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.

5. Asynchroniczny algorytm równoległy dla zagadnienia przydziału

5.1. Wstęp

Zagadnienie przydziału (oznaczane dalej (ZP)) w klasycznym sformułowaniu polega na znalezieniu optymalnego przydziału pracowników do stanowisk pracy. Występuje ono niejednokrotnie jako problem pomocniczy w algorytmach rozwiązujących inne zadania programowania dyskretnego, np. w zadaniu komiwojażera lub w kwadratowym zadaniu przydziału. Niniejszy rozdział zajmuje się bardzo ogólną wersją (ZP) - współczynniki macierzy kosztów są elementami uporządkowanej półgrupy, zaś funkcja celu ma postać pewnego wielomianu. Takie sformułowanie obejmuje między innymi przypadek liniowy, minimaksowy czy leksykograficzny.

(ZP) można rozpatrywać jako szczególny przypadek zadania przepływu w sieci, tak też potraktowane jest ono np. w pracy [1], gdzie do rozwiązania liniowego zadania przydziału zastosowano asynchroniczny algorytm "aukcyjny", którego efektywność jest obiecująca dla zadań o rzadkiej macierzy kosztów. Algorytmy oparte na zadaniu przepływu wymagają jednak dużej liczby procesorów, co najmniej rzędu rozmiaru zadania.

Paragraf 5.2 zawiera sformułowanie zagadnienia. W paragrafie 5.3 zaprezentowano algorytm *MIMD* dla algebraicznego zadania przydziału. Jest to algorytm progowy, który można realizować już na kilku procesorach. W 5.4 podane są wyniki symulacji przebiegu czasowego tego algorytmu, dzięki której można oszacować uzyskiwane przyspieszenie. Algorytm został przystosowany do realnej maszyny wieloprocesorowej - sieci dwutransputerowej i praktycznie na niej przetestowany. Wyniki tego testu omówione są w paragrafie 5.5.

5.2. Sformułowanie problemu

Niech $S = (S, \oplus, \leq)$ będzie uporządkowaną półgrupą, z przemiennym działaniem $\oplus$ i relacją porządku $\leq$, $A = [a_{ij}]$, $i, j = 1,..., n$ macierzą o wyrazach należących do S , π-zbiorem wszystkich permutacji n-elementowego zbioru $I = \{ 1,..., n \}$.

Algebraiczne k-zagadnienie przydziału (oznaczane dalej przez $AZP (k)$) jest to zadanie minimalizacji w S , polegające na znalezieniu permutacji $\pi_0 \in \Pi$ takiej, że

$$F_A (\pi_0) = min\, F_A (\pi) ,$$

gdzie funkcja F_A zadana jest wzorem

$$F_A (\pi) := \max_{I_k \subseteq I} \bigoplus_{i \in I_k} a_{i\pi(i)} ,$$

I_k jest tu k-elementowym podzbiorem zbioru I .

Tak sformułowane zagadnienie obejmuje liczną klasę zadań optymalizacji, w szczególności klasyczne zadania przydziału, zadanie minimaksowe, zadanie leksykograficzne typu czas-koszt.

Przykładowymi działaniami $\oplus$ są:

(1) $x \oplus y := x + y$ w zbiorze liczb rzeczywistych R .
Odpowiadające tej półgrupie zagadnienie przydziału jest klasycznym liniowym zadaniem przydziału dla $k = n$, natomiast dla $k = 1$ - zadaniem minimaksowym.

(2) $x \oplus y = x * y$ w zbiorze liczb rzeczywistych dodatnich R^+ .

(3) $x \oplus y = max (x, y)$ w R.

(4) $x \oplus y = \sqrt{(x * y + x * y)}$ w R^+ .

(5) $x \oplus y = (x + y) - x * y$ na odcinku $[0,1]$.

(6) $x \oplus y = (x + y) / x * y$ na odcinku $[0,1]$.

(7) W $R \times R^+$ z porządkiem leksykograficznym i działaniem określonym wzorem:

$$(x, y) \oplus (s, t) = \begin{cases} (x, y) & \text{jeśli } x > s, \\ (s, t) & \text{jeśli } x < s, \\ (x, y + t) & \text{jeśli } x = s \end{cases}$$

Przy konstrukcji algorytmów i w dowodzie ich zbieżności potrzebne są dodatkowe założenia o rozpatrywanych obiektach. Szczegółowa dyskusja przeprowadzona jest w [4, 6]. Tam też podane są twierdzenia i konstrukcja algorytmu na nich oparta.

