2004－ARC－160　(5)
2004／12／2

組合せオークションの勝者決定問題に対する分枝限定解法の機能並列化手法

Chen QIAO　水戸 将弥　田頭 茂明　藤田 聡

広島大学 大学院工学研究科 情報工学専攻
〒 739-8527 東広島市鏡山 1-4-1

概要

本稿では，従来のデータ並列に代わり機能並列に着目した分枝限定解法の新しい並列化手法を提案する．提案手法では，ネットワーク上に分散した自律エージェントの集合を利用することにより，異質で不規則となる分枝限定解法の機能並列を柔軟に実現する．また，提案手法の実現へ向けた幾つかの設計事項を検討し，その性能確認のために実施した事前実験の結果について示す．結果から各エージェントが環境に応じて自身の機能を自律的に変更することで，性能の飛躍的な向上が期待できることを確認できた．

Parallel Branch-and-Bound Scheme for the Winner Determination Problem in Combinatorial Auctions

Chen QIAO　Masaya MITO　Shigeaki TAGASHIRA　Satoshi FUJITA

Department of Information Engineering, Graduate School of Engineering
Hiroshima University, Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527 Japan

Abstract

In this paper, we propose a new class of parallel branch-and-bound (B&B) schemes. The main idea of the scheme is to focus on the *functional parallelism* instead of conventional *data parallelism*, and to support such a heterogeneous and irregular parallelism by using a collection of autonomous agents distributed over the network. After examining several design issues toward the implementation of a prototype of the distributed B&B system, we illustrate the result of our preliminary experiments conducted to estimate the performance of the proposed scheme. The result shows that it could cause a significant performance improvement if each agent autonomously changes its function type according to the change of the underlying environment.

1　Introduction

According to the recent advancement of network technologies, it emerges an increasingly strong requirement for high performance computing over the large-scale interconnection networks. In general, a high complexity of server procedures will limit the scalability of distributed systems, and it motivates the study of *fully distributed systems* such as grid computers and pure peer-to-peer (P2P) systems. A P2P system consists of a collection of host computers called nodes or peers [1, 4], and those nodes are connected with each other by an interconnection network such as the Internet. In recent years, a lot of important services such as shared file systems and Domain Name Systems (DNS) are constructed over the P2P model, and they have been used in many application fields, such as electronic bulletin board, network auction systems, and so on.

In this paper, we propose a new application field for such fully distributed systems, and discuss several implementation issues to realize it in actual distributed environments. As the concrete target of our research, we will focus our attention to a distributed execution of parallel branch-and-bound (B&B) schemes [2, 5], that have been applied to many important fields as a generic solver to generate an optimum solution to computationally hard optimization problems in a rel-

atively short computation time. In addition, as the concrete problem to be solved, we will focus on the Winner Determination Problem (WDP, for short) in combinatorial auctions, that has also been studied extensively in recent years to realize a fair matchmaking among individual customers participating to e-Markets and e-Auctions (a formal definition of WDP will be given in the next section). It should be worth noting that in most of previous work, parallel B&B schemes are designed by merely focusing on the *data parallelism* that naturally exists in exhaustive tree search schemes. Although it would be slightly complicated compared with a simple OR parallelism, the difference is mainly due to the mutual dependency between the upper and the lower bounds, that could be efficiently handled by adopting an appropriate broadcast mechanism within the framework of data parallelism.

In this paper, we propose a new class of parallel B&B schemes. The main idea of the scheme is to focus on the *functional parallelism* instead of conventional data parallelism, and to support such a heterogeneous and irregular parallelism by using a collection of autonomous agents. More concretely, we prepare a logical shared space to keep the entire search tree, and provide an infrastructure in which each agent autonomously updates upper and lower bounds for each partial solution corresponding to a vertex in the search tree, in a fully distributed manner. After examining several design issues toward the implementation of a prototype of the distributed B&B system, we will illustrate the result of our preliminary experiments conducted to estimate the performance of the proposed scheme. The result of experiments shows that the goodness of a given function strongly depends on the characteristics of the given instance, and it could cause a significant performance improvement if each agent autonomously changes its function type according to the change of the underlying environment.

The remainder of this paper is organized as follows. Section 2 describes necessary definitions and concepts, including a formal definition of the problem and an overview of the branch-and-bound method. Section 3 describes the proposed method. The result of initial evaluation of the proposed scheme is given in Section 4, and finally in Section 5, we conclude the paper with future problems.

