JSIAM Letters Vol.3 (2011) pp.29–32 ©2011 Japan Society for Industrial and Applied Mathematics JSIAM Letters

Approximation algorithms for a winner determination problem of single-item multi-unit auctions

Satoshi Takahashi[1] and Maiko Shigeno[1]

[1] Graduate School of System and Information Systems, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan

E-mail *stakahashi@sk.tsukuba.ac.jp*

Received September 30, 2010, Accepted January 13, 2011

Abstract

This paper treats a winner determination problem of a Vickrey-Clarke-Groves mechanism based single-item multi-unit auction. For this problem, two simple 2-approximation algorithms are proposed. One is a linear time algorithm using a linear knapsack problem. The other is a greedy type algorithm. In addition, a fully polynomial time approximation algorithm based on a dynamic programming is described. Computational experiments verify availabilities of our algorithms by comparing computational times and approximation ratios.

Keywords auction theory, winner determination, approximation algorithm

Research Activity Group Discrete Systems

1. Introduction

Recent Internet auctions with huge participators require to compute an optimal allocation and payments as quick as possible. A winner determination problem on auction theory consists of an item allocation problem and a payment determination problem, which depends on an auction mechanism. One of the most desirable auction mechanisms is due to Vickrey, Clarke and Groves, which is called VCG [1]. Throughout this paper, we consider only VCG based auctions. Winner determination problems of VCG based auctions are known as $\mathcal{NP}$-hard. Therefore, it is important to consider fast approximation algorithms for a winner determination problem in the Internet auction environment.

We treat single-item multi-unit auctions, where a seller who wants to sell M units of a single item and n bidders participate. Each bidder i submits sets of anchor values $\{d_i^k \mid k = 0, \ldots, \ell_i\}$ and of unit values $\{e_i^k \mid k = 1, \ldots, \ell_i\}$, where anchor values satisfy $d_i^{k-1} < d_i^k$ for any $0 < k \leq \ell_i$ and e_i^k implies a unit value over half-open range $(d_i^{k-1}, d_i^k]$ of item quantity. Without loss of generality, we assume that $d_i^0 = 0$ and $d_i^{\ell_i} \leq M$ for every bidder i. Let $N = \{1, \ldots, n\}$ be a set of bidders and $\ell = \sum_{i \in N} \ell_i$. We define a value function $v_i : \mathbb{R}_+ \to \mathbb{R}$ of bidder i by

$$v_i(x) = \begin{cases} e_i^k \cdot x & (d_i^{k-1} < x \leq d_i^k,\ k = 1, \ldots, \ell_i), \\ 0 & (x = d_i^0 \text{ or } x > d_i^{\ell_i}). \end{cases}$$

Our item allocation problem (AP) is to find each quantity x_i that bidder i gets such that the total valuation is maximized. It is formulated as

$$(AP) \quad \left| \begin{array}{ll} \text{maximize} & \sum_{i \in N} v_i(x_i) \\ \text{subject to} & \sum_{i \in N} x_i \leq M,\ x_i \geq 0 \quad (\forall i \in N). \end{array} \right.$$

Denoted by $\boldsymbol{x}^\star$, an optimal solution for (AP) is. We say that a solution for (AP) satisfies an "anchor property" if there are at least $n-1$ bidders whose getting quantities are given by their anchor values.

Lemma 1 ([2]) *A problem (AP) has an optimal solution satisfies the anchor property, when every bidder's unit values are monotone non-increasing for k.*

To compute a payment of bidder j, we need to solve a restricted problem excepting j from bidders. Let $N^{-j} = N \setminus \{j\}$ and $\boldsymbol{x}^{-j}$ be an optimal solution of an item allocation problem under the set N^{-j}, that is,

$$\left| \begin{array}{ll} \text{maximize} & \sum_{i \in N^{-j}} v_i(x_i) \\ \text{subject to} & \sum_{i \in N^{-j}} x_i \leq M,\ x_i \geq 0 \quad (\forall i \in N^{-j}). \end{array} \right.$$

