Optimization Heuristics for the Combinatorial Auction Problem

Michael Schwind
Dept. of Economics, esp. Information Systems
Mertonstr. 17
D-60054 Frankfurt, Germany
schwind@wiwi.uni-frankfurt.de

Tim Stockheim
Dept. of Economics, esp. Information Systems
Mertonstr. 17
D-60054 Frankfurt, Germany
stockheim@wiwi.uni-frankfurt.de

Franz Rothlauf
Dept. of Information Systems 1
Schloss
D-68131 Mannheim, Germany
rothlauf@uni-mannheim.de

Abstract- This paper presents and compares three heuristics for the combinatorial auction problem. Besides a simple greedy (SG) mechanism, two metaheuristics, a simulated annealing (SA), and a genetic algorithm (GA) approach are developed which use the combinatorial auction process to find an allocation with maximal revenue for the auctioneer. The performance of these three heuristics is evaluated in the context of a price controlled resource allocation process designed for the control and provision of distributed information services. Comparing the SG and SA method shows that depending on the problem structure the performance of the SA is up to 20% higher than the performance of the simple greedy allocation method. The proposed GA approach, using a random key encoding, results in a further improvement of the solution quality. Although the metaheuristic approaches result in higher search performance, the computational effort in terms of used CPU time is higher in comparison to the simple greedy mechanism. However, the absolute overall computation time is low enough to enable real-time execution in the considered IS application domain.

1 Introduction

Price-controlled resource allocation in IT systems is a common research topic since the use of distributed information systems became more widespread [1]. One way of achieving an optimal allocation of the resources needed to provide information services is to use auction mechanisms [2]. A crucial problem in this context is the nonlinear valuation of the requested resource bundles due to technological interdependencies. These interdependencies can be addressed by the formulation of appropriate bid prices for resource bundles depending on the usability of a specific factor combination in the service provision process. The allocation of the constrained resources is then accomplished by a *combinatorial auction* (CA). The underlying *combinatorial auction problem* (CAP) is NP-hard [3]. Therefore, CAP-algorithms dealing with time critical applications, such as the provision of distributed IT services, rely on the use of heuristics and metaheuristics.

In this work we develop three (meta)heuristics for the CAP and evaluate their performance in a *price controlled resource allocation scenario* (PCRAS). The scenario emulates resource requirements occurring, for instance in connection with the provision of distributed *information services and information production* (ISIP), such as the simultaneous usage of network and computing capacity to enable web-based video conferencing and telecommunication applications between corporations. Alternatively, the supply and accounting of video-on-demand has similar application properties. The time-delayed transfer of data, which is collected during daily business activities in large corporations, could be seen as a further instance of an IT problem where optimal automated resource allocation is necessary. As a first solution we present a simple greedy heuristic that is based on a bundle price-resource load ratio. Based on this bid selection strategy we also propose a simulated annealing (SA) algorithm and a genetic algorithm (GA) using the random key encoding. Both methods improve the solution quality for the CAP.

2 Solving the Combinatorial Auction Problem

Performing CAs in a PCRAS context means that a bidder agent a_i submits bid bundles $b_{i,j}$ which include the requests $q_{i,j}(r,t)$ for the resources r in a specific quantity q at a particular point of time t. i denotes the number of the agent, and j the number of the agent's bid.

Due to the bidders time-dependent nonlinear valuation of the resource allocation, especially the complementary property of the bidders utility function $v_i(\{b_{i,j}\})$, the CAP is NP-hard [3, 4]. For the PCARS the valuation of two bids ($j = 1, 2$) submitted by an agent i is super-additive if technological complementarities exist:[1]

$$v_i(\{b_{i,1}\}) + v_i(\{b_{i,2}\}) < v_i(\{b_{i,1}\} \cup \{b_{i,2}\}) \qquad (1)$$

The CAP is often denoted as the winner determination problem (WDP) according to the traditional auctioneers task of

[1] The bidder (producer) applies for a particular combination of resources at a specifi c point of time t. If not all requested resources are assigned to him, the partial acquisition of the resources has much less value because of production delays or even production failures. This results in the higher than linear (superadditive) valuation of the bundled goods compared to the single items valuation.