2 Preliminaries

2.1 Problem

Let $S = \{x_1, x_2, \ldots, x_m\}$ be a set of **goods** sold by the auctioneer. In combinatorial auctions, buyers submit a set of bids to the auctioneer, where a bidding is made on a subset of goods instead of a single good as in classical auctions, and the auctioneer selects a subset of those bids in such a way to maximize the revenue of the auctioneer. A bidder of a selected bid is called a "winner" of the auction. In this paper, we assume that each bidder can submit any number of bids, and can be a winner of several bids, without loss of generality[1]. Note that this assumption enables us to separate bids from bidders.

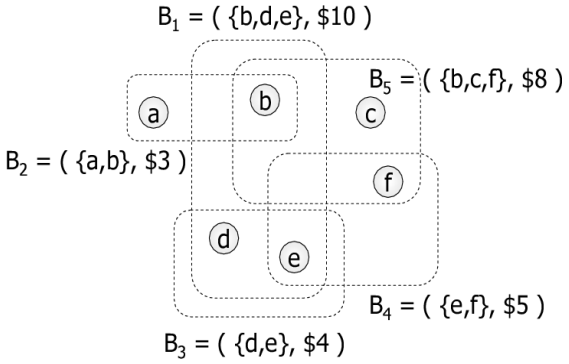


Figure 1: An instance of WDP (each good is represented as a small circle, and each bid B_i enclosed by a dotted line is a pair of its bid set and bid value).

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ be a set of bids submitted by the bidders. Each bid $B_i \in \mathcal{B}$ is an ordered pair $\langle S_i, v_i \rangle$, where S_i is a nonempty subset of S called **bidset** (or simply "bid") and v_i is an integer referred to as the **bid value** (or simply "value"). A subset $\mathcal{B}'$ of $\mathcal{B}$ is said to be **feasible** if any two bids in the subset do not intersect with each other. In addition, a bid $B_i (\in \mathcal{B})$ is said to be feasible with respect to $\mathcal{B}' (\subseteq \mathcal{B})$ if set $\mathcal{B}' \cup \{B_i\}$ is feasible. The **revenue** of subset $\mathcal{B}' (\subseteq \mathcal{B})$, denoted by $r(\mathcal{B}')$, is the sum of bid values contained in $\mathcal{B}'$. The winner determination problem (WDP) is the problem of, given a finite set of bids $\mathcal{B}$, finding a feasible subset $\mathcal{B}'$ of $\mathcal{B}$ with a maximum revenue.

Figure 1 illustrates an instance of WDP, that consists of six goods and five bids. A feasible solution to the instance is $\{B_2, B_4\}$ with revenue 8 $(= 3 + 5)$ dollars, and an optimum solution to the instance is $\{B_3, B_5\}$ since the revenue of the solution takes a maximum value 12.

2.2 Branch-and-Bound Method

The basic idea of the branch-and-bound (B&B) method for solving WDP is described as follows. In what follows, a feasible subset of $\mathcal{B}$ is referred to as a **partial solution** (thus, the objective of this problem could be restated as to find a partial solution with a maximum revenue).

Let $\mathcal{B}'$ be a feasible subset of $\mathcal{B}$. In a **list scheduling** (LS, for short) method, all bids in $\mathcal{B}$ are first given

[1] It is known that this assumption could violate an efficient assignment of wins to the bidders when there are substitutional goods in S. Such a problem could be resolved by introducing dummy goods shared by substitutional bids [3].

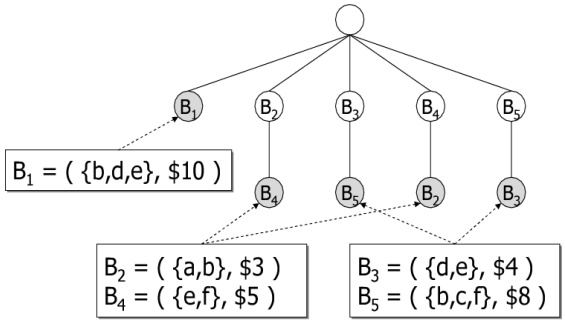


Figure 2: Search tree for the instance in Figure 1.