On a VCG based auction, a payment p_j of bidder j is defined by

$$p_j = \textstyle\sum_{i \in N^{-j}} v_i(x_i^{-j}) - \sum_{i \in N^{-j}} v_i(x_i^\star). \qquad (1)$$

We now review briefly approximation algorithms for a winner determination problem of single-item multi-unit auctions. With respect to constant-factor approximations for (AP), Kothari, Parkes and Suri [2] proposed a 2-approximation algorithm with $\mathrm{O}(\ell^2)$ time for the so-called generalized knapsack problem which models an item allocation problem in reverse auction. When their greedy algorithm is applied to (AP) directly, it returns a solution whose approximation ratio may not be bounded by two. Zhou [3] said that he improved this algorithm to run in $\mathrm{O}(\ell \log \ell)$ time. Moreover, he showed a 3-approximation algorithm with $\mathrm{O}(\ell)$ time and a $(9/4)$-approximation algorithm with $\mathrm{O}(\ell \log \ell)$ time for the so-called interval multiple-choice knapsack problem whose special case is (AP). According to [3], it is an open problem to compute a 2-approximation of (AP) in linear time. With respect to fully polynomial time approximation schemes (FPTAS) for a winner determination problem, Kothari, Parkes and Suri [2] proposed the first one which is based on dynamic programming and uses the

anchor property. It finds a solution with an approximation ratio at most $(1+\epsilon)$ for (AP) in $\mathrm{O}(n\ell^2/\epsilon)$ time and calculates every bidders' payment in $\mathrm{O}((n\ell^2/\epsilon)\log(n/\epsilon))$ time. In order to solve (AP), their algorithm repeats fixing a specified bidder j and index $0 < k \le \ell_j$, and solving the problem (AP) adding a constraint $d_j^{k-1} < x_j \le d_j^k$. This FPTAS was improved by Zhou [3]. His algorithm does not repeat to compute (AP) with an additional constraint. Thus, Zhou's FPTAS solves a winner determination problem in $\mathrm{O}((n\ell/\epsilon)\log(n/\epsilon))$ time. Moreover, by employing a technique of vector merge, he say that his algorithm can run in $\mathrm{O}((n\ell/\epsilon)\log n)$ time. However, a solution found by his algorithm may not satisfy the anchor property.

In the next section, we propose two 2-approximation algorithms for solving (AP). One is an $\mathrm{O}(\ell)$ algorithm, which gives a positive answer to the open problem in [3]. The other is an $\mathrm{O}(\ell(\log n + \ell_{\max}))$ algorithm based on greedy method improved from [2], where $\ell_{\max} = \max_{i\in N}\ell_i$. We also describe an FPTAS for solving a winner determination problem in $\mathrm{O}((n\ell/\epsilon)\log(n/\epsilon))$ time, which finds a solution satisfying the anchor property. Section 3 shows computational experiments comparing performances of proposed algorithms.

2. Approximation algorithms

This section proposes simple 2-approximation algorithms for (AP) and an FPTAS that is modified version of Zhou's algorithm [3] for a winner determination problem to obtain a solution satisfying the anchor property.

2.1 2-approximation algorithms for item allocation problems

We propose two 2-approximation algorithms for (AP). One is based on Dyer's polynomial time algorithm [4] for linear knapsack problem:

$$(LKP)\quad \left|\begin{array}{ll} \text{maximize} & \sum_{i\in N}\sum_{k=0}^{\ell_i}(e_i^k d_i^k)y_i^k \\ \text{subject to} & \sum_{i\in N}\sum_{k=0}^{\ell_i} d_i^k y_i^k \le M, \\ & \sum_{k=0}^{\ell_i} y_i^k = 1 \quad (\forall i\in N), \\ & y_i^k \ge 0 \quad (\forall i\in N,\ \forall k = 0,\dots,\ell_i), \end{array}\right.$$

where, for any $i \in N$, e_i^0 is given by an arbitrary value.

