AUTOMATYKA • 2005 • Tom 9 • Zeszyt 1–2

Marcin Tusiewicz*

Rozproszony przegląd zupełny z obcięciami w drzewie poszukiwań dla problemów *NP*-trudnych

1. Wstęp

Liczną grupę zadań optymalizacyjnych zajmują problemy zaklasyfikowane jako problemy *NP*-trudne. Istnieje wiele różnych niedeterministycznych metod poszukiwań na zbiorze rozwiązań dopuszczalnych, które pozwalają znaleźć element o wartości funkcji celu uznawanej za „wystarczająco bliską" optimum. Niejednokrotnie ze względu na postawione wymagania lub uwarunkowanie problemu jesteśmy zmuszeni do przejrzenia całego zadanego zbioru. Techniką pomocną w deterministycznym wykluczaniu podzbiorów, w których z pewnością optimum nie istnieje, jest metoda obcięć w drzewie poszukiwań (*branch-and-bound*) [1–3]. Ze względu na swoją naturę oraz naturalności stosowania zasady „dziel i zwyciężaj" sekwencyjne algorytmy rozwiązujące problemy *NP*-trudne zazwyczaj w bardzo prosty i stosunkowo wydajny sposób udaje się przekształcić na wersje równoległe, dokonujące obliczeń na wielu procesorach. Zadaniem trudniejszym jest uzyskanie wysokiej wydajności w rozległych środowiskach rozproszonych [13], w których przesyłanie komunikatów między poszczególnymi węzłami wymaga stosunkowo dużego nakładu czasowego. W literaturze zaproponowano wiele metod równomiernego rozproszenia obliczeń (*load balancing*) wspomnianej klasy problemów [4, 6–9, 11–12]. W niniejszym artykule wystąpi szczególne nawiązanie do [6, 8], w których przedstawione rozwiązania dotyczą rozległych środowisk rozproszonych; nastąpi konkatenacja przedstawionych metod z zaproponowanym przez autora modelem środowiska rozproszonego [5, 10] oraz charakterystyka usprawnień algorytmu możliwa dzięki zastosowaniu nowego modelu. Całość zostanie zilustrowana na przykładzie dyskretnego optymalizacyjnego problemu plecakowego.

2. Dyskretny optymalizacyjny problem plecakowy – opis

Rozważania dotyczące konstrukcji i wydajności przedstawionych algorytmów zobrazowane zostaną na podstawie poniżej przedstawionego problemu (w ang. lit.: 0–1 *Knapsack Problem*).

* EAIiE, Katedra Informatyki, Akademia Górniczo-Hutnicza w Krakowie; tusiew@agh.edu.pl

Dany jest wektor:

$$\text{OBJECTS} = \{(weight_i, value_i)\}_{i = 1..n} \quad weight_i, value_i \in \text{N}$$

oraz CAPACITY $\in$ N. Zadaniem jest znaleźć wektor P ze zbioru W rozwiązań dopuszczalnych

$$P \in W = \{(b_1, \ldots, b_n), b_i \in \{0, 1\}\}$$

spełniający założenia:

$$\sum_{i=1}^{n} weight_i * b_i \leq capacity \tag{1}$$

$$\sum_{i=1}^{n} value_i * b_i \rightarrow \max \tag{2}$$

Algorytm sekwencyjny rozwiązujący powyższy problem może mieć postać przechodzenia przez binarne drzewo decyzyjne, w którym ścieżka od korzenia do liścia utożsamiana jest z poszczególnymi elementami zbioru W, a „obcięcie" następuje w przypadku sprzeczności z założeniem (1) na dowolnym poziomie drzewa. Idea rozproszenia obliczeń polega wtedy zazwyczaj na przekazaniu pewnego poddrzewa do innego węzła obliczeniowego, a odpowiedni wybór tego poddrzewa jest kluczowym czynnikiem decydującym o wydajności.