identifying the winner. The formal description of the CAP could be considered as a special variant of the *weighted set packing problem* (WSPP) [5] and is formulated as:

$$\max \sum_{i=1}^{I} \sum_{j=1}^{J} p_{i,j}\ x_{i,j}$$

subject to

$$\sum_{i=1}^{I} \sum_{j=1}^{J} q_{i,j}(r,t)\ x_{i,j} \leq q_{max}(r,t), \qquad (2)$$

where $r \in \{1,\ldots,R\}, t \in \{1,\ldots,T\}$ and

$$\sum_{j=1}^{J} x_{i,j} \leq 1, \quad \text{where } i \in \{1,\ldots,I\}.$$

Resources:	r	$\in$	$\mathbb{N}$
Time slots:	t	$\in$	$\mathbb{N}$
Resource requests:	$q_{i,j}(r,t)$	$\in$	$\mathbb{N}$
Price for bid $b_{i,j}$:	$p_{i,j}$	$\in$	$\mathbb{R}^+$
Acceptance variable:	$x_{i,j}$	$\in$	{0,1}
Bid j of agent i:	$b_{i,j}$	$\in$	B

The goal is to maximize the auctioneers income. $q_{max}(r,t)$ is the maximum capacity of resources at time t available to the auctioneer and B is the set of all bids $b_{i,j}$. Furthermore, we refer to the set of accepted bids as B_{acc} (with $B_{acc} \subseteq B$).

The search for an optimal, or at least near optimal, solution to the CAP is mainly done by approaches like integer programming [6, 7] or branch and bound [8, 9]. Due to the high computational effort of such approaches, heuristics based on greedy allocation strategies are employed to solve the CAP, accepting a trade-off between solution quality and computational effort. Mostly, a simple greedy CA algorithm (SG-CAA) consists of two steps:

- According to a revenue oriented criteria (e.g. average price per bid-bundle, single-item respectively) submitted bids are sorted in an ordered list.
- CA allocation is done by adding ordered bids from the list as long as they are not ruled out by bids which are already included.

The allocation quality achieved with the SG-CAA usually depends on the sorting criteria and the bidder's utility function. Many approaches combine greedy allocation with more sophisticated heuristics like SA [10]. [11] presented an SA-based approach to solve the CAP in a supply chain setting, where contracts for task allocation are negotiated based on temporal and precedence constraints. The use of a GA as a metaheuristic to solve the CAP is proposed by [12]. In this work tasks are matched to a service provider using a fitness function based on criteria such as bandwidth and server capacity. Other GA approaches addressed the WSPP to find an optimal solution for real-world problems like flight crew scheduling [13]. [14] implemented a parallel GA, which uses up to 128 subpopulations. Although this approach is able to handle a large number of bundles, the algorithm has difficulties when dealing with highly constrained set partitioning problems.

3 Price-Controlled Resource Allocation Scenario

In the following section we propose a *general price-controlled resource allocation scenario* for evaluating and comparing the performance of the three heuristics.

A request $b_{i,j}$ is formulated as a 2-dimensional *bid matrix* (BM) describing which resource[2] r, where $r \in \{1,\ldots,R\}$, is requested at time t, where $t \in \{1,\ldots,T\}$. A simple example for a PCRAS request is presented in Table 1. Each entry of the BM denotes the amount $q_{i,j}(r,t)$ of resource units needed. To every bid matrix a bid price p belongs which indicates the bidders willingness-to-pay.

After describing the load request on the demand side, the resource allocation on the supply side should be modeled. In our scenario this is done by employing an *allocation matrix* (AM), which has the same structure as BM. Integrating a task into the schedule of the resource provider is done simply by aggregating the current BM into the AM. A violation of a resource load constraint can be detected by comparing the AM with the constraint q_{max} (which is equal for all resource types over all time slots).

The agents willingness $p_{i,j}$ to pay for a bid depends on the overall resource load

$$q_{i,j}^{ovl} = \sum_{r=1}^{S} \sum_{t=1}^{N} q_{i,j}(r,t) \qquad (3)$$

and is calculated as

$$p_{i,j} = q_{i,j}^{ovl}\ p_{fact}. \qquad (4)$$

p_{fact} is a random decimal number from the interval $[p_{min}, p_{max}]$ and q_{bmax} denotes the maximum resource load that can be requested by a bidder for a single BM-element $q_{i,j}(r,t)$. In our three test instances each entry in the BM is occupied with probability p_{tso}. This means, $b_{i,j} = 0$ with probability $1 - p_{tso}$. If $b_{i,j} \neq 0$, $b_{i,j}$ is uniformly distributed (in our test scenarios $b_{i,j}$ is uniformly distributed between 1 and 3).