a total ordering, and those bids are sequentially selected to be contained in the partial solution, in such a way that any two selected bids do not intersect with each other. It is known that LS is a "complete" scheme in the sense that for any instance, there exists a total ordering of bids in $\mathcal{B}$ to generate an optimum solution under the scheme. In other words, by attempting LS for all of the $n!$ permutations, we can always find an optimum solution to WDP. This idea can be realized by conducting an exhaustive search in a tree structure satisfying the following three properties: 1) each vertex of the tree corresponds to a partial solution, 2) the root of the tree corresponds to a partial solution with respect to an empty set of bids, and 3) if a vertex x corresponds to a partial solution, then a child of x corresponds to a partial solution that is obtained by greedily appending a single bid to x, that is not contained in x and does not intersect with any bid in x. Note that in such search trees, any path from the root to a leaf corresponds to a permutation over a feasible subset of $\mathcal{B}$.

Figure 2 illustrates a search tree corresponding to the instance shown in Figure 1. The tree consists of ten vertices including the root vertex, and the depth of the tree (i.e., the maximum path length from the root to a leaf vertex) is two. In this figure, each leaf vertex is painted gray, and is associated with a partial solution.

The B&B method performs a depth-first (or best-first) search over the above tree structure in an exhaustive manner. Recall that LS merely examines a single path from the root to a leaf, and it attempts no backtracking. A trick to reduce the execution time is to "prune" subtrees if it is guaranteed that there can exist no better solutions than the currently best one on the subtrees. Such a guarantee is generally realized by evaluating an *upper bound* for each partial solution, where an upper bound concerned with a partial solution implies that any solution generated from the partial solution can not be better than that bound. Note that in a partial solution $\mathcal{B}'$, the tight upper bound concerned with a child vertex is no greater than the tight upper bound concerned with its parent vertex.

3 Proposed Scheme

3.1 Design Concept

In this paper, we consider a distributed execution of parallel B&B schemes. The main issues for realizing efficient B&B schemes could be summarized as follows:

1. How to find a better partial solution quickly, and
2. How to calculate a sharp upper bound quickly.

It should be worth noting that those two issues are closely related with each other. That is, the time before finding a better partial solution could be reduced by pruning as many meaningless branches as possible, and the possibility of pruning a branch at a given upper bound could generally be increased by providing a better partial solution. In other words, those two issues must be simultaneously taken into account in designing time-efficient B&B schemes. It should also be noted that the estimated upper bound for a vertex in the search tree must be recorded at the vertex in order to make it possible to prune the subtree rooted at the vertex when a better partial solution is discovered later.

In our proposed scheme, the function of each agent is designed by focusing on the following two points. The first point is concerned with the upper bound: *Point 1) There is a trade-off between the cost and the accuracy of calculating an upper bound.* That is, in general, we could obtain a sharper upper bound by spending more calculation time. However, since the objective of calculating a sharp upper bound is to prune meaningless branches as much as possible, in this context, this problem could be regarded as a simple YES/NO problem (i.e., the result is whether we could prune the subtree rooted at the partial solution or not). Hence in order to realize a pruning with a low calculation cost, we should prepare several methods for calculating upper bounds, and should apply them sequentially in the order of lower calculation cost. The next point we have to consider is about the lower bound: *Point 2) There is a dilemma in determining the expansion order of partial solutions.* In general, a bid order that quickly derives a better lower bound could not derive partial solutions that are unlikely to be pruned by upper bounds. Such a dilemma could particularly be observed when we could determine the bid selecting order for each partial solution independently. More concretely, a subtree that could not be efficiently pruned is a branch whose upper bound could not be accurately calculated, that is generally different from a branch that is likely to derive a better lower bound.

The above trade-off and dilemma are due to the fact that *we have to make a selection from several*

candidates, and thus, could be relaxed by introducing the notion of parallel execution.

First, as for the trade-off on the upper bound, we could resolve the problem by preparing (at least) two kinds of agents, i.e., basic agent and advanced agent, and by executing those agents concurrently, in such a way that: 1) basic agents calculate the initial upper bound for each partial solution, and 2) advanced agents try to improve the initial upper bound for several selected partial solutions. In the selection of partial solutions, for example, we could take into account the success rate of pastly executed pruning operation, the level of partial solution in the search tree, and the expected calculation time for the improvement. In realizing such a mechanism in distributed environments with no centralized control, we have to design each agent in such a way that those selections are conducted in a heuristic and autonomous manner.