3. Charakterystyka grafowego modelu środowiska rozproszonego

W pracy [5] autor przedstawił model środowiska rozproszonego. Elastyczność zaprezentowanego modelu pozwala w stosunkowo czytelny, a jednocześnie nieskomplikowany sposób przedstawić topologie połączeń w sieciach małych, jak i rozległych środowiskach rozproszonych składających się z bardzo dużej liczby maszyn. W rozważaniach autora szczególnie istotną rolę odgrywał drugi przypadek, który zostanie poddany bardziej wnikliwej analizie także w niniejszym artykule.

Kluczowymi elementami modelu są dwie struktury pozwalające z odpowiednio dobraną szczegółowością oddać potencjalne konfiguracje rzeczywistych środowisk:

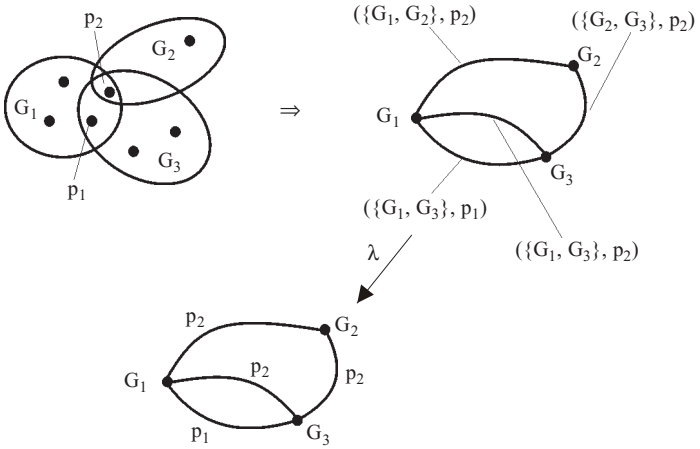
graf środowiska I poziomu – graf zwykły $S_1 = (P, E_p)$, gdzie P jest zbiorem węzłów środowiska, natomiast E_p jest zbiorem krawędzi takim, że $\{p_1, p_2\} \in E_P \Leftrightarrow \exists G \subset P \wedge p_1, p_2 \in G$, gdzie z definicji G jest zbiorem węzłów, w którym dwa dowolne elementy mogą obustronnie i bezpośrednio przesyłać między sobą komunikaty;

graf środowiska II poziomu – multigraf bez pętli $S_2 = (\Gamma, E, f)$, gdzie:

- Γ jest podzbiorem grup węzłów: $\Gamma \subset \{G_i\}_{i=1,\ldots,K}$,
- $E \subset \Gamma_2 \times P$, gdzie $\Gamma_2 \subset P_2(\Gamma)$, natomiast P jest zbiorem węzłów[1)],
- $f : E \to P_2(\Gamma)$ jest odwzorowaniem takim, że

$$f^{-1}(d) = \{(\{G_i, G_j\}, p) : \{G_i, G_j\} = d, p \in G_i \wedge p \in G_j\}.$$

Graf środowiska I poziomu ma na celu szczegółową reprezentację możliwości przepływu komunikatów między węzłami obliczeniowymi. Struktura ta jest używana zazwyczaj tylko na poziomie danej grupy G (utożsamianej z podsiecią komputerów), której dotyczy. Graf środowiska II poziomu został natomiast wprowadzony w celu zmniejszenia złożoności czasowej algorytmów ustalających możliwości przesyłania komunikatów pomiędzy poszczególnymi grupami (rys. 1). Dodatkowo wprowadzone zostało odwzorowanie λ: E $\to$ P identyfikujące platformy łączące węzły: $\lambda(\gamma, p) = p$.



Rys. 1. Przykładowa topologia grup węzłów (z lewej) oraz odpowiadający jej graf środowiska II poziomu z etykietami krawędzi (z prawej) oraz graf środowiska II poziomu naniesionymi na krawędzie wartościami odwzorowania λ (u dołu)

We wspomnianej pracy [5] przewidziano także możliwość określania parametrów modelu środowiska. Na potrzeby niniejszych rozważań zdefiniowane zostały:

pow(p $\in$ G) $\in \Re$ – moc obliczeniowa danego węzła,

pow_G(G) $\in \Re$ – moc obliczeniowa grupy liczona jako suma mocy poszczególnych węzłów.