In our work we use three different test instances. Each of them is a representative instance of relevant real-world scenarios:

- **Unstructured Bids:** The fist type of BMs contains single resource requests with a maximum bid length of one time slot associated with a specific resource load. A BM depicting this bid type is shown in Table 1.
- **Substructured Bids:** In a more realistic environment, bidder agents require resources with the same intensity for a longer period of time (up to l_{max} slots). This results in continuous bids of varying length,

[2] Computing power, volatile computer memory, non-volatile storage capacity, and data transfer bandwidth are the four resources relevant in our problem.

called substructured bids. An example is shown in Table 2.

- **Structured Bids:** Some application cases require resources to be allocated synchronously within shifts. The coherent bids must not cross the shift limits, but may have different constant load intensities for the particular resource types during the shift. A sample for such a structured BM is displayed in Table 3.

resource	time slot t											
	1	2	3	4	5	6	7	8	9	10	11	12
r_1	1		2		3		1			2		2
r_2		3			1	2	3		1		3	1
r_3			1	1			1				2	
r_4	1				1					1	3	1

Table 1: Example of an unstructured PCRAS bid matrix

resource	time slot t											
	1	2	3	4	5	6	7	8	9	10	11	12
r_1	3	3	3			2	2	2				
r_2		2	2	2				1	1	1		
r_3			1	1					2	2	2	
r_4	1	1	1							3	3	3

Table 2: Example of a substructured PCRAS bid matrix

resource	time slot t											
	1	2	3	4	5	6	7	8	9	10	11	12
r_1	3	3	3			2	2		2			
r_2	2	2	2			3	3		1			
r_3	1	1	1			1	1		2			
r_4	1	1	1			2	2		3			

Table 3: Example of a structured PCRAS bid matrix

4 Three Heuristics for the Combinatorial Auction Problem

In the following section algorithms for the solution of the CAP in the ISIP resource bundle allocation context are presented. As mentioned above, they are based on three common types of heuristics: SG, SA and GA.

4.1 Simple Greedy CA Algorithm

We start with the presentation of a simple greedy CA algorithm (SG-CAA). The SG-CAA implementation uses a max_f-function to sort the bids according to the ratio $f_{i,j} = p_{i,j}/q_{i,j}^{ovl}$ (price over the overall resource load). See Table 4 for an example.

All bids $b_{i,j}$ are labeled from 1 to L, where $L = I * J$ is the total number of bids. The SG-CAA iteratively inserts a bid to the accepted bid set B_{acc} if no other agent's bid is already in B_{acc} (XOR-condition), and if the resource load constraint is not violated (compare step 4):

bid nr.	2	6	4	5	1	3
res. load	111	141	133	85	126	93
bid price	294	344	251	132	157	104
$p_{i,j}/q_{i,j}^{ovl}$	2.65	2.44	1.89	1.55	1.25	1.12

Table 4: List of bids

1. Let $i = 0$, B_{acc} be an (initially) empty set of accepted bids, and r^g be a sorted bid list according to the above mentioned criteria (compare Table 4).
2. Let v be the number at the ith position of the permutation r^g and a_v the corresponding bidder agent.
3. If B_{acc} contains a bid of a_v continue with step 6.
4. If the insertion of the bid with number v in B_{acc} would violate the resource load constraint ($\sum_{i,j} q_{i,j}(r,t)x_{i,j} < q_{max} \quad \forall_{r\in\{1,..,R\},t\in\{1,..,T\}}$) continue with step 6.
5. Insert the bid with number v in B_{acc}.
6. Stop, if $i > L$.
7. Increment i and continue with step 2.

The results of the SG-CAA are used as a benchmark for the algorithms presented in the next sections.

4.2 Simulated Annealing CA Algorithm

Due to its simple functionality the greedy strategy implemented in the SG-CAA is not very promising. One possibility to enhance the efficiency of the greedy strategy is to apply a stochastic improvement process on the initial allocation, trying to remove sub-optimal bids from the AM and to replace them by bids which result in a higher reward for the auctioneer. We want to use a simulated annealing approach, which is based on the original proposal of [15]. The fitness function used is the expected income of the auctioneer. The acceptance probability of a worse solution is controlled by the temperature T.

The simulated annealing CA algorithm (SA-CAA) proceeds as follows: Starting with an empty AM, the auctioneer tries to add a bid which is submitted by an agent that has none of its bids accepted. If the resulting allocation AM violates the resource allocation constraint, the new allocation is discarded and another bid is tried. The fitness of an allocation is the auctioneer's income.

