ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Tadeusz DYDUCH, Ewa DUDEK-DYDUCH
Politechnika Krakowska, Akademia Górniczo-Hutnicza

# RÓWNOLEGŁY ALGORYTM BRANCH & BOUND DO ROZWIĄZYWANIA PROBLEMU KOMIWOJAŻERA

Streszczenie: Artykuł dotyczy paralelizacji algorytmu branch & bound zastosowanego do rozwiązywania zadania Travelling Salesman Problem. Przedstawiono analizę teoretyczną wykazującą, że dzięki równomierności przeszukiwania drzewa decyzyjnego zaproponowany algorytm równoległy powoduje /statystycznie/ wzrost szybkości obliczeń większy, niż wynika z sumarycznej mocy obliczeniowej użytych procesorów. Omówiono realizację algorytmu w sieci komputerowej z zastosowaniem pakietu Parallel Virtual Machine 3.2.

# PARALLEL BRANCH & BOUND ALGORITHM FOR TRAVELLING SALESMAN PROBLEM

Summary: The paper deals with parallelisation of back-track algorithm, one of the best known algorithms for searching tree graphs. The algorithm is a part of branch & bound method, being applied to solving Travelling Salesman Problem. It is shown by theoretical analysis that parallelisation makes the algorithm more efficient /in statistical meaning/. The reason for that is a more adequate search order. The paper describes the use of Parallel Virtual Machine software system for implementationof the algorithm in computer network.

# L'ALGORITHME PARALLÈLE POUR RÉSOUDRE LE PROBLÈME DE TRAVELLING SALESMAN PROBLEM

Résumé: L'article porte sur la parallélisation d'un algorithme back -track, l'un des algorithmes fondamentaux de recherche des graphes à branches, utilisé pour résoudre le problème Travelling Salesman Problem à l'aide d'une méthode branch&bound. L'analyse théorique présentée démontre que grâce à une recherche systématique d'un graphe, l'algorithme parallele proposé est, statistiquement, plus efficace que cela n'en résulte de la puissance totale des processeurs utilisés. Enfin, on a examiné l'exécution de l'algorithme appliquant le système de programmation Parallel Virtual Machine 3.2.

## 1. Wprowadzenie

Kolejnym etapem rozwoju systemów komputerowych są układy wieloprocesorowe. Dzielą się one na układy ze wspólną pamięcią (shared-memory) oraz z pamięcią dzieloną (distributed-memory). Układy o wspólnej pamięci wymagają rozwiązania konfliktu

jednoczesnego dostępu kilku procesów (wątków) do jednej komórki (bloku) pamięci. Jest to rozwiązywane m.in. za pomocą specjalnych instrukcji maszynowych, takich jak Test&Set(X,R) lub Swap(X,R), umożliwiających regulację dostępu z wykorzystaniem pojedynczych semaforów [7], [9]. Programy dla takich komputerów wymagają uwzględnienia zarówno indywidualnie implementowanych instrukcji, o których wyżej wspomniano, jak i architektury komputera: podziału pamięci, sposobu jej komunikacji z procesorami. Szereg prac m.in. J. Hernandeza i in. [6] pokazuje, że tego typu architektura szczególnie nadaje się do rozwiązywania problemów wymagających operacji na macierzach dużych rozmiarów.

Układy o pamięci dzielonej są łatwiejsze do unifikacji (sformułowania jednolitego podejścia), umożliwiając opracowanie procedur wykonywanych zarówno na pojedynczym wieloprocesorowym komputerze, jak i w systemie rozproszonym, na wielu komputerach jedno- i wieloprocesorowych połączonych siecią. Wraz z upowszechnieniem się sieci komputerowych, głównie za sprawą systemu operacyjnego UNIX i jego odmian, podjęto prace nad realizacją procesów obliczeniowych w sieciach. Od 1989r. w Oak Ridge National Laboratory oraz w University of Tennessee rozwijany jest projekt PVM (Parallel Virtual Machine) [5]. PVM (aktualna wersja 3.2) jest oprogramowaniem, umożliwiającym wykorzystywanie heterogenicznej (niejednorodnej) sieci komputerów UNIX'owych jak jednego dużego wieloprocesorowego komputera o dzielonej pamięci.