[1)] $P_2(\Gamma)$ oznacza zbiór wszystkich 2-elementowych podzbiorów zbioru Γ. Formalnie: $P_2(\Gamma) = \{A \subset \Gamma : |A| = 2\}$, gdzie $|A|$ jest mocą zbioru.

4. Algorytm rozproszonego przeglądu zupełnego z obcięciami w drzewie poszukiwań

Proponowany algorytm ma charakter przeglądu całego zbioru rozwiązań dopuszczalnych poprzez jego podział na rozłączne podzbiory i uruchomienie procedury poszukiwań elementu optymalnego na tych podzbiorach, a następnie na wyborze wśród uzyskanych wyników jednego, najlepszego elementu (być może spośród wielu jednakowo dobrych). Podczas przeglądu procedura obcięć wyklucza podzbiory nie spełniające warunku (1). Konstruowane binarne drzewo decyzyjne jest wynikiem bijektywnego przekształcenia zbioru W – każdy jego element jest tożsamy z dokładnie jedną ścieżką od korzenia do liścia. Rozproszenie polega na wyselekcjonowaniu poddrzew (w których poszukiwania są zadaniami wzajemnie niezależnymi) i przekazaniu obliczeń do innych węzłów środowiska sytemu.

Algorytm składa się z następujących głównych procedur:
- inicjalizacji rozproszenia i wyboru liderów grup
- wyboru dawcy obliczeń,
- detekcji zakończenia.

4.1. Inicjalizacja rozproszenia i wybór liderów grup

Do przeprowadzenia obliczeń zadane jest środowisko rozproszone w postaci grafu środowiska II poziomu S_1 oraz dla każdej grupy G_i grafu środowiska I poziomu $S_{2,i}$ charakteryzującego tę grupę i „widocznego" tylko w jej obrębie.

Inicjalizacja obliczeń polega na przekazaniu całości obliczeń do dowolnego węzła dowolnej grupy – oznaczmy odpowiednio p_I i G_I (oczywiście $p_I \in G_I$). Węzeł p_I tworzy następnie drzewo $T \subset S_1$ o wierzchołku G_I rozpinające graf S_1. Drzewo tworzone jest algorytmem przeglądu grafu wszerz. Ustalane są także wagi poddrzew będące sumą mocy grup wchodzących w skład poddrzewa

$$pow_T(T_i \subset T) = \sum_{\bar{G} \in \Gamma(T_i)} pow_G(\bar{G}) \tag{3}$$

Wybierani są liderzy grupy według schematu:
a) dla G_I jest to p_I,
b) tworzony jest zbiór $D = \{G_I\}$,
c) następnie dla każdej grupy $G_j \in D$ wybierany jest wierzchołek sąsiedni $G_i \notin D$ na podstawie drzewa T.

Węzeł p_i należący do krawędzi łączącej G_j i G_i ($p_i = \lambda(\{G_j, G_i\}, p_i)$) jest obierany liderem grupy, a G_i dodawany jest do zbioru D ($D = D \cup \{G_i\}$). Punkt c) jest powtarzany, gdy $D \neq \Gamma(T)$. Ze specyfiki algorytmu wynika, że pewne węzły mogą pełnić rolę liderów więcej niż jednej grupy – nie jest to przypadek sprzeczny, gdyż obydwie funkcje są pełnione współbieżnie, niezależnie od siebie[2)] .

[2)] W tym przypadku modyfikacji ulega wartość funkcji pow_T (3) – moc obliczeniowa węzła wliczana jest tylko do jednej grupy (tej, o największym potencjale, lub dowolnej w przypadku równości).

