Managing Risk in Recurrent Auctions for Robust Resource Allocation

Victor MUÑOZ and Dídac BUSQUETS
{vmunozs,busquets}@eia.udg.es
University of Girona

Abstract. Auctions can be used to solve resource allocation problems where tasks have to be assigned to resources in such a way that no resource gets overused and an objective function is optimized. In some cases a robust solution is preferable to the optimal solution as it may still be applicable even if unexpected changes in the environment occur. In this paper we present a robustness mechanism for auctions, producing feasible and near optimal solutions even if non-planned events occur. The proposed mechanism has been used in a real problem obtaining successful results.

Keywords. Robustness, auctions, resource allocation

1. Introduction

The problem of *resource allocation* is present in many real-world applications, ranging from assigning memory and computing power to processes, to distributing tasks to machines in a factory, or selecting the personnel to carry out a set of tasks, among many others. This problem is a particular case of Constraint Optimization Problems, in which a set of resources has to be assigned to a set of agents (which represent the entities that need to use the resources). As an optimization problem, the goal of the solvers is to find the optimal solution. That is, the solution has to fulfill a set of constraints (usually regarding the resources), while maximizing or minimizing a given objective function (such as cost, revenue, makespan, etc.). However, sometimes obtaining the optimal solution is not the best choice, since it could fail in case the environment changed (a machine breaking down, a process taking longer than expected, etc.). In such cases, it would be much better to have a *robust solution* that could still be applicable even if unexpected events occurred. Obviously, the price of robustness is optimality [2], since usually a robust solution will be suboptimal. Therefore, there is a need of balancing the optimality and the robustness of the solutions.

In some scenarios the resources are shared and the assignments are only valid for a given period of time, after which the resources must be reallocated. Therefore, the allocation process is continuously repeated over time with the same set of agents. In real environments, for instance an industrial one, the access to the resources is vital for carrying out the production activity of the agents. Thus, it could happen that some agent tries to use more resources than those it was assigned, or even use some that were not

actually assigned to it. This would cause an overuse of the resources, and the rest of the agents would be negatively affected.

One approach to solve this *disobedience problem* is to impose penalties or fines for not abiding by the allocation outcome [12]. However, even with such a mechanism, an agent could still be willing to pay if the benefit of unlawfully using a resource is higher than the penalty for doing so. Thus, in order to prevent conflicting situations where resources are overused, the allocation should be prepared to deal with such unexpected situations. That is, the allocation should be robust to potential disobedience behavior of the agents. However, penalties may be present as well, as otherwise the agents would be always disobeying.

In this paper we focus on market-based mechanisms, namely auctions, for assigning resources to agents. Auction mechanisms, borrowed from the field of Economics, have become a popular approach for dealing with the problem of resource allocation. The advantage of using auctions is that they provide a greater degree of privacy to the agents, since they do not have to reveal too much information, and provide also more autonomy in the decision-making, in comparison with purely centralized solvers. However, the problem of deciding which agents are the winners of the auction, known as the *Winner Determination Problem* (WDP), is also an optimization problem, and therefore, most of the developed algorithms for solving it are again focused on finding optimal solutions instead of robust ones.

Although robustness has already been addressed in the field of planning and scheduling [1,5,3], as far as we know, the only approach that deals with robustness in auctions has been presented in [8]. This work uses the concept of super-soultions [7] to address the problem of bid withdrawal and generate solutions with a repair cost below a given threshold. However, the problem of bid withdrawal is not the only one requiring robust solutions. In particular, the disobedience problem presented above also needs to be dealt with, and the current algorithms for solving auctions do not take it into account. Thus, we have focused our efforts in developing a mechanism for adding robustness to auctions based on learning the obeying behavior of the agents.

2. Market-based Resource Allocation

Resource allocation problems have been usually solved using a centralized approach, where given all the requests, the resources are distributed in a way that there are no conflicts between them. Such centralized approach implies that the central element would made all the decisions. However, such decisions should be made distributedly, since the requesters may not be willing to disclose private information related to their internal functioning upon which their decisions are based. Thus, in order to preserve privacy, a distributed approach is preferable [4]. In a distributed scenario there is a central element representing the shared resources of certain capacities, and a set of individuals competing for the resources.