Projekty, takie jak PVM dostarczają doświadczeń i rozwiązań umożliwiających standaryzację oprogramowania wykorzystującego systemy rozproszone oraz rozwiązań architektury komputerów wieloprocesorowych. Dlatego też cieszą się zainteresowaniem i wsparciem wielkich firm komputerowych.

Niniejsza praca dotyczy zrównoleglenia algorytmu branch & bound (B&B) dla problemu komiwojażera. Przedstawia analizę realizacji tego algorytmu na wirtualnym komputerze równoległym, zrealizowanym za pomocą pakietu PVM.

## 2. Szeregowa procedura B&B dla zadania komiwojażera

Jak wiadomo, problem komiwojażera polega na znalezieniu minimalnego cyklu Hamiltona w grafie G, którego łukom przypisane są dodatnie współczynniki wagi. W najbardziej popularnej interpretacji wierzchołki grafu reprezentują miasta, zaś łuki i przyporządkowane im współczynniki odpowiadają połączeniom między miastami i odległościom między nimi. Problem ten może być również interpretowany jako problem szeregowania zadań na jednej maszynie przy uwzględnieniu czasów przezbrojeń.

W pracy [8] przedstawiono algorytm podziału i ograniczeń dla tego problemu. Jest to algorytm typu konstrukcyjnego [2],[3]. Algorytm wykorzystuje macierzową prezentację grafu

połączeń, a więc współczynnik $a_{i,j}$ macierzy A odpowiada wadze łuku (i,j) w grafie G (określa odległość między miastem i-tym oraz j-tym ).

Zbiór rozwiązań związanych z każdym wierzchołkiem drzewa decyzyjnego tego algorytmu wyznaczony jest przez podzbiór łuków, które na pewno wystąpią w generowanych rozwiązaniach, oraz podzbiór łuków, które na pewno w nich nie wystąpią. Reguła rozgałęziania polega na podziale na dwa podzbiory (drzewo binarne), przy czym podział jest wyznaczony przez wybór łuku różnicującego te podzbiory (a więc takiego, który będzie dołączony do jednego z nowych podzbiorów i który z pewnością nie będzie należał do drugiego). Podział dokonywany jest w taki sposób, aby zapewnić maksymalny wzrost dolnego ograniczenia. Początkowe dolne ograniczenie konstruuje się w wyniku analizy macierzy A, która następnie jest przekształcana (redukowana). Dolne ograniczenia dla kolejnych wierzchołków drzewa decyzyjnego konstruuje się w wyniku analizy zredukowanych macierzy związanych z tymi wierzchołkami.

Biorąc pod uwagę regułę wyboru wierzchołka do rozgałęzienia wyróżniamy dwie wersje tego algorytmu:

a) wybieramy ten wierzchołek, dla którego dolne ograniczenie przyjmuje wartość minimalną,

b) konstruujemy drzewo decyzyjne w oparciu o strategię back - track (B - T).

Przytoczymy schemat według [8] rekurencyjnej procedury realizującej ten algorytm w wersji b).

niech: REDUCE(A) - procedura redukcji (przekształcenia) macierzy danych A i wyznaczania przyrostu dolnego ograniczenia,