Na etapie inicjalizacji z góry definiowane są dodatkowo dwie wielkości określające minimalne wysokości poddrzew przesyłanych wewnątrz (h_{min}), jak i między grupami (H_{min}) przy rozpraszaniu obliczeń. Zabieg ten ma na celu wstrzymanie nieopłacalnego, zbytniego „rozdrobnienia" poddrzew i związanego z nim opóźnienia powodowanego narzutem komunikacyjnym pomiędzy węzłami.

Następnym krokiem jest rozdzielenie obliczeń na całe środowisko, które odbywa się w dwóch fazach:

1. podział danych między liderów grup – węzeł p_I dzieli drzewo poszukiwań pomiędzy liderów poszczególnych grup w taki sposób, aby każda grupa otrzymała kompletne poddrzewo, którego wielkość $q = 2^h$ (gdzie h – wysokość drzewa) powinna być jak najbliższa współczynnikowi $pow_G(G_i)$ wyrażającemu moc obliczeniową grupy[3)];
2. podział danych wewnątrz grupy – lider grupy otrzymane od p_I drzewo dzieli pomiędzy węzły swojej grupy wg zasad analogicznych do podziału pierwotnego drzewa opisanego powyżej.

Przedstawiony algorytm podziału środowiska, wyboru lidera grupy oraz podziału danych ma za zadanie umożliwienie jak najlepszego zbalansowania obciążenia całego, przewidywalnie rozległego systemu oraz zminimalizowanie liczby przesyłanych komunikatów. Opisany algorytm różni się od spotykanych dotychczas, a głównym tego powodem jest dodatkowa wiedza zawarta w modelu opisującym środowisko rozproszone. Dla przykładu zaproponowane w [6] rozwiązanie polegało na sukcesywnym podziale na pół otrzymanego poddrzewa pomiędzy siebie, a kolejnego sąsiada. W rezultacie występowało słabe zrównoważenie rozkładu potencjalnych „obliczeń" pomiędzy węzły wynikające z nikłej wiedzy dotyczącej środowiska.

4.2. Wybór dawcy obliczeń – *load-balancing*

Czasy zakończenia obliczeń pierwotnie przydzielonych węzłom przewidywalnie będą od siebie bardzo odległe. Sytuacja taka spowodowana jest to różną ilością danych, które zostaną wysłane, możliwością wystąpienia obcięć w drzewie poszukiwań, jak i również różną mocą obliczeniową poszczególnych jednostek. W tym celu skonstruowany został mechanizm jak najlepszego zrównoważenia obciążenia (*load-balancing*).

W literaturze spotyka się dwa ogólne rozwiązania powyższego problemu – inicjacja przez:

1) dawcę,
2) biorcę [7].

Pierwsze z nich polega na poszukiwaniach biorcy prowadzonych przez węzeł z nadmiernym obciążeniem, w celu pozbycia się części pracy. Drugie – wykorzystane przez autora – na poszukiwaniu danych przez jednostkę bezczynną.

W prezentowanym algorytmie równoważenie obciążenia dokonywane jest na dwóch poziomach – wewnątrz grupy oraz pomiędzy grupami. Jak wspomniano wcześniej wystę-

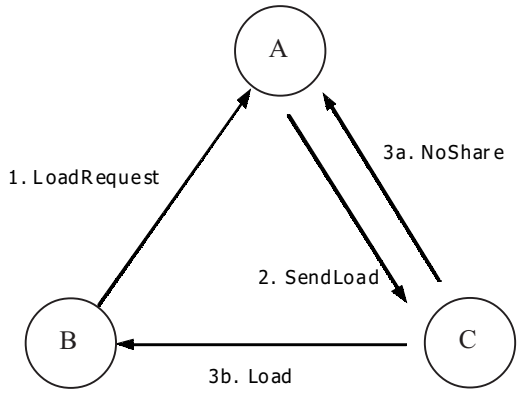
---

[3)] Złożoność obliczeniowa algorytmu zaproponowanego przez autora wynosi N * log(N), gdzie N jest liczbą grup w środowisku (algorytm nie został przedstawiony w niniejszej pracy).

pują pewne minimalne wartości przesyłanej „pracy do wykonania" (oznaczone jako h_{min} oraz H_{min}), poniżej których nieopłacalne staje się oddanie części poddrzewa bezczynnemu węzłowi kosztem wysłania komunikatu.