Algorytm dla zadania $AZP(k)$ jest algorytmem progowym, w którym rozwiązuje się ciąg zadań pomocniczych $ZP(b_j)$ skonstruowanych w poniższy sposób : jeśli b jest wyrazem macierzy A, to przez $B(b) := [b_{ij}]$, $i, j = 1,\ldots, n$, oznaczmy macierz progową, której elementy zdefiniowane są następująco:

$$b_{ij} := max(a_{ij}, b)$$

Zadanie $ZP(b)$ polega na znalezieniu permutacji $\pi_1 \in \Pi$ takiej, by

$$G_B(\pi_1) = \min_{\pi \in \Pi} G_B(\pi)$$

gdzie G_B jest funkcją celu zadania pomocniczego:

$$G_B(\pi) := \bigoplus_{i \in I} b_{i\pi(i)}$$

W celu rozwiązania zadania $AZP(k)$ wystarczy dla każdego elementu a_{ij} macierzy A rozwiązać zadanie pomocnicze $ZP(a_{ij})$, a następnie spośród uzyskanych rozwiązań wybrać permutację, dla której wartość funkcji F_A jest najmniejsza.

Na mocy twierdzeń podanych w [6] możemy znacznie zmniejszyć liczbę rozwiązywanych zadań pomocniczych i po uporządkowaniu elementów macierzy:

1) zacząć rozwiązywanie zadań pomocniczych począwszy od progu b_0 takiego, że w macierzy progowej $B(b_0)$ istnieje $(n-k)$ elementów niezależnych równych progowi b_0;

   (m wyrazów macierzy nazywamy elementami niezależnymi w tej macierzy , jeśli w każdym wierszu i w każdej kolumnie istnieje co najwyżej jeden element).

2) skończyć rozwiązywanie zadań pomocniczych, jeśli dla pewnego progu b uzyskane rozwiązanie π ma tę własność, że dla każdego $i = 1,\ldots, n$ elementy wyznaczone przez permutację π w macierzy progowej $B(b)$, $b_{i\pi(i)}$, równe są progowi b .

W przeprowadzonych testach dla serii zadań, w których macierze kosztów generowane były losowo, dzięki kryteriom 1) i 2) wystarczyło rozwiązać jedynie 10 ÷ 20% wszystkich zadań pomocniczych.

5.3. Algorytm asynchroniczny

Algorytm dla *AZP* (k) składa się z następujących kroków:

Algorytm *AZP* (k)

KROK 1

Ustawienie wszystkich (różnych) wyrazów macierzy A w ciąg rosnący:
$b_1 < b_2 < \ldots < b_l$;

KROK 2

Odszukanie takiego pierwszego progu b_{i0}, dla którego macierz progowa $B(b_{i0})$ zawiera $(n-k)$ elementów niezależnych równych progowi b_{i0}; znalezienie odpowiadającego $(n-k)$ przydziału częściowego;

KROK 3

$i := i_0 - 1$; $F_A := \infty$; *finis* := **false** ;

repeat

$i := i + 1$;

KROK 4

Skonstruowanie macierzy progowej $B(b_i)$;

KROK 5

Rozwiązanie zadania $ZP(b_i)$; $\pi^{(i)}$ niech będzie uzyskanym rozwiązaniem;

KROK 6

Obliczenie $x := F_A(\pi^{(i)})$;

jeśli $x < F_A$ to $F_A := x$; zapamiętanie tego lepszego rozwiązania;

KROK 7

Sprawdzenie, czy wyrazy w macierzy $B(b_i)$ wyznaczone przez permutację $\pi^{(i)}$ są równe b_i, jeśli tak, to *finis* := **true** ;

until *finis* = **true** .

END Algorytm *AZP* (*k*)

Każde z zadań $ZP(b)$ jest liniowym algebraicznym zadaniem przydziału. Można dla niego zastosować algorytm o złożoności

$O(n^3)$ ([2], [3]). W naszym zadaniu użyteczny będzie algorytm Friezego, który startując z pewnego przydziału częściowego złożonego z k_1 elementów wykonuje $(n - k_1)$ iteracji, w każdej z nich powiększając przydział o 1.

U nas, po wykonaniu wcześniej kroku 2, możemy rozpoczynać obliczenia w zadaniach pomocniczych $ZP(\,)$ z $(n - k)$ -przydziału częściowego i wobec tego złożoność kroku 5 wynosi $O(kn^2)$.