UNREDUCE(A) - procedura przywrócenia poprzednich wartości A,

BESTEDGE(A, r, c, most) - procedura wyboru najlepszego łuku (r,c) rozgałęziającego,

most - wzrost dolnego ograniczenia spowodowany odrzuceniem łuku (r,c),

cost - aktualny koszt częściowy rodziny rozwiązań,

edges - liczba włączonych (dodanych) łuków,

tweight - wartość najlepszego aktualnie rozwiązania (ograniczenie górne),

```
procedure EXPLORE(edges, cost, A);
begin
    cost ← cost + REDUCE(A);
    if cost < tweight then
      if edges = n - 2 then begin
          dołącz dwa końcowe łuki;
          tweight ← cost;
          zapamiętaj to rozwiązanie (ciąg łuków);
      end
```

```
    else begin
        BESTEDGE(A, r, c, most);
        lowerbound ← cost + most;
        eliminuj ewentualny cykl;
        newA ← A - (wiersz r) - (kolumna c);
        EXPLORE(edges + 1, cost, newA);
        A ← newA + (wiersz r) + (kolumna c);
        if lowerbound < tweight then begin
            a_{r,c} ← ∞;
            EXPLORE(edges, cost, A);
            a_{r,c} ← 0;
        end
    end;
  UNREDUCE(A);
end.
```

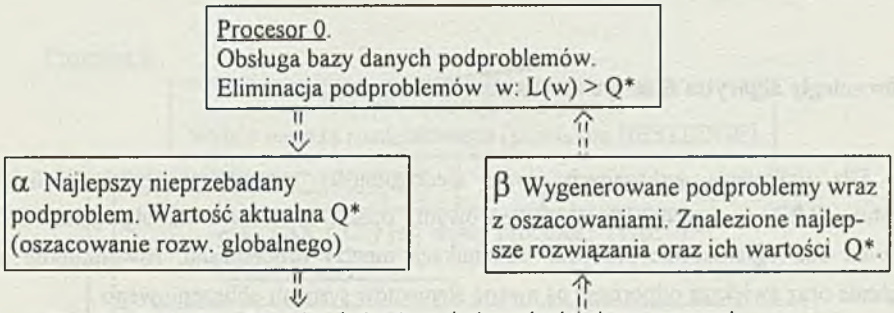
Algorytm back - track, zwany także przeszukiwaniem w głąb, gwarantuje uporządkowane przeglądanie drzewa rozwiązań, bez potrzeby odrębnego ewidencjonowania przebytych łuków i zapisu rozwiązań częściowych. Realizuje to autonomicznie mechanizm kolejnych zagłębień procedur rekurencyjnych, wbudowany w kompilator użytego języka programowania. Wadą stosowania tego algorytmu jest to, że narzuca on określony porządek przeszukiwania drzewa (usystematyzowany). W miejsce strategii back - track można zastosować odpowiednią regułę wyboru wierzchołka do rozgałęzień, np. wg wartości dolnego ograniczenia, co poprawia efektywność algorytmu B&B. Wymaga to jednak swobody przenoszenia się, w trakcie analizy drzewa rozwiązań, między różnymi poziomami i gałęziami drzewa, i stwarza konieczność ewidencjonowania rozwiązań częściowych.

Efektywność procesu B&B zależy od wielkości poddrzew wyeliminowanych z przeszukiwania dzięki odpowiedniemu oszacowaniu ich zbiorów rozwiązań. Zależy to od dwu elementów: precyzji oszacowania podzbioru rozwiązań oraz aktualnie znanego najlepszego rozwiązania. Zatem szybkie natrafienie na dobre, bliskie optymalnemu rozwiązanie poważnie przyspiesza proces obliczeniowy. W tym celu stosuje się algorytmy heurystyczne typu greedy (zachłanne) lub przegląd B-T poddrzewa pełnego drzewa poszukiwań, utworzonego przez arbitralną eliminację części łuków w grafie początkowym [4].