Besides adding and removing bids from the AM, the SA algorithm can also handle both operations simultaneously to obtain a new solution. The probability of accepting a new solution P_{acc} is determined by the *metropolis probability* [16], which depends on the temperature T.

$$P_{acc}(\Delta E) \sim \exp\left(\frac{\Delta E}{T}\right).$$

ΔE denotes the change in the auctioneers income due to the insertion/removal step. The SA-CAA algorithm used for our experiments can be summarized as:

1. Let $i = 0$, B_{acc} be an empty set of accepted bids, and b_{out} and b_{in} be empty.
2. With a probability of 0.25 continue with step 4.
3. Select a bid b_{in} randomly from $B \setminus B_{acc}$. Continue with step 5 with a probability of 0.33.
4. Select a bid b_{out} randomly from B_{acc}.
5. If the insertion of b_{in} and removal of b_{out} from B_{acc} would violate the resource load constraint, continue with step 9.
6. If the insertion of b_{in} and removal of b_{out} from B_{acc} would increase the auctioneer's revenue, or if $random(0,1) \leq P_{acc}(\Delta E_i)$, continue with step 8.
7. Insert b_{in} into B_{acc} and remove b_{out} from B_{acc}.
8. If $i > maxSteps$ stop optimization.
9. If $i \mod N_e = 0$ and thermodynamic equilibrium indicated, decrease $temperature$.
10. Increment i, let b_{out} and b_{in} be empty, and continue with step 2.

When using SA, the starting temperature and cooling rate are essential for the performance of the algorithm. However, optimizing these parameters manually seems to be unpromising. Therefore, we decided to employ a temperature control technique proposed by [17]. This determines the starting temperature such that the algorithm accepts about 80% of exchange operations leading to a deterioration of the fitness function:

$$\sum_{i=1}^{N} \frac{1}{N} \exp\left(\frac{\Delta E_i}{T}\right) = 0.8$$

yielding $T = 5/N * \sum_{i=1}^{N} \Delta E_i$ after *Taylor* expansion.

The annealing process is executed awaiting the occurrence of a thermodynamic equilibrium indicated by a constant $\sum_{i=1}^{N_e} \Delta E_i / N_e$, where N_e is the number of observed steps. The temperature between successive annealing stages is decreased by a cooling factor $\alpha = 0.995$, which is a suitable rule of thumb. The annealing process can be stopped if lowering the temperature does not significantly change the average energy difference.

Figure 1 shows the increasing fitness of the SA-CAA during the annealing process averaged over 50 runs. The regression line has the typical shape of an annealing process. Initially, the temperature is high. Therefore, there is a random walk and no increase in the fitness of the solutions. This is followed by an accelerating, and later decelerating, increase in fitness. Finally, when the temperature is low enough the fitness is fixed in a (local) optimum.

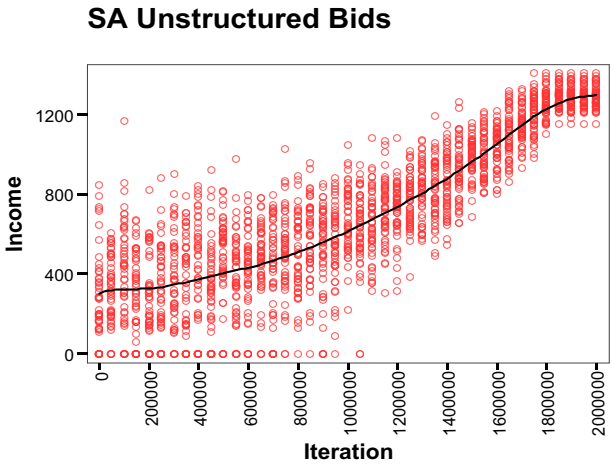

Figure 1: Fitness (auctioneer's income) when using SA-CAA for unstructured bids

4.3 Genetic CA Algorithm

The third approach is a standard generational GA using the random key (RK) encoding. The RK encoding was introduced by [18] and can advantageously be used for ordering and scheduling problems if the relative ordering of tasks is important. Using the RKs for the combinatorial auction problem allows us to incorporate the resource load constraint into the random key encoding and to generate and process only valid solutions. Therefore, no additional penalties for invalid solutions are necessary.

Later, the encoding was also proposed for single and multiple machine scheduling, vehicle routing, resource allocation, quadratic assignments, and traveling salesperson problems [19]. Norman and Bean [20] refined this approach [21] and applied it to multiple machine scheduling problems [22]. An overview of using random keys for scheduling problems can be found in the (unpublished) PhD thesis of Norman [23]. Norman, Smith, and Arapoglu [24, 25] applied random keys to facility layout problems. Rothlauf et al. [26] used random keys for the representation of trees.