On the other hand, as for the dilemma on the way of expansion, we could resolve the problem by expanding several branches simultaneously, while we have to introduce a kind of strategies since the amount of available resources is finite (if an infinite number of processes were available, we could solve any NP-complete problem in a polynomial time !). One possible strategy is to use the following two phase control; i.e., initially, a quick improvement of the lower bound is given a higher priority, and after observing the saturation of the improvement speed, it switches to another heuristic in which a branch that is unlikely to be pruned is given a higher priority.

3.2 Upper Bound Agents

In the prototype system that will be examined in the next section, the following two types of upper bound (UB) agents are prepared, and each UB agent continuously and asynchronously tries to improve the upper bound on partial solutions.

3.2.1 Type TRV

An agent of this type calculates an upper bound on the revenue that could be derived from the partial solution, based on a heuristic estimation of expected revenue [6]. Given feasible set of bids $\mathcal{B}'(\subseteq \mathcal{B})$, let us define an *estimated revenue* with respect to $\mathcal{B}'$ as follows:

$$h(\mathcal{B}') \stackrel{\text{def}}{=} \sum_{x \in S'} \left\{ \max_{S_j \ni x, S_j \cap (S - S') = \emptyset} \left(\frac{v_i}{|S_i|} \right) \right\} \qquad (1)$$

where S' is the set of goods that are not contained in bids in $\mathcal{B}'$. By using the calculated value, an upper bound on the partial solution $\mathcal{B}'$ is calculated as

$$r(\mathcal{B}') + h(\mathcal{B}')$$

since it has already selected bids with total revenue $r(\mathcal{B}')$.

For example, an upper bound for the root vertex in the search tree shown in Figure 2, is calculated as follows: Since $\mathcal{B}' = \emptyset$, $S' = S = \{a, b, c, d, e, f\}$. For example, good b in S' is contained in three bids $B_1 = \langle\{b, d, e\}, 10\rangle$, $B_2 = \langle\{a, b\}, 3\rangle$, and $B_5 = \langle\{b, c, f\}, 8\rangle$, and for those bids, the value of $v_i/|S_i|$ is calculated as $v_1/|S_1| = 10/3$, $v_2/|S_2| = 3/2$, and $v_5/|S_5| = 8/3$, respectively. In Equation (1), the maximum of them, i.e., $10/3$ is selected as the maximum contribution of good b to the estimated revenue. In a similar way, the maximum contribution of each good to the expected revenue is estimated as $3/2$ for a, $8/3$ for c, $10/3$ for d, $10/3$ for e, and $8/3$ for f; and by taking a summation of them, we have an estimated revenue with respect to the root vertex as:

$$h(\emptyset) = \frac{3}{2} + \frac{10}{3} + \frac{8}{3} + \frac{10}{3} + \frac{10}{3} + \frac{8}{3} = 16 + \frac{5}{6}.$$

Since $r(\emptyset) = 0$, it outputs $16 + 5/6$ as an upper bound for the root vertex.

Note that the value of function h could be obtained very quickly since we can calculate $v_i/|S_i|$ for each bid B_i beforehand, and goods contained in subset S' could be efficiently identified by using a simple counter-based scheme.

3.2.2 Type LP

An agent of this type calculates an upper bound for each partial solution $\mathcal{B}'$ by solving the corresponding linear programming (LP), defined as follows:

$$\begin{aligned} &\text{maximize} \sum_{B_i \in \mathcal{B}'} v_i p_i \\ &\text{subject to} \sum_{B_i \in \mathcal{B}'} a_{ji} p_i \leq 1 \text{ for all } j \in S \end{aligned}$$

where $0 \leq p_i \leq 1$ and $a_{ji} = 1$ if $x_j \in S_i$ and 0 otherwise. In the above formulation, several bids containing the same good in common can be selected in a fractional manner with fraction p_i, as long as the sum of such fractions does not exceed one. Note that the original formulation of WDP does not allow such a fractional selection, and an optimum solution to the above (relaxed) LP is not smaller than an optimum solution to the original problem.

4 Experiments

4.1 Environment

To estimate the goodness of the proposed B&B scheme, we conducted several experiments under the following environment:

- A distributed system consisting of five homogeneous host computers (one master and four workers) with the following characteristics connected by a network of 1Gbps: CPU: 3.2GHz Pentium IV; Memory: 2GB; OS: FreeBSD 4.7

- All programs are written in C language.