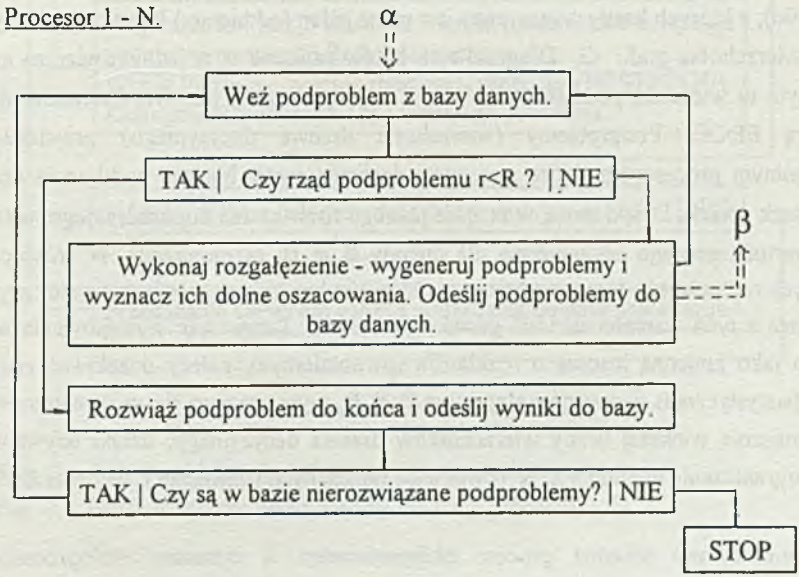
W dalszym tekście będziemy używać zamiennie określeń "zbiór rozwiązań związanych z wierzchołkiem drzewa decyzyjnego" oraz "podproblem".

### 3. Zrównoleglenie algorytmu B&B poprzez wielodostępną bazę rozwiązań częściowych

Oczywistą słabością podanego poniżej schematu jest groźba nadmiernego rozrośnięcia się bazy danych w trakcie obliczeń. Jest ona mniej istotna przy obecnym rozwoju technologii pamięci masowych i dostępu do nich. Poważniejszą groźbę stanowi możliwość zdominowania łącznego czasu obliczeń przez procedury komunikacji z bazą. Będzie tak wówczas, gdy podstawowy cykl - jednorazowe rozgałęzienie - jest zbyt prosty w realizacji w stosunku do procedur komunikacji (przesyłania informacji o podproblemach).




Rys.1 a. Przepływ danych między bazą i równoległymi procesorami
Fig. 1 a. Data flow between data base and parallel processors




Rys. 1 b. Schemat kolejności działań realizowany przez procesory równoległe
Fig. 1 b. Flow chart

Realizacja algorytmu B&B w wersji a) w systemie wieloprocesorowym może być oparta na powyższych schematach (rys.1), w których:

R - ustalony parametr; N - liczba procesorów; Q* - wartość kryterium dla najlepszego znanego aktualnie rozwiązania; L(w) - dolne ograniczenie podproblemu w.

Ten ogólny, uniwersalny schemat wymaga zorganizowania wielodostępnej bazy danych - rozwiązań częściowych (podproblemów). Można tu wykorzystać mechanizmy przestrzeni krotek, występujące w języku LINDA [1].

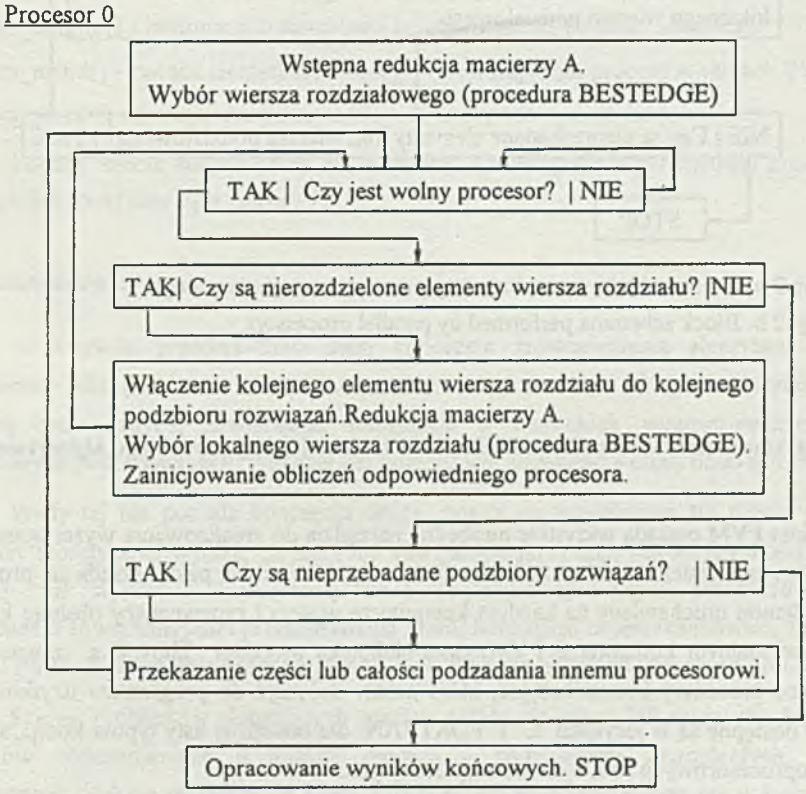
## 4. Równoległy algorytm B & B z mechanizmem back-track