The RK encoding uses random numbers for the encoding of a solution. A key sequence of length L, where L is the number of bids $b_{i,j}$, is a sequence of L distinct real numbers (keys). The values are initially chosen at random, are floating numbers between zero and one, and are only subsequently modified by mutation and crossover. An example for a key sequence of length $L = 4$ is $r = (0.07, 0.75, 0.56, 0.67)$. Of importance for the interpretation of the key sequence is the position and value of the keys in the sequence. If we assume that $Z_L = \{0, \ldots, L-1\}$ then a permutation σ can be defined as a surjective function $\sigma : Z_L \rightarrow Z_L$. For any key sequence $r = r_0, \ldots, r_{L-1}$, the permutation σr of r is defined as the sequence with elements $(\sigma r)_i = r_{\sigma(i)}$. The permutation r^s corresponding to a key sequence r of length L is the permutation σ such that σr is decreasing (i.e., $i < j \Rightarrow (\sigma r)_i > (\sigma r)_j$). The ordering corresponding to a key sequence r of length L is the sequence $\sigma(0), \ldots, \sigma(L-1)$, where σ is the permutation

corresponding to r.

We want to give a brief example for the construction of the permutation r^s of bids from the key sequence r. The positions of the keys in the key sequence r must be ordered according to the values of the keys in descending order. In our example we have to identify the position of the highest value in the key sequence (0.75 at position 2). The next highest value is 0.67 at position 4. We continue ordering the complete sequence and get the permutation $r^s = 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$. In the context of our CA problem this permutation can be interpreted as a list of bids that are considered sequentially by the auctioneer. (The auctioneer starts by accepting bid 2, then accepts bid 4, bid 3, and bid 1). From a key sequence of length L, we can always construct a permutation of L numbers (bids). Every bid number between 1 and L (resp. 0 and $L-1$) appears in the permutation only once as the position of each key is unique.

In a next step the accepted bids are calculated from the permutation r^s representing an ordered list of bids:

1. Let $i = 0$, B_{acc} be an (initially) empty set of accepted bids, and r^s the permutation of length L that can be constructed from the key sequence r. All bids $b_{i,j}$ are labeled from 1 to L.
2. Let v be the number at the ith position of the permutation r^s and a_v the corresponding bidder agent.
3. If B_{acc} already contains a bid of a_v continue with step 6.
4. If the insertion of the bid with number v in B_{acc} would violate the resource load constraint, continue with step 6.
5. Insert the bid with number v in B_{acc}.
6. Stop, if $i > L$.
7. Increment i and continue with step 2.

For our experiments we use a simple generational GA [27] with two-point crossover, no mutation, and tournament selection with replacement of size 2. For the encoding of the bids we use the RK encoding. The fitness of the individuals is the resulting income. Due to the use of the RK encoding there are no invalid solutions and no penalties have to be added to the fitness of the individuals.

Figure 2 shows an example of the increasing fitness of the GA-CAA for unstructured bids averaged over 50 runs. We plot the averaged best income of the auctioneer over the number of generations and use a population size of $N = 500$. The plots show a continuous improvement in the income of the auctioneer.

5 Performance of the three CAP solution heuristics

To evaluate the performance of the three solution heuristics we simulated the CAs using the JAVA Agent Development Environment (JADE). All bidders and the auctioneer

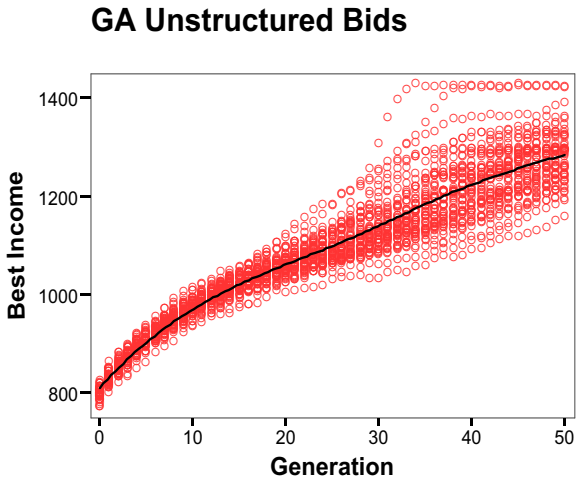

Figure 2: Best fitness (auctioneer's income) in the process of the GA-CAA with 500 individuals for unstructured bids

are implemented as agents. After emitting a call for proposals the auctioneer collects the bids submitted by the agents and calculates the optimal allocation according to the chosen heuristic. The results of 50 simulation runs (auctions) per parameter setting are logged and analyzed using SPSS[3].