W momencie braku danych proces bezczynny /B/ wysyła do lidera grupy /A/ komunikat /1/ z żądaniem danych do prowadzenia dalszych obliczeń (rys. 2). Lider grupy próbuje najpierw znaleźć dawcę wewnątrz grupy wysyłając komunikat /2/ do kolejnych procesów z nakazem podzielenia się danymi zawierający proces docelowy; może uzyskać w ten sposób odmowę /3a/ lub brak wiadomości świadczący o przesłaniu części danych /3b/.

Nie bez znaczenia jest wewnętrzny algorytm lidera grupy wybierający potencjalnego dawcę obliczeń, mający w jak najlepszy sposób rozłożyć obciążenie oraz zminimalizować tym samym liczbę potencjalnych późniejszych komunikatów. Posiada on listę L oraz kolejki Q_T i Q. Na liście L oraz w kolejce Q_T utrzymywane są węzły, do których nie należy kierować nakazów dzielenia się danymi, natomiast kolejka Q jest zbiorem potencjalnych dawców, w której początkowo znajdują się wszystkie węzły grupy. Należy dodać, ze każdy węzeł może znajdować w maksymalnie jednej z wymienionych struktur na raz – wszystkie opisane dalej operacje wstawiania usuwają go automatycznie z pozostałych.

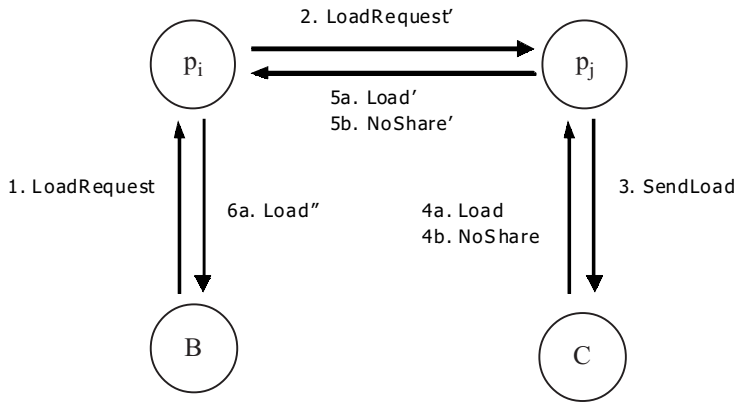


Rys. 2. Schemat komunikacji w obrębie grupy
(A – lider grupy, B – proces bezczynny, C – potencjalny dawca danych)

Przyjmując oznaczenia zgodnie z rysunkiem 2 algorytm przedstawia się następująco: po wysłaniu komunikatu /2/ do pierwszego procesu z kolejki Q (w tym przypadku /C/), węzły /B/ i /C/ dodawane są na koniec kolejki Q_T i jeżeli po ustalonym okresie czasu T nie nadejdzie komunikat /3a/, przesuwane są na koniec kolejki Q. Jeżeli natomiast dotrze komunikat /3a/ węzeł /C/, wstawiany jest do listy L, a proces poszukiwania dawcy danych dla /B/ jest powtarzany.

Jeżeli kolejka Q jest pusta rozpoczyna się proces poszukiwania dawcy poza lokalną grupą: /B/ wstawiany jest do listy L, a lider /p_i/ wysyła żądanie przekazania danych do lidera /p_j/ grupy sąsiedniej (rys. 3). Lider /p_j/ bierze odpowiedzialność za znalezienie dawcy – w przypadku znalezienia pobiera od niego dane (przesuwając na koniec własnej kolejki Q), przesyła dane do /p_i/, który następnie przekazuje je do /B/ (i wstawia go na koniec

lokalnej Q). W przypadku gdy /p_j/ nie znajdzie dawcy, informuje o tym /p_i/, przesyłając także własną listę wykluczonych grup – ten rozpoczyna proces poszukiwania kolejnego lidera wśród pozostałych grup przeszukując graf S_1 wszerz. Grupy, których liderzy odpowiedzieli brakiem danych do podziału (oraz grupy z listy przez nich samych wykluczone) nie są już nigdy odwiedzane (także podczas kolejnych poszukiwań).