Lemma 2 *The optimal value of (LKP) gives an upper bound of (AP).*

Proof For a feasible solution $\boldsymbol{x}$ of (AP), we can construct a feasible solution $\boldsymbol{y}$ for (LKP) by setting, if there exists an index k_i with $d_i^{k_i-1} < x_i \le d_i^{k_i}$,

$$y_i^k = \begin{cases} x_i/d_i^{k_i} & (k = k_i), \\ 0 & (k > 0, k \ne k_i), \\ 1 - x_i/d_i^{k_i} & (k = 0), \end{cases}$$

$$\text{and, otherwise,}\quad y_i^k = \begin{cases} 1 & (k = 0), \\ 0 & (k \ne 0). \end{cases}$$

The objective values of these solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy

$$\begin{aligned}\sum_{i\in N}\sum_{k=0}^{\ell_i}(e_i^k d_i^k)y_i^k &= \sum_{\{i\in N\mid 0<x_i\le d_i^{\ell_i}\}} e_i^{k_i}x_i \\ &= \sum_{i\in N} v_i(x_i).\end{aligned}$$

Hence, the optimal value of (LKP) is not less than the optimal value of (AP). **(QED)**

With respect to a feasible solution $\boldsymbol{y}$ of (LKP), we call an index i as *saturated* if $y_i^k = 1$ holds for some k. It is known that there exists an optimal solution for (LKP) with at most one unsaturated index. Let $\boldsymbol{y}^*$ be such an optimal solution and i^* be the unsaturated index. From $\boldsymbol{y}^*$, we construct two solutions of (AP) by setting

$$\hat{x}_i = \begin{cases} \sum_{k=0}^{\ell_i} d_i^k y^{*k}_i & (i \ne i^*), \\ 0 & (i = i^*), \end{cases} \tag{2}$$

$$\text{and}\quad \tilde{x}_i = \begin{cases} 0 & (i \ne i^*), \\ d_i^{\tilde{k}_i} & (i = i^*), \end{cases} \tag{3}$$

where $\tilde{k}_i$ is an index attaining $\max_{0<k\le\ell_i} e_i^k d_i^k$. Obviously, both solutions $\hat{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$ are feasible for (AP). Moreover, we have

$$\begin{aligned}&\textstyle\sum_{i\in N}\sum_{k=0}^{\ell_i} e_i^k d_i^k y^{*k}_i \\ &\quad= \textstyle\sum_{i\in N^{-i^*}}\sum_{k=0}^{\ell_i} e_i^k d_i^k y^{*k}_i + \sum_{k=0}^{\ell_{i^*}} e_{i^*}^k d_{i^*}^k y^{*k}_{i^*} \\ &\quad\le \textstyle\sum_{i\in N} v_i(\hat{x}_i) + \sum_{i\in N} v_i(\tilde{x}_i) \\ &\quad\le 2\cdot\max\{\textstyle\sum_{i\in N} v_i(\hat{x}_i), \sum_{i\in N} v_i(\tilde{x}_i)\}. \end{aligned}\tag{4}$$

Our approximation algorithm can be described as follows.

Algorithm AA1

Step 1 Find an optimal solution $\boldsymbol{y}^*$ of (LKP) with at most one unsaturated index i^*.

Step 2 From $\boldsymbol{y}^*$, get two feasible solutions $\hat{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$ by (2) and (3). If $\sum_{i\in N} v_i(\hat{x}_i) \ge \sum_{i\in N} v_i(\tilde{x}_i)$, then return $\hat{\boldsymbol{x}}$, otherwise, return $\tilde{\boldsymbol{x}}$.

Theorem 3 *Algorithm AA1 finds a 2-approximation solution for (AP) in $\mathrm{O}(\ell)$ time.*

Proof Lemma 2 and inequality (4) derive our approximation ratio. Because (LKP) can be solved in linear time [4], we obtain our time complexity. **(QED)**

Since Algorithm AA1 runs in linear time, it gives a positive answer to the open problem, which in [3].

We now turn to the other 2-approximation algorithm that is greedy type. Our algorithm uses sloop functions $p_i^k : \mathbb{R} \to \mathbb{R}$, for $i \in N$ and $0 < k \le \ell_i$, given by a gradient of the value function v_i between a current allocated unit x and each anchor value d_i^k, i.e.,

$$p_i^k(x) = (v_i(d_i^k) - v_i(x))/(d_i^k - x).$$

We describe our greedy type algorithm as follows.