Dla uniknięcia wskazanych wyżej niedogodności autorzy proponują realizację algorytmu B&B w systemie wieloprocesorowym, opartą częściowo na algorytmie B-T. Zapewnia ona ograniczenie procedur komunikacji między procesorami, równomierne ich obciążenie oraz zwiększa odporność na awarie elementów systemu obliczeniowego.

Modyfikacja polega na zmianie reguły podziału na jednym lub dwu najwyższych piętrach drzewa decyzyjnego. Zbiór wyjściowy dzielimy na n-1 podzbiorów (zamiast na dwa, jak poprzednio), z których każdy wyznaczany jest przez jeden (odmienny) łuk, wychodzący z tego samego wierzchołka grafu G. Długości tych łuków zapisane są w jednym wierszu macierzy A zwanym tu wierszem podziałowym. Wiersz ten wyznaczany jest przez element obliczany procedurą EDGE. Podproblemy (wierzchołki drzewa decyzyjnego) przydzielane są poszczególnym procesorom, które generują i analizują swoje binarne poddrzewa decyzyjne metodą back - track. Dzięki temu, w miejsce jednego rozwiązania dopuszczalnego, ustanawiającego wartość górnego ograniczenia dla metody B & B, otrzymujemy N równocześnie obliczonych rozwiązań dopuszczalnych, z których każde ma swoją wartość kryterium. Najmniejsza z tych wartości stanowi górne ograniczenie. Przyjmując występowanie wartości kryterium jako zmienną losową o rozkładzie równomiernym, należy oczekiwać znacznego wzrostu (statystycznie) sprawności algorytmu B & B, wyrażającego się w wyeliminowaniu z analizy znacznie większej liczby wierzchołków drzewa decyzyjnego, dzięki używaniu jako górnego ograniczenia minimum z N równolegle obliczanych rozwiązań i wartości kryterium.

Proponowany schemat procesu obliczeniowego w systemie wieloprocesorowym przedstawiono na rys.2.