Auctions provide an efficient mechanism to assign the resources to the requesters. The goal of an auction is to select a subset of the requests, which will gain the right to use some resources for a given time period, while the remaining should wait for another opportunity. The selection criteria is based on the bids submitted by the participants. These bids represent the urgency that each of them has to use the resource. A high bid

indicates that it really needs (or wants) the resource, while a low bid indicates that it could delay the task that requires the resource and therefore it can miss the opportunity to perform it at the auctioned time.

Formally, the problem to solve in an auction where there are multiple resources of different capacities is the Winner Determination Problem (WDP) for multi-unit combinatorial auctions [9] (similar to the multi-dimensional knapsack problem):

$$\max \sum_{i=1}^{NR} x_i \cdot v_i$$
$$\text{s.t.} \sum_{i=1}^{NC} x_i \cdot q_{i,j} \leq Q_j \quad \forall j \in C \tag{1}$$

where NR is the number of requests, $x_i \in \{0, 1\}$ represents whether request i is denied or authorized, $v_i \in I\!R^+$ is the bid value for request i, $q_{i,j}$ is the capacity requirement of the resource j for the request i, Q_j is the resource j capacity, and C is the set of resources.

The auction process is repeated every time a new allocation of resources is needed. This leads to a *recurrent auction*, where the same bidders are continuously competing for the same resources. In this kind of auctions the Bidder Drop Problem comes out: it can cause the participants to stop obeying the outcome of the auction and start behaving on its own, which could conflict with the behavior of the ones agreeing with it.

The bidder drop problem has been typically addressed in the literature using fairness mechanisms [10,11]. However, although fairness incentivizes agents to participate in the auctions, it does not produce robust solutions by itself. Robustness is a desired feature in these situations, as it would produce solutions taking into account those agents which are most likely to disobey the decisions of the auctioneer if unauthorized, thus preventing overuse of the resources.

3. Robustness in Auctions

As mentioned before, in some domains an interesting feature on auctions is to incorporate robustness. It represents the ability of a solution to overcome unexpected changes in the environment. Thus, we are willing to accept a suboptimal solution in order to ensure that the solution remains feasible and near optimal even when the data changes. There are two general approaches for dealing with uncertainty: *proactive* and *reactive*. Roughly speaking, proactive robustness means that the obtained solution is robust by itself, being able to absorb some level of unexpected events, while reactive robustness addresses the problem of how to recover from a disruption once it has occurred, providing an alternative solution in case that the primary solution becomes unapplicable.

We will focus only on proactive robustness, describing how to add a robustness model to recurrent auctions. The robustness model consists in three main components:

- **Trust model** of the agents requesting the resources
- **Risk function** of the agent selling the resources (i.e. the *coordinator*)
- **Robust solution generation**

The first component is concerned with the agents requesting resources. It models their behavior by learning from their actions the circumstances in which an agent does not obey the decisions of the coordinator. Then the coordinator uses these models in order to know in advance which agents are going to disobey if they are unauthorized to perform a task, and act correspondingly. The second component is related to the coordinator and its risk function, as the robustness mechanism varies depending on the risk attitude of this agent. Finally, with the inputs coming from all the agents, the robustness is achieved by combining the risk of the coordinator with the trust on the agents requesting the resources.

3.1. Trust model

An agent requesting resources to perform tasks can disobey the decisions of the auctioneer for several reasons. It is not usually the case that an agent disobeys every decision of the auctioneer independently of the characteristics of the task to perform. Normally, an agent would disobey only the decisions that deny some tasks that it needs to perform for some reason. Therefore the trust model should not contain only a unique global value for the degree of trust of an agent, but the trust value should be related to a given task features, as an agent probably disobeys differently as a function of the characteristics of a task. Therefore, the trust model maintains concrete information about the distinct kinds of tasks, in order to learn which tasks are most likely for the respective agent to be disobeyed in case they were denied. Possible task features to build the trust model with include the resources capacity requirements, the task duration, etc.

The information stored about the trust itself is not only the probability of disobeying, but it is generalized with a lie magnitude, as an agent may request to perform some tasks using a given capacity of resources and later use a higher capacity than requested. Consequently, the trust model also stores for each of the considered characteristics these two trust measures:

- **Probability of disobeying**. This value $\in [0..1]$ can be measured in different ways, being the most intuitive the average of disobediences in relation to the total number of auctions the agent has been involved in. However, it could be measured not only counting the number of times that the agent has performed the task when unauthorized, but counting also the times where the agent has performed the authorized task but using a higher amount of capacity than requested.
- **Lie magnitude**. This value $\in [0..\infty]$ represents the degree of the disobedience. For example a value of 1 would represent that when the agent disobeys, it uses the quantity of resources requested for the task, while a value of 1.5 would represent that it uses the 150% of the requested capacity.