Algorithm AA2

Step 1 Set $x_i = 0$ for any $i \in N$.

Step 2 Find a pair (i^*, k^*) such as $p_{i^*}^{k^*}(x_{i^*}) = \max\{ p_i^k(x_i) \mid i \in N,\ x_i < d_i^k\}$. If $p_{i^*}^{k^*}(x_{i^*}) \le 0$, then return $\boldsymbol{x}$, otherwise, update $x_{i^*} = d_{i^*}^{k^*}$.

Step 3 If $\sum_{i\in N} x_i < M$, go to Step 2.

Step 4 Make two solutions $\hat{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$ by

$$\hat{x}_i = \begin{cases} x_i & (i \ne i^*), \\ M - \sum_{j\ne i^*} x_j & (i = i^*), \end{cases}$$

$$\text{and} \quad \tilde{x}_i = \begin{cases} 0 & (i \neq i^*), \\ x_{i^*} & (i = i^*). \end{cases}$$

If $\sum_{i \in N} v_i(\hat{x}_i) > \sum_{i \in N} v_i(\tilde{x}_i)$, then return $\hat{\boldsymbol{x}}$, otherwise, return $\tilde{\boldsymbol{x}}$.

Theorem 4 *Algorithm AA2 finds a 2-approximation solution of* (AP) *in* $\mathrm{O}(\ell(\log n + \ell_{\max}))$ *time, where* $\ell_{\max} = \max_{i \in N} \ell_i$.

Proof When AA2 stops at Step 2, we can show that the returned solution is an optimal for (AP). When AA2 stops at Step 4, for a solution $\boldsymbol{x}$ at the end of AA2, let $M' = \sum_{i \in N} x_i$. It can be shown that the solution $\boldsymbol{x}$ is optimal for

$$\left| \begin{array}{ll} \text{maximize} & \sum_{i \in N} v_i(x_i) \\ \text{subject to} & \sum_{i \in N} x_i \leq M',\ x_i \geq 0 \quad (\forall i \in N). \end{array} \right.$$

Since $\boldsymbol{x}^\star$ is also feasible for the above problem, we have $\sum_{i \in N} v_i(x_i^\star) \leq \sum_{i \in N} v_i(x_i)$. It comes from the definitions of $\hat{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$ that $\sum_{i \in N} v_i(x_i) \leq \sum_{i \in N} v_i(\tilde{x}_i) + \sum_{i \in N} v_i(\hat{x}_i)$ holds. Thus, we obtain the desired approximation ratio.

It is clear that the number of iteration of AA2 is at most ℓ. If we store $\max\{p_i^k(x_i) \mid x_i < d_i^k\}$ for all $i \in N$ in a heap, Step 2 can be performed in $\mathrm{O}(\log n)$. After Step 2, we need to compute $\max\{p_{i^*}^k(x_{i^*}) \mid x_{i^*} < d_{i^*}^k\}$ for updated x_{i^*}, which runs in $\mathrm{O}(\ell_{i^*})$. Hence, the total running time is bounded by $\mathrm{O}(\ell(\log n + \ell_{\max}))$.

(QED)

2.2 FPTAS for winner determination problems

We show a modified version of Zhou's algorithm [3] such that it finds a solution satisfying the anchor property, when every bidder's unit values satisfy $e_i^{k-1} \geq e_i^k$ for all $1 \leq k \leq \ell_i$. Let $\epsilon > 0$ be a relative error and V be an objective value obtained by a 2-approximation algorithm. We define a scaled value function $\tilde{v}_i : \mathbb{R}_+ \to \mathbb{R}$ of bidder i by

$$\tilde{v}_i(x) = \lfloor (n \cdot v_i(x))/\epsilon \cdot V \rfloor.$$

We denote an item allocation problem over this scaled value function by $(\widetilde{AP})$. For an optimal solution $\boldsymbol{x}$ of $(\widetilde{AP})$, we have

$$\begin{aligned} \sum_{i \in N} v_i(x_i^\star) &< \sum_{i \in N} (\epsilon V/n)(\tilde{v}_i(x_i^\star) + 1) \\ &\leq \sum_{i \in N} (\epsilon V/n) \tilde{v}_i(x_i) + \epsilon V \\ &\leq \sum_{i \in N} v_i(x_i) + \epsilon \sum_{i \in N} v_i(x_i^\star). \end{aligned}$$