Our experimental setting tried to reach the optimal allocation of ISIP tasks in the AM ($q_{max} = 8$) for four basic resources $S = 4$ and $N = 24$ time slots. The maximum resource request per BM of the agents was limited to $q_{bmax} = 3$. Each agent was allowed to submit four bids $J = 4$ of which only one could be allocated (XOR-condition). Bids are generated with time slot occupancy probability $p_{tso} = 0.33$. For the calculation of the bid price we used $p_{min} = 1$ and $p_{max} = 3$. The number of agents varies between $I = 5$ and $= 100$ in steps of 5 for each heuristics type as can be seen in Figure 3.

	bid type	income	time	perf.	stdev.
SG	unstr.	887.28	0.007	1.00	74.86
	substr.	991.82	0.011	1.12	127.89
	str.	1123.04	0.011	1.27	162.74
SA	unstr.	960.22	48.334	1.08	68.15
	substr.	1048.04	84.792	1.18	75.01
	str.	1200.78	84.768	1.35	110.35
GA	unstr.	961.70	9.051	1.08	54.98
	substr.	1082.50	15.067	1.22	63.77
	str.	1269.04	13.455	1.43	89.80

Table 5: Performance of the SG / SA / GA CA-algorithm for 10 agents submitting four bids measured in income and CPU time (sec) for 50 simulations

The plots show that the auctioneer's income increases with a larger number of agents. This can be explained as the probability of finding a compact allocation increases with the number of bids the auctioneer can choose from. More interestingly, the auctioneer's maximal revenue depends on