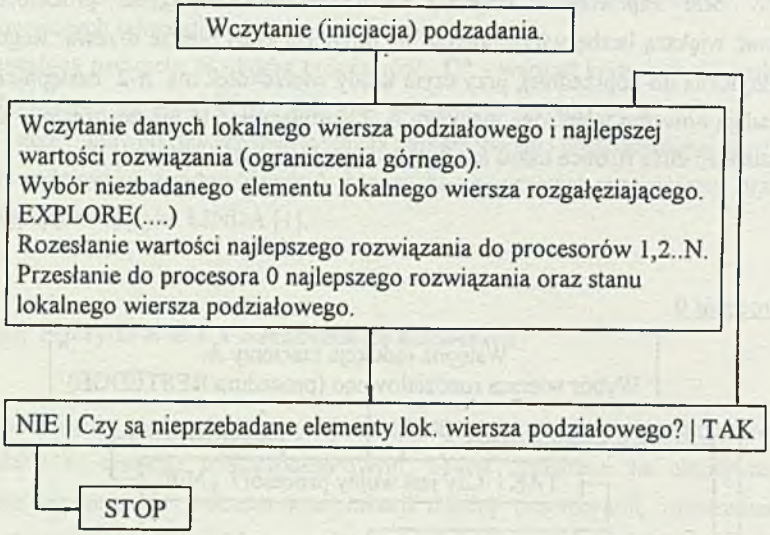
W celu zapewnienia większej równomierności obciążenia procesorów można generować większą liczbę wierzchołków na drugim z kolei piętrze drzewa. Reguła podziału jest analogiczna do poprzedniej, przy czym każdy wierzchołek ma n-2 następników, zaś łuki odpowiadają nowemu wierszowi macierzy A. Zabezpieczenie to ma znaczenie w sytuacji, gdy mogą zaistnieć duże różnice czasu analizy poszczególnych poddrzew.




Rys. 2 a. Schemat blokowy realizowany przez procesor główny

Fig. 2 a. Block schemma performed by the main processor

Procesor 1. 2 ..N




Rys.2 b. Schemat blokowy realizowany przez procesory równoległe

Fig. 2 b. Block schemma performed by parallel processors

## 5. Wykorzystanie procedur pakietu PVM 3.2 w realizacji proponowanego algorytmu

Pakiet PVM posiada wszystkie niezbędne narzędzia do zrealizowania wyżej opisanych algorytmów i zapewnienia niezbędnej komunikacji między nimi. Na pakiet składa się program 'pvmd3' (demon uruchamiany na każdym komputerze w sieci i zapewniający obsługę komunikacji w wirtualnym komputerze PVM) oraz biblioteka procedur libpvm3.a zawierająca poszczególne procedury komunikacyjne, które należy dołączyć do programów użytkowych. Procedury dostępne są w językach C i FORTRAN dla obszernej listy typów komputerów, także wieloprocesorowych i ich systemów operacyjnych.

Procedury w języku C niezbędne do realizacji podanego algorytmu są następujące:

- pvm_spawn(program,nproc,tids) - uruchamia procesy o treści 'program' w liczbie nproc; wektor tids zawiera identyfikatory uruchomionych procesów lub komunikaty o błędach,
- pvm_exit() - odłącza dany proces od PVM.

Procedury przesyłania danych zarówno po stronie nadawania,jak i odbierania zawierają etap pakowania i odpowiednio rozpakowywania danych oraz mogą być realizowane

synchronicznie: proces oczekuje na pozytywny wynik działania procedury, lub asynchronicznie.

- pvm_initsend() - inicjacja bufora wysyłkowego.
- pvm_pk*() - pakowanie danych typu * (int, float, ....) do bufora.
- pvm_send() - wysłanie zawartości bufora do konkretnego adresata.
- pvm_mcast() - wysłanie zawartości bufora do wielu adresatów.
- pvm_recv() - odebranie paczki danych z oczekiwaniem na jej nadejście (synchronicznie).
- pvm_nrecv() - próba odebrania paczki danych, o ile nadeszła (asynchronicznie).
- pvm_upk*() - rozpakowanie danych typu * z bufora.
- pvm_bufinfo() - informacja o zawartości bufora (długość, etykieta, nadawca).
- pvm_mytid() - zwraca identyfikator danego - wywołującego procesu w ramach PVM.
- pvm_parent() - zwraca identyfikator procesu, który zainicjował dany.

Podany wykaz nie jest pełny ani dokładny: nie uwzględnia parametrów procedur ani ich wartości. Dokładny opis zawiera [5].

## 6. Podsumowanie

W artykule przedstawiono dwie koncepcje zrównoleglenia algorytmu podziału i ograniczeń dla problemu komiwojażera. Pierwsza koncepcja polega na wykorzystaniu wpólnej bazy danych, zawierającej informacje o wszystkich wygenerowanych i nie odrzuconych podproblemach. Wadą jej jest konieczność przesyłania dużej ilości informacji.