Thus, an optimal solution for $(\widetilde{AP})$ is a solution with a relative error at most ϵ for (AP). In order to solve $(\widetilde{AP})$ by dynamic programming, for two parameters t and r, the value $\min\{\sum_{i=1}^t x_i \mid \sum_{i=1}^t \tilde{v}_i(x_i) \geq r\}$ is stored in $G[t,r]$ and $H[t,r]$, where in $G[t,r]$ each x_i is restricted to an anchor value, and in $H[t,r]$ each x_i except only one bidder is restricted to an anchor value. An optimal solution of $(\widetilde{AP})$ is obtained from a solution $\boldsymbol{x}^{[n,r^*]}$ establishing $H[n,r^*]$, where r^* attains $\max_r\{\sum_{i \in N} \tilde{v}_i(x_i^{[n,r]}) \mid H[n,r] \leq M\}$. In order to obtain r^*, it is enough to search $H[n,r]$ for r from 0 to $\lfloor (2n)/\epsilon \rfloor$, since the optimal value of $(\widetilde{AP})$ is bounded by $\lfloor (2n)/\epsilon \rfloor$. Thus, our algorithm finds an optimal solution for $(\widetilde{AP})$ by computing $H[t,r]$ recursively, together with $G[t,r]$, for $r = 0, \ldots, \lfloor (2n)/\epsilon \rfloor$ and $t = 0, \ldots, n$. It is obvious that we can initialize $G[0,0] = H[0,0] = 0$ and $G[0,r] = H[0,r] = \infty$ for any $r > 0$. For convenience, we set $G[t,r] = H[t,r] = \infty$ for any $t \in N$ and $r < 0$. By recursively, we can represent

$$G[t,r] = \min\{G[t-1,r], \min_k (G[t-1, r - \tilde{v}_t(d_t^k)] + d_t^k)\}.$$

Defining $m[t,r]$ by

$$\min_{0 \leq r' \leq r} \{\min\{x_t \mid \tilde{V}_G[t-1,r'] + \tilde{v}_t(x_t) \geq r\} + G[t-1,r']\},$$

where $\tilde{V}_G[t-1,r']$ is the value $\sum_{i=1}^{t-1} \tilde{v}_i(x_i)$ for a solution $\boldsymbol{x}$ that establishes $G[t-1,r']$, we have the following recurrence for H:

$$\begin{aligned} H[t,r] = \min\{&H[t-1,r], \\ &\min_k (H[t-1, r - \tilde{v}_t(d_t^k)] + d_t^k), m[t,r]\}. \end{aligned}$$

In $m[t,r]$, we can rewrite $\min\{x_t \mid \tilde{V}_G[t-1,r'] + \tilde{v}_t(x_t) \geq r\}$ by

$$\min\{x_t \mid \lfloor n e_t^k x_t/\epsilon V \rfloor \geq r - \tilde{V}_G[t-1,r'], d_t^{k-1} < x_t \leq d_t^k\}. \tag{5}$$

Since $r - \tilde{V}_G[t-1,r']$ is an integer, the smallest x_t satisfies the first condition in (5) is given by $(r - \tilde{V}_G[t-1,r'])/e_t^k \cdot (\epsilon V/n)$. Thus (5) is equivalent to

$$\begin{aligned} \min_k \{&(r - \tilde{V}_G[t-1,r'])/e_t^k \cdot (\epsilon V/n) \\ &\mid d_t^{k-1} < (r - \tilde{V}_G[t-1,r'])/e_t^k \cdot (\epsilon V/n) \leq d_t^k\}. \end{aligned}$$