Rys. 3. Schemat poszukiwania dawcy poza grupą
(p_i, p_j – liderzy grup, B – proces bezczynny, C – potencjalny dawca danych)

Do weryfikacji liczby przesyłanych komunikatów – jako wyznacznika sprawności działania systemu – użyty został zaproponowany przez autora symulator środowiska rozproszonego. Zasada jego działania polega na uruchomieniu algorytmów liczących w zadanej topologii sieci i zbadaniu aktywności komunikacyjnej.

Zaproponowany przez autora sposób równoważenia obliczeń jest nieco bardziej skomplikowany, lecz jak wykazały doświadczenia, wnosi poprawę, ograniczając liczbę przesyłanych komunikatów. Inne prace opisujące schemat obliczeń w rozległych środowiskach rozproszonych, bazując na bardziej ubogich modelach, nie mogły uwzględnić struktury połączeń, co odbijało się na uzyskiwanej wydajności.

Dla przykładu w [6] opisano algorytm, według którego węzeł bezczynny pobierał dane od swojego sąsiada – na skutek eksperymentów na ubogich w liczbę maszyn sieciach nie zauważono, że prowadziło to do dużego niezrównoważenia, a także do zbyt częstego przesyłania komunikatów między odległymi maszynami. Dodatkową niewygodą był przymus „odgórnej" numeracji wszystkich maszyn biorących udział w obliczeniach. Dla dużych problemów rozwiązanie autora zmniejszało liczbę przesyłanych komunikatów o około 35%. Wspomnieć należy, że dużą rolę odgrywało odpowiednie dobranie współczynników h_{min} oraz H_{min}.

W [8] przedstawiono sposób wyboru węzła-dawcy polegający na wprowadzeniu „potencjalności" opisującej przewidywalną długość czasu obliczeń. Jej wartość estymowana była przez wydzieloną jednostkę w niewielkiej grupie węzłów, prowadzącą dość wzmożoną komunikację z innymi maszynami w celu ustalenia zaawansowania obliczeń. Na podobnej

zasadzie działa schemat wyboru węzła-dawcy w grupie przedstawiony przez autora, jednak czynnikiem estymującym najlepszego kandydata jest nie bezpośrednia wiedza o jego obliczeniach, lecz moment, w którym dzielił się on posiadaną porcją danych. Jak sprawdzono, jest to rozwiązanie minimalizujące liczbę przesyłanych komunikatów, przy jednoczesnym zachowaniu nieznacznie gorszej, lecz wystarczająco dobrej trafności predykcji.

4.3. Detekcja zakończenia obliczeń

Omawiając sposób dystrybucji danych, nie wspomniano o sposobie otrzymywania wyników obliczeń cząstkowych. Otóż wynik zwracany jest liderowi grupy razem z komunikatem *LoadRequest*. Lider przekazuje go następnie do p_I w momencie, gdy cała grupa zakończy obliczenia – dzieje się to wtedy, gdy nie występują już grupy-dawcy oraz wszystkie węzły grupy lokalnej zakończyły przydzielone prace (wszystkie wysłały komunikat *LoadRequest* nie otrzymując później danych). Jeżeli istnieje wiele optimów, na każdym poziomie wybierane jest jedno dowolne. W momencie gdy p_I otrzyma odpowiedzi od wszystkich grup, zwraca wynik globalny.