Wady tej nie posiada koncepcja druga, oparta na zmieniającej sie regule podziału. Z analizy teoretycznej wynika, że algorytm równoległy powoduje wzrost szybkości obliczeń większy, niż wynika z sumarycznej mocy obliczeniowej użytych procesorów, dzięki zapewnieniu równomierności przeszukiwania grafu. Koncepcja ta jest realizowana za pomocą pakietu PVM. Przeniesienie programu na np. iPSC/860 nie nastręcza większych problemów.

Szereg problemów dotyczących grafów należy do klasy NP-trudnych. Stosowanie systemów obliczeniowych o dużych mocach - komputerów równoległych pozwala rozwiązywać większe problemy tej klasy. Istotnym, zupełnie odmiennym niż w komputerach szeregowych elementem realizacji dużych, długotrwałych obliczeń równoległych jest ich odporność na awarie i przypadkowe zakłócenia. Algorytmy realizowane w systemach rozproszonych pozwalają na dublowanie części obliczeń bez konieczności powtarzania całości, umożliwiają także elastyczne obciążanie poszczególnych węzłów.

Badania nad przetwarzaniem danych w komputerowych strukturach równoległych, tj. komputerach wieloprocesorowych i sieciach komputerowych tworzących systemy rozproszone,

dotyczą możliwości rozdziału zadań między procesory/komputery widzianych łącznie z możliwościami komunikowania się procesorów/procesów między sobą oraz możliwościami kontrolowania procesu obliczeniowego jako całości. Pakiet PVM stanowi w tych badaniach interesujące narzędzie zarówno do obliczeń, jak i przygotowania programów użytkowych na komputery równoległe.

LITERATURA

[1] Ben-Ari M.: Principles of concurrent and distributed programming. Englewood Cliffs, NY, Prentice-Hall 1990.

[2] Dudek-Dyduch E.: Control of discrete event processes - branch and bound method. Prepr. of IFAC/Ifors/Imacs Symposium Large Scale Systems: Theory and Applications, Chinese Association of Automation, vol. 2, 1992, 573 - 578.

[3] Dudek-Dyduch E.: Optimization of discrete manufacturing processes - branch and bound method. in P. Gritzman, R. Hettich, R. Horst, E. Sachs (eds) Operations Research '92. Springer-Verlag, Heidelberg 1992, 19 - 22.

[4] Dyduch T.: O zastosowaniu teorii gier pozycyjnych do syntezy układów sterujących. Praca doktorska AGH, Kraków 1981.

[5] Geist Al i in.: Oak Ridge National Laboratory Report TM - 12187, 1993.

[6] Hernandez V., Vidal A.M.: Parallel orthogonal triangularization of a rectangular matrix on a distributed memory multiprocessor. Working Conf. on Decentralized Systems, Lyon, December 1989.

[7] Iszkowski W., Maniecki M.: Programowanie współbieżne. W-wa WNT 1982.

[8] Sysło M., Deo N., Kowalik J.: Algorytmy optymalizacji dyskretnej. W-wa PWN 1993.

[9] Weiss Z., Gruźlewski T.: Programowanie współbieżne i rozproszone. W-wa WNT 1993.

Recenzent: Prof. dr hab. inż Mirosław Zaborowski

Wpłynęło do Redakcji do 30.04.1994 r.

**Abstract**

The paper deals with parallelisation of branch & bound method, being applied to solving Travelling Salesman Problem. Two methods are analysed. The first one is based on multi-user data base of subproblems. The second utilizes back-track algorithm, one of the best known algorithms for searching tree graphs. It is shown by theoretical analysis that parallelisation makes the back-track algorithm more efficient /in statistical meaning/. The reason for that is a more adequate search order. The paper describes the use of Parallel Virtual Machine software system for implementation the algorithm in computer network.