[3]SPSS Statistical Product and Service Solutions http://www.spss.com

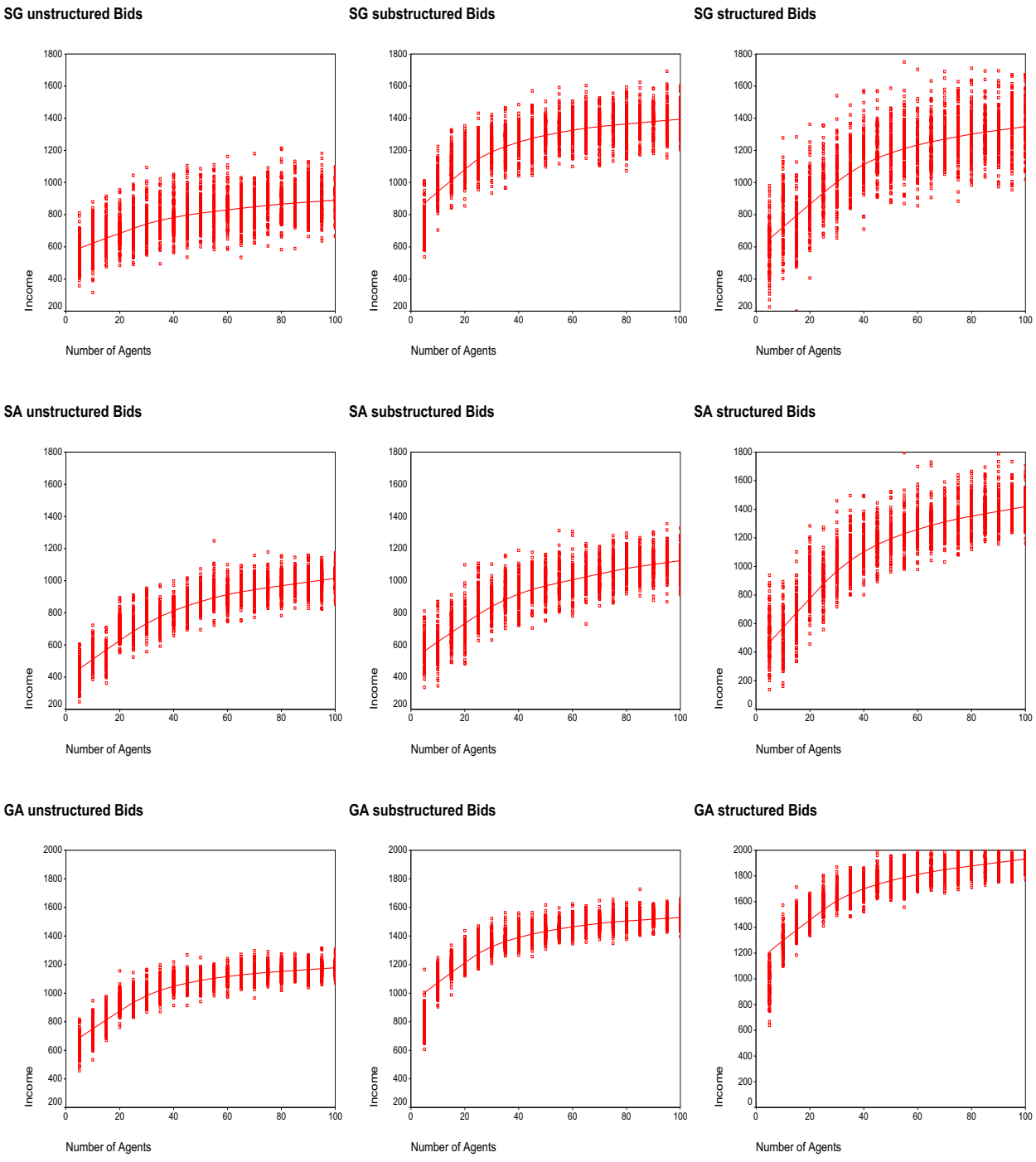

Figure 3: Performance of the SG-CAA, SA-CAA and GA-CAA for unstructured, substructured and structured bids. We show the auctioneer's income over the number of bidding agents.

the structure of the bids and increases from unstructured over substructured to structured bids. A higher structure in the bids allows the auctioneer to allocate the bids more efficiently and to gain a higher revenue.

Table 5 to 7 summarizes the results of Figure 3 and compares the three different optimization heuristics for 10 (Table 5), 50 (Table 6), and 100 (Table 7) bidding agents. Comparing the maximal auctioneer's revenue for different optimization heuristic shows that the SG-CAA is outperformed by the SA-CAA and the GA-CAA. When using the SA-CAA the auctioneer's revenue is about 20% higher in comparison to the SG-CAA. However, computing time in-

creases up to a factor of 10 000. The GA-CAA performs slightly better than the SA-CAA. However, this increase in performance comes with higher computational effort.

	bid type	income	time	perf.	stdev.
SG	unstr.	1084.98	0.049	1.22	92.18
	substr.	1291.08	0.092	1.46	99.94
	str.	1602.72	0.089	1.81	110.11
SA	unstr.	1251.70	55.046	1.41	48.67
	substr.	1447.08	92.369	1.63	62.78
	str.	1718.22	92.481	1.94	76.38
GA	unstr.	1241.42	81.453	1.40	44.57
	substr.	1474.16	139.999	1.66	50.08
	str.	1766.82	132.324	1.99	74.12

Table 6: Performance of the SG / SA / GA CA-algorithm for 50 agents submitting four bids measured in income and CPU time (sec) for 50 simulations

	bid type	income	time	perf.	stdev.
SG	unstr.	1114.36	0.114	1.26	78.93
	substr.	1393.32	0.207	1.57	79.35
	str.	1831.66.	0.208	2.06	116.46
SA	unstr.	1302.32	63.335	1.47	47.61
	substr.	1531.70	100.809	1.73	66.55
	str.	1876.12	100.911	2.11	117.12
GA	unstr.	1282.91	189.512	1.45	55.59
	substr.	1525.80	314.326	1.72	56.67
	str.	1929.76	302.544	2.17	87.44

Table 7: Performance of the SG / SA / GA CA-algorithm for 100 agents submitting four bids measured in income and CPU time (sec) for 50 simulations

6 Conclusions

In this work we developed three heuristics to solve the combinatorial auction problem. The first heuristic is based on a simple greedy allocation mechanism using the ratio of bid price over the overall resource load. The second algorithm employs a simulated annealing procedure to find a high-quality allocation for the CAP making use of a fitness function which is solely based on the auctioneers income. The third heuristic, a genetic algorithm, uses a random key encoding to represent the allocation quality during the optimization process.

Comparing these three heuristics on three different problem instances, which are related to resource allocation tasks in distributed information services and information production environments, reveals that the greedy algorithm shows the lowest performance. Although simulated annealing demands higher computational ressouces, it performs considerably better for the three instances (unstructured, substructured and structured bids) of our test set. The genetic algorithm performs best for the three test types, however requires the highest computational effort. A further result of our experiments is that the allocation quality increases with a higher number of bidding agents and also when using better structured bids.

Bibliography

[1] Jeffrey O. Kephart, Tad Hogg, and Bernado A. Huberman, *The Ecology of Computation*, North-Holland, Amsterdam, 1988.

[2] William E. Walsh and Michael P. Wellman, "A market protocol for decentralized task allocation", in *Revised and extended version of a paper presented at the Third International Conference on Multiagent Systems (ICMAS-98), Paris, France*, 1998.