In the experiments, we used the following two classes of instances, each of which models bid distributions observed in real-world applications [7].

Random: Each bid B_i is constructed by selecting k_i goods from S without replacement and by assigning a bid value v_i to it, where k_i is a random value drawn from $\{1, 2, \ldots, m'\}$, where $m' \leq m$, and v_i is a random value drawn from $\{1, 2, \ldots, Max\}$.

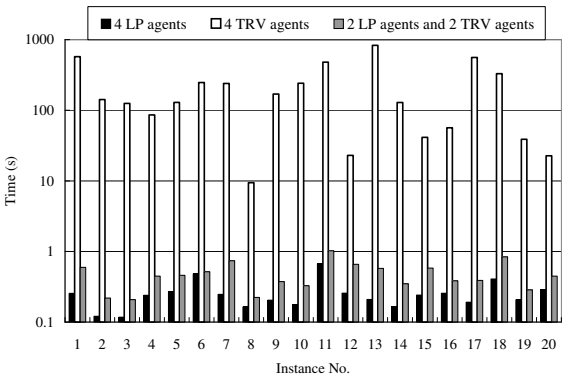
Uniform: Modify Random in such a way that the size of each bid is fixed to a constant k.

More concretely, we use the following two suites of instances, each of which consists of 20 randomly generated instances: Random with average bid size three, 100 goods, and 150 bids (abbreviated as R3 hereafter) and Uniform with bid size ten, 100 goods, and 1500 bids (abbreviated as U10 hereafter).

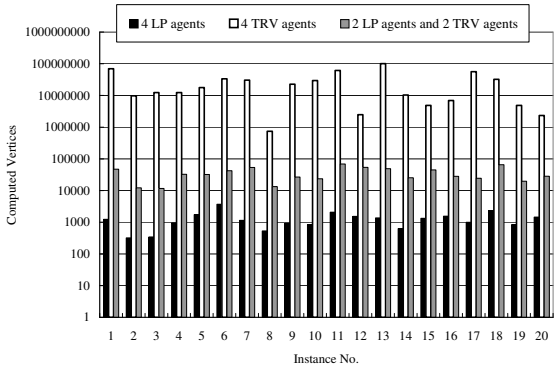
4.2 Results

In the following experiments, we associate one UB agent for each of four workers, and evaluate the computation time and the total number of examined vertices by varying the percentage of advanced agents from 0% to 100% (note that if a vertex in the search tree is examined by two different kinds of UB agents, it is counted as "twice" instead of "once"). More concretely, we consider the following three cases in the experiments: Case 1) all of the four agents are of type LP; Case 2) all of the four agents are of type TRV; and Case 3) two agents are of type LP and the remaining two agents are of type TRV. In each case, the master partitions a given instance to 64 subproblems and associate them to four workers in a static manner. Each worker then begins to improve the lower bound for the associated subproblem by using a best-first search based on a normalized bid value; i.e., bids in $\mathcal{B}$ are examined in a non-increasing order of $v_i/|S_i|$.

Figure 3 summarizes the results for suite R3, where (a) represents the computation time and (b) represents the total number of examined vertices. The horizontal axis of the figures represents the identifier of instances that is numbered from 1 to 20, and the vertical axis of the figures is represented in a logarithmic scale. As is shown in Figure 3 (a), the first scheme (with four LP agents) could solve the instances much quicker than the second scheme (with four TRV agents), and the computation time of the mixed scheme could be bounded as small as the first scheme. A superiority of the first scheme could be well explained by examining the total number of vertices. See Figure 3 (b). As is shown in the figure, by using four LP agents instead of four TRV agents, the number of examined vertices could be reduced to less than 1/10000, whereas the overall computation time reduces to less than 1/100. The number of examined vertices in the mixed scheme is ten times greater than



(a) Computation time.

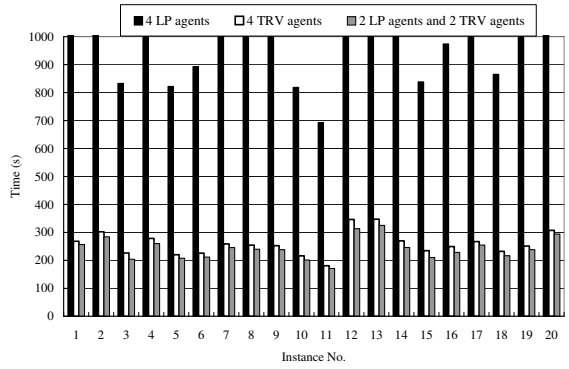


(b) Total number of exemined vertices.