By using this formula, the values $m[t,r]$ for all $r = 1, \ldots, \lfloor (2n)/\epsilon \rfloor$ can be found simultaneously in $\mathrm{O}((n\ell_i/\epsilon) \log(n/\epsilon))$ time. After obtaining the values of $m[t,r]$ for all $r = 1, \ldots, \lfloor (2n)/\epsilon \rfloor$, we can compute each $G[t,r]$ and $H[t,r]$ in $\mathrm{O}(\ell_i)$ time. Therefore we obtain entire elements of G and H in $\mathrm{O}((n\ell/\epsilon) \log(n/\epsilon))$ time.

Finally, we compute each bidder's payment defined by (1) by employing the method of Kothari, Parkes and Suri [2]. In their method, all payments can be computed in the same time complexity to obtain G and H.

Theorem 5 *Our algorithm finds a solution with a relative error at most* ϵ *of* (AP) *in* $O((n\ell/\epsilon) \log(n/\epsilon))$ *time. It also finds every payment in the same time complexity.*

If a vector merge technique by [3] is applied, our algorithm solves a winner determination problem in $\mathrm{O}((n\ell/\epsilon) \log n)$ time.

3. Experimental results

This section shows computational results of algorithms described in Section 2. All computations were conducted on a personal computer with Core2 Duo CPU (3.06GHz) and 4GB memory. Our code was written by python2.6.5. For given numbers of bidders n and of units M, all instances used in this experiment were generated using random numbers. The number of anchor values ℓ_i for each bidder i was selected uniformly from integers within the interval $[1, 15]$. Every unit value e_i^k and anchor value d_i^k were selected uniformly from integers in $[1, 100]$ and in $[1, M]$, respectively.

Table 1 shows averages of computational times and

Table 1. Averages of computational times and of approximation ratios of 2-approximation algorithms for ten instances of each (n, M).

instance	comp. times (sec.)		app. ratios		instance	comp. times (sec.)		app. ratios	
(n, M)	AA1	AA2	AA1	AA2	(n, M)	AA1	AA2	AA1	AA2
$(10, 200)$	0.00460	0.00119	1.873	1.093	$(10, 50)$	0.00388	0.00101	1.299	1.251
$(50, 200)$	0.01171	0.00409	1.736	1.244	$(10, 100)$	0.00314	0.00102	1.458	1.296
$(100, 200)$	0.02331	0.00768	1.505	1.195	$(10, 200)$	0.00460	0.00119	1.873	1.093
$(200, 200)$	0.04721	0.01440	1.673	1.429	$(50, 50)$	0.01237	0.00392	1.398	1.485
$(400, 200)$	0.09666	0.02869	1.492	1.399	$(50, 100)$	0.01430	0.00390	1.356	1.371
$(800, 200)$	0.19884	0.05754	1.559	1.316	$(50, 200)$	0.01171	0.00409	1.736	1.244
$(1000, 200)$	0.25427	0.06933	1.749	1.639	$(100, 50)$	0.02468	0.00755	1.831	1.206
$(5000, 200)$	1.84318	0.34113	1.550	1.328	$(100, 100)$	0.02401	0.00737	1.566	1.618
$(10000, 200)$	5.08805	0.72498	1.548	1.609	$(100, 200)$	0.02331	0.00768	1.505	1.195

Table 2. Averages of computational times (sec.) and relative errors of our FPTAS for ten instances with $n = 10$ and $M = 50$.