5. Wnioski końcowe oraz kierunki przyszłych badań

Zaproponowany przez autora algorytm rozpraszania obliczeń przeglądu zupełnego połączonego z techniką obcięć w drzewie poszukiwań charakteryzuje się zmniejszoną liczbą przesyłania komunikatów. Uzyskanie optymistycznych wyników spowodowane jest użyciem zaproponowanego przez autora modelu środowiska [5], który przy niskim stopniu skomplikowania pozwala w stosunkowo dokładny sposób oddać rzeczywistą sytuację występującą w rozległych sieciach komputerowych. Obecnie trwają prace nad przystosowaniem oraz instalacją rozwijanego przez autora systemu wieloagentowego, przeznaczonego między innymi do prowadzenia obliczeń rozproszonych [10]. W planach istnieje wykorzystanie wielu niezależnych, „oddalonych" od siebie podsieci komputerowych.

Dodatkowym atutem, który należy podkreślić jest uniwersalność zaproponowanej metody – algorytm rozpraszania obliczeń z powodzeniem można zastosować do rozwiązywania innych problemów *NP*-trudnych w środowiskach rozproszonych. Deterministyczny algorytm wyznaczania obcięć drzewa poszukiwań można zastąpić metodą niedeterministyczną, nie zmniejszając przy tym wydajności zaproponowanych przez autora rozwiązań, gdyż nie są one bezpośrednio zależne od momentu wystąpienia obcięcia.

Celem przyszłych badań jest głównie wzbogacenie algorytmu o możliwość wystąpienia awarii w rozległym systemie komputerowym. Założeniem autora jest minimalizacja strat powodowanych nagłym odłączeniem pojedynczych węzłów, jak i całych podsieci dokonujących obliczeń.

Literatura

[1] Lawler E.L., Wood D.E.: *Branch and bound methods: A survey.* Operations Research, (14), 1966, 670–719

[2] Boyd S., Ghosh A., Magnani A.: *Branch and Bound Methods.* Notes for EE392o, Stanford University, 2003

[3] Cormen, Leiserson, Rivest: *Wprowadzenie do algorytmów*. WNT, 2000
[4] Gendron B., Crainic T.G.: *Parallel branch-and-bound algorithms: Survey and synthesis*. Technical Report 913, Centre de recherche sur les transports, Montreal (Canada), May 1993
[5] Tusiewicz M.: *Grafowy model środowiska systemu wieloagentowego*. VI Międzynarodowe Warsztaty Doktoranckie OWD 2004, vol. 4
[6] Drummond, Filho, Uchoa, Castro: *Towards a Grid Enabled Branch-and-Bound Algorithm*. International Symposium on Mathematical Programming (ISMP 2003), Copenhagen, Denmark
[7] Clausen J.: *Parallel Branch-and-Bound, Principles and Personal Experiments*. Kluwer Academic Publishers, 1997, 239–267
[8] Cung V. D., Dowaji S., Le Cun B., Mautor T., Roucairol C.: *Parallel and Distributed Branch-and-Bound/A* Algorithms.* (94/31). Laboratoire PRISM, Université de Versailles, 1994
[9] Dowaji, Roucairol: *Load balancing strategy and priority of tasks in distributed environments*. Fourteenth Annual IEEE Conference, International Phoenix Conference on Computers and Communications, p.15–22, Scottsdale, Arizona, USA, March 28–31, 1994
[10] Tusiewicz M.: *System wieloagentowy: teoria, projekt, implementacja oraz przykłady zastosowań*. Kraków, Instytut Informatyki, Uniwersytet Jagielloński, 2003
[11] Bruin, Kindervater, Trienekens: *Asynchronous Parallel Branch-and-Bound and Anomalies*. Erasmus University, Department of Computer Science, EUR-CS-95-05, Roterdam, Netherlands, 1995
[12] Yahfoufi, Dowaji: *A Self-Stabilizing Distributed Branch-and-Bound Algorithm*. Proceedings of the 1996 IEEE 15th Annual International Phoenix Conference on Computers and Communications
[13] www.grid.org