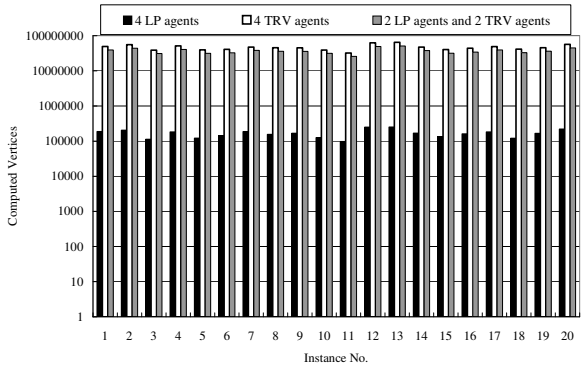
Figure 3: Results for suite R3.

the first scheme, that is probably because of redundant calculations of upper bounds, conducted by the remaining two TRV agents in the mixed scheme.

Figure 4 summarizes the results for suite U10, where we bounded the maximum execution time by 1,000 seconds. For this suite, in contrast to suite R3, the second scheme exhibits a better performance than the first scheme, and in addition, the mixed scheme gives a slightly better performance than the second one. The badness of the first scheme is obviously due to the inaccuracy of upper bounds calculated by LP agents compared with the case for suite R3. See Figure 4 (b) for illustration. As is shown in the figure, although it increases the total number of examined vertices to at most ten times of suite R3 for the second scheme, it increases the number of examined vertices to 100 times for the first scheme. In other words, for suite U10, the second scheme unnecessarily examines many vertices because of the inaccuracy of the upper bounds calculated by LP agents. On the other hand, the superiority of the third scheme compared with the second scheme could be explained by the inheritance of the upper bound derived for an ancestor vertex by LP agents; i.e., could be explained by the fact that an upper bound for a vertex (that has been calculated



(a) Computation time.



(b) Total number of examined vertices.

Figure 4: Results for suite U10.

by LP agents by spending a long computation time) is an upper bound for all of its descendant vertices. For example, by using the mixed scheme instead of the second one, we could reduce the computation time for the 15th instance by 10%, and the number of examined vertices by 22%; i.e., from 40,440,489 vertices to 31,715,122 vertices.

Finally, by the above results, we can conclude that the goodness of a given function strongly depends on the characteristics of the given instance, and it would cause a significant performance improvement if each agent could autonomously changes its function type according to the change of the underlying environment.

5 Concluding Remarks

In this paper, we proposed a new class of parallel B&B schemes by focusing on the functional parallelism existing in the "branch" and "bound" operations, and discussed several design issues toward the implementation of a prototype system in actual distributed environments. We also illustrated the result of our preliminary experiments, and the result shows that it could cause a significant performance improvement if each agent autonomously changes its function type according to the change of the underlying environment. We are now developing a prototype system of the distributed B&B system, and the evaluation of the overall performance of the system is left as an important future problem.

References

[1] I. Clarke, O. Sandberg, B. Wiley, T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *ICSI workshop on Design Issues in Anonymity and Unobsevability*, pp.46–66 (July 2000).

[2] J. Clausen , M. Perregaard. On the best search strategy in parallel branch-and-bound: Best-First Search versus Lazy Depth-First Search. *Annals of Operations Research*, Kluwer Academic Publishers, 1999, 90(1): 1-17(17), 1999.

[3] Y. Fujishima, K. Leyton-Brown, Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. IJCAI'99*, pp.548–553, 1999.

[4] Gnutella. `http://gnutella.wego.com/`.

[5] Portable Parallel Branch-and-Bound Library `http://wwwcs.upb.de/fachbereich/AG/monien/SOFTWARE/PPBB/ppbblib.html`

[6] Y. Sakurai, M. Yokoo, K. Kamei. An efficient approximate algorithm for winner determination in combinatorial auctions. In *ACM Conf. on Electronic Commerce*, pp.30–37, 2000.

[7] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2): 1–54, 2002.