A graphical representation of this trust model using only one characteristic of the task is shown in Figure 1 (to use more task characteristics, additional dimensions would be added). Note that this model is general enough to allow including even the case where an industry does never disobey the auctioneer (it only performs the task when it is authorized to, so it has a disobey probability of 0 for all task characteristics), but it uses a higher amount of capacity than requested (having a lie magnitude greater than 1 at disobey probability of 0). This is particularly useful in problems where the resource capacity requirements of the agents are quite dynamic.

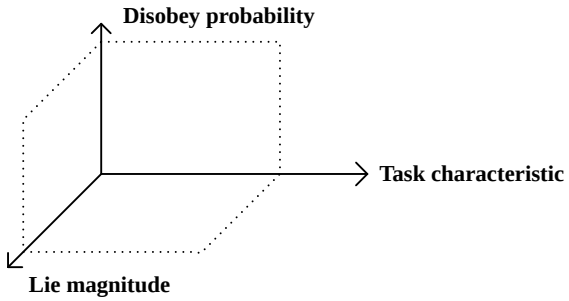

Figure 1. Trust model.

The trust model is learned by the auctioneer agent at execution time. Every time a task is performed the trust model of the respective agent is updated with the new trust values obtained and related to the characteristics of the current task, i.e. if the task has been performed after the authorization of the auctioneer or not (the agent has disobeyed), and checking if the resource capacity used is the same as what was requested. The trust model is also consulted each time an agent requests to perform a task; the auctioneer looks for the trust model of the agents willing to perform a task at a given time and gets their trust values in order to solve the current conflict taking them into account.

3.2. Risk function

The auctioneer's risk attitude characterizes its willingness to face dangerous situations. Risk attitudes are generally categorized in three distinct classes: risk aversion, neutrality and proclivity. Risk aversion is a conservative attitude for individuals who do not want to be at stake. Risk neutral agents display an objective predilection for risk, whilst agents with a proclivity for risk are willing to engage in gambles where the utility of the expected return is less than the expected utility.

To produce a robust solution the risk attitude of the auctioneer is considered together with the trust models of the agents. For example, a risk-averse auctioneer would consider that every request with a probability of disobeying greater than 0 is going to use the resources even if unauthorized, and thus it would auction only the remaining resources capacities over the rest of the requests. On the other hand a risk-proclive auctioneer would consider that if a request has a low probability of being disobeyed, it would not be the case at this time and hence the auctioneer would auction a bigger amount of resources capacities, although with a higher risk of being overused.

The risk function f_{risk} gives the risk attitude of the auctioneer (between 0 and 1) as a function of the probability of disobeying of a given agent and a given request. An example of a risk function is shown in Figure 2(a). In this case it represents a risk-averse auctioneer, since the resulting *risk value* is almost always 1 (it considers risky requests as if they are going to surely use the resources even if unauthorized), regardless of the probability of disobeying. On the other hand, a risk-proclive auctioneer would have the final value almost always set to 0, as seen in Figure 2(b), and a risk-neutral one would have it set accordingly to the probability of disobeying.

We can guess that a risk-averse auctioneer will face fewer overuses of the resources. However, as the agents will have less access to the resources, there will be more requests

delayed and thus the makespan will be longer. Instead, with a risk-proclive auctioneer the makespan will be shorter, although the resource may be overused. However, in the results section we will observe that this fact is not always happening.

3.3. Robust solution generation

Once the auctioneer has defined its risk function and the trust model about the agents performing tasks with different resources requirements has been learned, all of this information can be used to solve any forthcoming conflict in the resources. When a conflict is detected, the auctioneer is faced with a set of requests, each associated with a set of trust features, obtained from the trust model.

Then the auctioneer, taking into account the trust levels associated to each task decides which to authorize and which not in function of its risk. To solve this situation a new constraint, the *robustness constraint*, is added to the constraint optimization problem previously formulated in Equation 1, where we have n variables $X = \{x_1...x_n\}$ (one for each request involved in the conflict), each one representing whether the request is authorized or denied.