Wykonanie kroku 2 polega na rozwiązaniu zadania przepływu w grafie dwudzielnym przy nakładzie obliczeń $O(n^3)$, zaś krok 1 ma złożoność $O(n^2 \log n)$. Złożoność całego algorytmu wynosi zatem $O(kn^4)$.

Kroki 1 i 2 algorytmu muszą być wykonane sekwencyjnie przed krokiem 3, natomiast kroki 4, 5, 7 są niezależne i mogą być wykonane równolegle. Pętla 3 wykonywana jest aż do chwili spełnienia warunku w kroku 7.

Załóżmy, że mamy do dyspozycji maszynę równoległą typu *MIMD* ([7], [8], rozdział 2 w tym tomie) z $(m + 1)$ jednakowymi, asynchronicznie pracującymi procesorami $P_0, P_1, \ldots, P_m$, tworzącymi następującą strukturę hierarchiczną: procesory $P_1, \ldots, P_m$ połączone są z procesorem nadzorującym P_0, który komunikuje się ze światem zewnętrznym oraz przydziela zadania podległym sobie procesorom P_i (w rozdziale 2 tego tomu taką strukturę nazywa się gwiazdą). Zakładamy, że procesory P_i, $i = 1, \ldots, m$, komunikują się ze sobą tylko za pośrednictwem procesora P_0. Ponadto przyjmujemy, że P_0 może w dowolnym momencie przerwać obliczenia w podległych sobie procesorach.

Dla takiej maszyny równoległej algorytm wygląda następująco:

Algorytm *AZP (k) równoległy*

(ITERACJA ZEROWA)

najpierw P_0 wykonuje sekwencyjnie kroki 1 i 2;

(1-SZA ITERACJA ALGORYTMU)

P_0 rozsyła do każdego z procesorów P_i wartość progu b_j, $j = 1, \ldots, m$, oraz otrzymany w kroku 2 startowy przydział częściowy;
Procesory P_j wykonują kroki 4, 5, 7 dla odpowiedniej wartości progowej i po zakończeniu kroku 7 przesyłają do P_0 uzyskane rozwiązanie (permutację) i wartość zmiennej logicznej *finis* ;

(*l*-TA ITERACJA ALGORYTMU)

Jeśli od danego procesora P_j dotrze do P_0 informacja o zakończeniu rozwiązywania zadania pomocniczego dla danej wartości progowej, wówczas P_0 wysyła do P_j wartość kolejnego progu ;
(równocześnie z biegnącymi procesami w procesorach $P_1, \ldots, P_m$) P_0 oblicza wartość funkcji celu dla otrzymanej permutacji, porównuje ją z dotychczasową wartością i zapamiętuje mniejszą z nich.

END Algorytm *AZP (k) równoległy*

Gdy w którymś z procesorów zadziała kryterium stopu z kroku 7 dla pewnego progu b_j, wówczas P_0 poleca przerwanie obliczeń we wszystkich procesorach, które rozwiązują zadania pomocnicze z progiem większym od b_j.

Dla tak skonstruowanego algorytmu asynchronicznego teoretyczne przyśpieszenie równe jest m .

5.4. Symulacja przebiegu czasowego algorytmu

W powyższym algorytmie można przyjąć rozmaite strategie wybierania kolejnego progu, który przesyłany jest do procesorów w kolejnych iteracjach algorytmu. Zadania pomocnicze rozwiązywane w kolejnych iteracjach algorytmu mają wprawdzie taką samą złożoność obliczeniową - $O(kn^2)$, jednak w praktyce czas trwania obliczeń dla różnych wartości progowych może się znacznie różnić. W serii testowanych zadań losowych czas obliczeń dla dużych progów był często kilka razy dłuższy niż dla początkowych wartości progowych. Zastosowanie różnych strategii wyboru kolejnego przesyłanego progu zostało przetestowane przy pomocy symulatora przebiegu czasowego algorytmu ([4]). Symulator ten oblicza czas, jaki spędza dany proces w j- tym procesorze i uzyskane przyśpieszenie. Czas komunikacji jest pomijany. Za czas pracy całego algorytmu równoległego dla $(m+1)$ procesorów uznajemy sumę czasu pracy najdłużej pracującego procesora oraz czasu pracy procesora P_0 wykonującego sekwencyjny fragment programu. Przyśpieszenie określamy jako stosunek czasu pracy algorytmu dla jednego procesora do czasu pracy algorytmu wykorzystującego $m+1$ procesorów,

Tabela 5.1 obrazuje uzyskane przyśpieszenia w przypadku dwóch strategii:

Strategia I: do procesora P_i wysyłane są w kolejnym cyklu l algorytmu progi b_{i+l*m}.