epsilon	comp. times (sec.)		relative errors	
	AA1	AA2	AA1	AA2
1.0	0.27925	0.66331	0.057	0.038
0.9	0.32014	0.78129	0.038	0.037
0.8	0.40650	0.98921	0.041	0.047
0.7	0.49900	1.20912	0.042	0.034
0.6	0.67769	1.66427	0.026	0.021
0.5	0.92052	2.26631	0.037	0.021
0.4	1.42961	3.50712	0.024	0.001
0.3	2.54312	6.25542	0.014	0.001
0.2	5.59242	13.77336	0.012	0.004
0.1	22.06652	54.52771	0.007	0.003

approximation ratios of two 2-approximation algorithms, Algorithm AA1 and Algorithm AA2, for ten instances of each size. Algorithm AA1 was implemented so that its time complexity was $O(\ell \log \ell)$, since we employed a sorting algorithm instead of linear-time median finding in Dyer's algorithm for (LKP). It is consistent with the theoretical complexities that resulting computational times depend on n but not M. On the other hand, Algorithm AA2 is faster than Algorithm AA1 in the average times, because our instances seem not to derive worst cases. Approximation ratios of both algorithms seem not to be affected by sizes of n and M. In our results, Algorithm AA2 tended to have better average approximation ratio than Algorithm AA1. This tendency was influenced by a few solutions with bad approximation ratios. At the end of both algorithms Algorithm AA1 and Algorithm AA2, they choose a solution $\hat{\boldsymbol{x}}$ or $\tilde{\boldsymbol{x}}$. Among 150 instances of this experiment, Algorithm AA1 returned a solution $\tilde{\boldsymbol{x}}$ in 89 instances and Algorithm AA2 returned $\tilde{\boldsymbol{x}}$ in 23 instances. Indeed, a solution given by $\tilde{\boldsymbol{x}}$, which was returned by Algorithm AA2 especially, did not have so good approximation ratio, because it allocated almost all units to only one bidder. This fact seems to affect evaluations of approximation ratios.

The second experiment evaluated behavior of our FPTAS described in Subsection 2.2 to solve a problem (AP). We investigated influence of a given relative error ϵ on computational times and on obtained relative errors. In addition, we compared performance of our FPTAS where the value V is given by Algorithm AA1 and Algorithm AA2, respectively. Table 2 shows averages of computational times and of relative errors for ten instances fixed with $n = 10$ and $M = 50$. In our result, while the case using Algorithm AA1 spent less times on computing than the case using Algorithm AA2, the latter frequently returned solutions with better relative errors than the former, which derived from the fact that Algorithm AA2 tended to return a greater value V than Algorithm AA1. Although the theoretical complexity does not depend on V, we say there is a difference of average computational times between the cases using Algorithm AA1 and Algorithm AA2. This difference comes from each computational time of $m[t, r]$. By using Algorithm AA2, our FPTAS returned almost optimal solutions, when ϵ is less than 0.4. However, the returned allocations were different from optimal.

4. Concluding remarks

For winner determination problems of a VCG based single-item multi-unit auction, we proposed two 2-approximation algorithms for item allocation problems. One runs in the linear time, which gives a positive answer to the open problem in [3]. The other does not run in the linear time, but it computes fast in some experiments. We also discussed an FTPAS, which returned an approximation solution satisfying the anchor property.

When some bidders know all bids and can compute optimal allocations and payments, they may not approve an approximate solution whose allocations and payments are entirely different from optimal ones. To approve approximate solutions in real auctions, we need some rules about allocations. For instance, Fukuta and Ito [5] discussed a rule that bidder j is allocated no more than an allocation of bidder i if $v_i(d) > v_j(d)$ for some anchor value d. It is our future work to develop an approximation algorithm for finding an allocation satisfying this rule.

References

[1] P. Milgrom, Putting Auction Theory to Work, Cambridge Univ. Press, 2004.

[2] A. Kothari, D. C. Parkes and S. Suri, Approximately-strategyproof and tractable multiunit auctions, Decision Support System, **39** (2005), 105–121.

[3] Y. Zhou, Improved multi-unit auction clearing algorithms with interval (multiple-choice) knapsack problems, in: Proc. of 17th Int. Sympo. on Algorithms and Computation, pp. 494–506, 2006.

[4] M. E. Dyer, An $O(n)$ algorithm for the multiple-choice knapsack linear program, Math. Program., **29** (1984), 57–63.

[5] N. Fukuta and T. Ito, An analysis about approximated allocation algorithms of combinatorial auctions with large numbers of bids (in Japanese), IEICE Trans. D, **J90-D** (2007), 2324–2335.