charakteryzacja wieloprocesorowej struktury Perfect Shuffle. Równoległy algorytm mnożenia macierzy i jego implementacja w tej strukturze". ZPM 9/89, IBS PAN, Warszawa.

1990

1. Grygiel G.: "Implementacja asynchronicznego algorytmu dla zadania przydziału". ZPM 26/90, IBS PAN, Warszawa.

2. Kaliszewski I.: "On determining minimal elements of discrete sets on a network of transputers". ZPM 4/90, IBS PAN, Warszawa.
3. Kulczycka D.: "Prosty algorytm równoległy znajdowania skojarzenia maksymalnego". ZPM 18/90, IBS PAN, Warszawa.
4. Majchrzak J.: "Transputer i sieci wielotransputerowe". ZPM 30/90, IBS PAN, Warszawa.
5. Słomiński L.: "Komunikacja w systemach wieloprocesorowych i jej wpływ na efektywność obliczeń". ZPM 22/90, IBS PAN, Warszawa.

1991

1. Bertocchi M., Butti A., Sergi P., Słomiński L., Sobczyńska J.: "Stochastic and Deterministic Optimization on 0-1 Multiknapsack Problem". *Quaderni del Dipartimento di Mathematica, Statistica, Informatica e Applicazioni*, No 14/91, Bergamo.
2. Kulczycka D.: " Algorytm O(|E|) skojarzenia w grafie dwudzielnym". ZPM 13/91, IBS PAN, Warszawa.

1992

1. Bertocchi M., Butti A., Słomiński L.: "Fixed Precision Random Search Procedure for the Binary Multiknapsack Problem". (Revised version). *Quaderni del Dipartimento di Mathematica, Statistica, Informatica e Applicazioni*, No 18/92, Bergamo.
2. Grygiel G., Kabanow P., Słomiński L., Sobczyńska J.: "Wykorzystanie sieci transputerowych do rozwiązania wielowymiarowego zadania załadunku metodą Monte-Carlo". /w języku rosyjskim/ ZPM 12/92, IBS PAN, Warszawa.