[3] David C. Parkes, "Optimal auction design for agents with hard valuation problems", *In Agent Mediated Electronic Commerce Workshop at the International Joint Conference on Artificial Intelligence (IJCAI-99)*, 2000.

[4] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham, "Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches", in *Proccedings of the International Joint Conference on Artificial Intelligence 1999 (IJCAI-99), Stockholm, Sweden*, 1999.

[5] Sven De Vries and Rakesh Vohra, "Combinatorial auctions: A survey", *INFORMS Journal on Computing*, 2001.

[6] Vijay Chandru and M. R. Rao, "Integer programming", Tech. Rep. T.R.No. IISc-98-04, Department of Computer Science and Automotion, Indian Institute of Science, Bangalore, India, 1998.

[7] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge, "Integer programming for combinatorial auction winner determination", in *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS '00), Boston, MA*, 2000, pp. 39–46.

[8] Tuomas Sandholm, "Algorithm for optimal winner determination in combinatorial auctions", *Artificial Intelligence*, vol. 135, no. 1-2, pp. 1–54, 2002.

[9] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine, "CABOB: A fast optimal algorithm for combinatorial auctions", in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01), Seattle, WA*, 2001, pp. 1102–1108.

[10] Holger H. Hoos and Craig Boutilier, "Solving combinatorial auctions using stochastic local search", in *Proceedings of the 17th National Conference on Artificial Intelligence (IAAI), Austin, TX*. AAAI, 2000, pp. 22–29.

[11] John Collins, Maria Gini, and Bamshad Mobasher, "Multi-agent negotiation using combinatorial auctions

with precedence constraints", Tech. Rep. T.R.No. 02-009, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN, 2002.

[12] Aneurin Easwaran and Jeremy Pitt, "A brokering algorithm for cost & QoS-based winner determination in combinatorial auctions", in *Intelligent Problem Solving, Methodologies and Approaches, 13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2000), New Orleans, LU*, Rasiah Loganantharaj and Günther Palm, Eds., Berlin, Germany, 2000, vol. 1821 of *Lecture Notes in Computer Science*, Springer Verlag.

[13] P. C. Chu and J. E. Beasley, "A genetic algorithm for the set partitioning problem", Tech. Rep., The Management School Imperial College London, 1995.

[14] David Levine, *A Parallel Genetic Algorithm for the Set Partitioning Problem*, PhD thesis, Argonne National Laboratory, Argonne, IL, 1994.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, pp. 671–680, 1983.

[16] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines", *Journal of Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, 1953.

[17] Erik Sundermann and Ignace Lemahieu, "PET image reconstruction using simulated annealing", in *Proceedings of the SPIE Medical Imaging '95 (Image Processing)*. SPIE, 1995, pp. 378–386.

[18] J. C. Bean, "Genetics and random keys for sequencing and optimization", Technical Report 92-43, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, June 1992.

[19] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization", *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.

[20] B. A. Norman and J. C. Bean, "Random keys genetic algorithm for job shop scheduling", Tech. Rep. No. 94-5, The University of Michigan, Ann Arbor, MI, 1994.

[21] B. A. Norman and J. C. Bean, "Scheduling operations on parallel machines", *IIE Transactions*, vol. 32, no. 5, pp. 449–459, 2000.

[22] B. A. Norman and J. C. Bean, "Operation sequencing and tool assignment for multiple spindle CNC machines", in *Proceedings of the Forth International Conference on Evolutionary Computation*, Piscataway, NJ, 1997, pp. 425–430, IEEE.

[23] B. A. Norman, *Scheduling Using the Random Keys Genetic Algorithm*, unpublished PhD thesis, University of Michigan, Ann Arbor, Michigan, 1995.

[24] B. A. Norman and A. E. Smith, "Random keys genetic algorithm with adaptive penalty function for optimization of constrained facility layout problems", in *Proceedings of the Forth International Conference on Evolutionary Computation*, Piscataway, NJ, 1997, pp. 407–411, IEEE.

[25] B. A. Norman, A. E. Smith, and R. A. Arapoglu, "Integrated facility design using an evolutionary approach with a subordinate network algorithm", in *Parallel Problem Solving from Nature, PPSN V*, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds., Berlin, 1998, pp. 937–946, Springer-Verlag.

[26] F. Rothlauf, D. E. Goldberg, and A. Heinzl, "Network random keys – A tree network representation scheme for genetic and evolutionary algorithms", *Evolutionary Computation*, vol. 10, no. 1, pp. 75–97, 2002.

[27] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, Reading, MA, 1989.