Strategia II: dla l parzystego procesor P_i otrzymuje próg $b_{l*m-(i-1)}$, zaś dla l nieparzystego - próg $b_{(l-1)*m+i}$.

Stosując strategię drugą spodziewamy się bardziej równomiernego obciążenia procesorów.

W serii zadań testowych dla macierzy kosztów o wyrazach generowanych losowo z rozkładem równomiernym korzystniejsza okazała się strategia II.

Tabela 5.1.

Liczba proces.	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Przysp. dla I strat.	1,2	1,9	2,7	3,7	4,6	5,0	5,7	6,4	6,8	7,2	7,8	8,1	8,4	9,1
Przysp. dla II strat.	1,2	2,1	3,0	4,0	4,8	5,2	5,9	6,7	7,1	7,5	8,2	8,5	8,7	9,4

5.5. Implementacja algorytmu na sieci transputerowej

Algorytm został zaimplementowany na realnej maszynie wieloprocesorowej: sieci transputerów.

W komputerze IBM PC AT zostały zainstalowane dwie karty transputerowe: IMS T414 (dalej oznaczany w skrócie przez T4) oraz IMS T800 (oznaczany T8). Transputer T4 połączony jest z AT, pełni zatem rolę korzenia w tej dwutransputerowej sieci.

Procesor T4 ma zegar 20 MHz, T8 - zegar 30 MHz. T8 wyposażony jest w jednostkę do operacji zmiennoprzecinkowych. Oba procesory posiadają własną lokalną szybką pamięć (on-chip memory) : T4 ma 2Kb, T8 - 4Kb. Dane o obu procesorach, czasy wykonywania operacji zmiennoprzecinkowych, wyniki testów szybkości działania procesorów i przepływu informacji między transputerami przy pomocy łączy można znaleźć w [9] (patrz także rozdział 2.8 tego tomu).

Z punktu widzenia projektowania algorytmu i przydzielania zadań procesorom istotne są różnice w czasie obliczeń tych samych procedur na obu transputerach. W zadaniu $AZP(k)$ występują dwojakiego rodzaju procedury:

(a) wykonujące tylko operacje na liczbach całkowitych,

(b) procedury, w których jest również pewna ilość operacji zmiennoprzecinkowych.

W serii zadań testujących oba typy procedur transputer T4 okazał się średnio: 1,5 raza wolniejszy od T8 dla testowanych procedur typu (a) i 3 razy wolniejszy dla procedur typu (b).

Istnieje jeszcze jedna cecha szczególna tej sieci: program skonfigurowany na dwa transputery musi zawierać systemowe zadanie filter (task filter, cf. [10], [11]), które, jak wykazują testy, dwukrotnie spowalnia prowadzone na transputerze obliczenia.

W przypadku tej konkretnej instalacji, złożonej z niejednakowych procesorów, konieczne jest przedefiniowanie pojęcia przyspieszenia algorytmu równoległego. Otóż za przyspieszenie algorytmu będziemy uważać liczbę:

$$s = \frac{t_1}{t_2},$$

gdzie t_1 jest czasem pracy algorytmu na jednym, szybszym procesorze T8, t_2 - czasem pracy na sieci dwuprocesorowej złożonej z T4 i T8.

Algorytm dla $AZP(k)$ dla rozpatrywanego zestawu transputerów testowany był dla różnych strategii przydzielania zadań transputerom. Dla tej konkretnej konfiguracji sprzętowej należało zastosować inną strategię rozdziału zadań niż ta zaprezentowana w poprzednim paragrafie dla symulatora, gdyż mając do dyspozycji

dwa procesory nie można jednego z nich "poświęcić" przydzielając mu rolę nadzorcy.

Istotna jest również dbałość o to, by procesy nie czekały wzajemnie na wymianę informacji.

Z testowanych kilku wersji rozdziału zadań najefektywniejsza okazała się następująca:

1) Procesor T4 (a dokładniej : proces wykonujący się na procesorze T4) przesyła do T8 wejściową macierz kosztów oraz wczytaną wartość k. Proces na T8 sortuje elementy macierzy i wyznacza pierwszy próg, począwszy od którego rozwiązywane będą zadania pomocnicze, tzn. wykonuje sekwencyjnie kroki 1. i 2. algorytmu (testy wykazały, że przeprowadzenie sortowania w T4 jest droższe czasowo niż przesłanie macierzy do T8, wykonanie tam sortowania i przesłanie posortowanego ciągu liczb do T4).
2) Procesor T8 przesyła do T4 posortowane elementy, wartość pierwszego progu, wartość startowego przydziału częściowego.
3) W T4 wykonywane są zadania pomocnicze dla kilku (l) pierwszych progów (l jest z góry ustalone przed rozpoczęciem pracy algorytmu), zaś procesor T8 wykonuje zadania pomocnicze dla progów o numerach $l+1$, $l+2$,... .

Przy tej strategii kryterium stopu dla zadań o dużych rozmiarach jest spełnione na ogół dla któregoś z zadań rozwiązywanych w procesorze T8.

Po spełnieniu kryterium stopu procesor T8 przesyła swoją najlepszą wartość funkcji celu wraz z odpowiadającą jej permutacją. W T4 porównuje się rozwiązania uzyskane w obu procesorach i wybiera lepsze.

Wybranie właściwej liczby zadań powierzonych do rozwiązania procesorowi T4 jest istotne z punktu widzenia uzyskanego przyśpieszenia w algorytmie. Jeśli l będzie zbyt duże, to proces w procesorze T8 zakończy się wcześniej niż proces w T4, T8 będzie

bezczynnie czekał na odbiór informacji. Ten czas oczekiwania wlicza się do czasu pracy całego algorytmu. Wartość l zbyt mała spowoduje z kolei bezczynne oczekiwanie procesu w procesorze T4.

Z przeprowadzonych obliczeń dla serii zadań testowych, w których wyrazy macierzy kosztów były generowane z rozkładem równomiernym, wynika, że l powinno być rzędu $1/4 \div 1/3$ liczby wszystkich zadań pomocniczych (dużych iteracji) rozwiązywanych przez algorytm. Przed zakończeniem pracy algorytmu nie wiemy jednak, ile iteracji on wykona. W testowanych przykładach jest to około $10 \div 20\%$ wszystkich możliwych zadań, tzn. wszystkich różnych elementów w macierzy wejściowej. W serii zadań testowych uzyskiwane przyśpiszenie s (liczone tak, jak to powyżej zdefiniowano) wynosiło średnio 1,2.

5.6. Uwagi końcowe

Zaprezentowany algorytm asynchroniczny dla ogólnie sformułowanego zadania przydziału pracuje na maszynach typu MIMD o niewielkiej liczbie procesorów. Przyśpieszenie jest proporcjonalne do liczby użytych procesorów. Zostało to potwierdzone podczas symulacji algorytmu dla maszyny równoległej, w której procesory tworzą gwiazdę, choć uzyskane tą drogą przyśpieszenie jest gorsze od teoretycznego.

Obliczenia przeprowadzone na realnej maszynie złożonej z dwóch niejednakowych procesorów wymagały interakcyjnego przydziału zadań do procesorów w celu uzyskania jak największego przyśpieszenia.

Literatura

1. Bertsekas D.P.(1992): "Auction Algorithms for Network flow problems: a Tutorial Introduction." *Computational Optimization and Applications* 1, 7 - 66.
2. Burkard R. E., Zimmermann U. (1979): "The Solution of Algebraic Assignment and Transportation Problems". W: Optimization and Operations Research, red. R. Henn, B. Korte, W.Oetli, Lecture Notess in Econ. and Math. Systems, 157, Springer Verlag, Berlin, 55 - 65.
3. Frieze A. M. (1979): "An Algorithm for Algebraic Assignment problems". *Discrete Appl. Math.* 1, 235 - 259.
4. Grygiel G. (1989): "Równoległość na poziomie podprocedur na przykładzie zadania przydziału". *Prace IBS PAN*, ZPM - 13, Warszawa.
5. Grygiel G. (1989): "Program AZP rozwiązujący algebraiczne zagadnienie przydziału - opis funkcjonalny". *Prace IBS PAN* ZPM - 18, Warszawa.
6. Grygiel G. (1991): "Assignment Problems in Ordered Structures". *Prace IBS PAN*, ZPM - 20, Warszawa.
7. Quinn M.J. (1988): Designing efficient algorithms for parallel computers. McGraw-Hill, New York.
8. Schwartz J. T. (1980): "Ultracomputers". *ACM Trans.* 2, 486 - 512.
9. (1987): Transputer reference manual. INMOS Ltd, Prentice Hall, New York.
10. (1988): Parallel Fortran User Guide. 3L Ltd, Edinburgh.
11. (1991): Parallel C User Guide. 3L Ltd, Edinburgh.

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.