The robustness constraint is formulated in a way that the solution finds a balance between the amount of resources required by the authorized requests and the assumed risk from the unauthorized requests (appropriately weighted by its probability of disobeying, lie magnitude and the risk function f_{risk} of the auctioneer). The objective is to not exceed the capacities of the resources (Q_j). This constraint is defined as follows:

$$\sum_{i\in[1,n]} x_i{\cdot}c_i + \sum_{i\in[1,n]} (1-x_i){\cdot}c_i{\cdot}f_{risk}(P_i){\cdot}M_i \leq Q_j \quad \forall j \in C \tag{2}$$

The first summatory represents the resources used by the authorized requests, and the second characterizes the resources potentially used by the unauthorized requests. Then the unauthorized requests are considered as if they were performed in the cases where the probability of disobeying of the associated agent (P_i) is high. However this value (appropriately weighted with the lie magnitude M_i) is considered as a function of the risk attitude of the auctioneer f_{risk}. In this case we have considered that the lie magnitude is directly multiplied by the *risk value*, but another function could be used as well.

Another way of understanding this equation is by moving the second summatory to the right side. Then it can be read as if the total capacities of the resources get diminished in some degree by the unauthorized requests that are likely to be performed anyway. Then the requests are auctioned normally although with less resources capacities available.

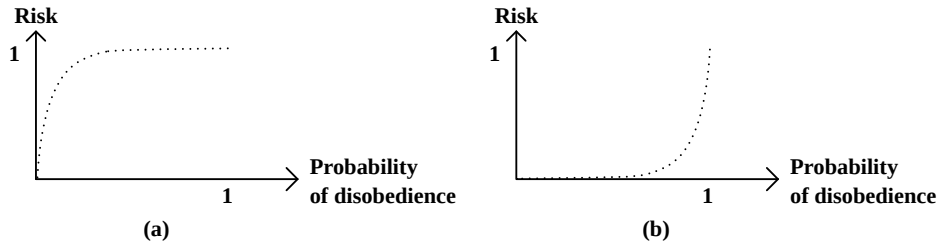

Figure 2. Risk attitude function: (a) averse, (b) proclive.

4. Experimentation

To test the robustness mechanism previously described, we have used a real-world problem: The Waste Water Treatment Plant Problem (WWTPP). The main components in this problem are the Waste Water Treatment Plant and the set of industries performing waste discharges to the sewage. The job of the treatment plant is to process the sewage coming from the industries, removing its contaminants in order to return a clean waterstream back to the river. The problem faced in this domain is to coordinate the industrial discharges so that all the polluted water entering the plant can be fully treated. If the discharges are done without any coordination, the amount of water arriving at the plant may exceed its capacity, which causes the overflow to go directly to the river without being treated and increasing its contamination. Thus, in order to prevent such dangerous environmental situation, the industrial discharges should be temporally distributed so that all of them can be fully treated.

Obviously, this coordination should not cause problems in the production processes of the industries, since this could have dangerous effects (drastic changes would cause production delays, missed delivery commitments and so on). However if the discharges can not be delayed, there is no coordination possible. Therefore, we assume that each industry has a retention tank (of a given capacity) where it can store a discharge whenever it is not authorized, and empty it later on. In this scenario the recurrent auction mechanism will determine which discharges to authorize and which to be temporarily stored in the tank in order to not exceed the plant's capacity.

Regarding the robustness mechanism proposed in this paper, it is easier to understand more clearly with this concrete problem why is it useful. In this scenario it is conceivable that industries may sometimes disobey the decisions of the plant. The most obvious reason is when an industry has its retention tank completely full; in this case if the forthcoming discharge is not authorized, the industry will be forced to discharge it anyway, thus disobeying the plant. However, an industry could disobey the decisions of the plant for other uncontrolled and unpredictable reasons, for example when the industry needs for some reason to have the retention tank empty (for maintenance purposes, for instance), or when a concrete discharge cannot be stored in the tank because of its high level of contamination, etc. That is the reason why the robustness mechanism has been designed to take into account the characteristics of the task in the trust model.

4.1. Solving the WWTPP

The WWTPP can be modeled as a recurrent combinatorial auction, where the auctioneer is the treatment plant, the resource being sold is its capacity, and the agents using the resource are the industries that perform discharges. Every discharge is defined as $D_i = \{industry\_id_i, s_i, d_i, q_i, \overline{c_i}\}$, where s_i and d_i are the start time and the duration of the discharge, and q_i and $\overline{c_i}$ are the flow and contaminant levels of the discharge. In this case the resource consumption (as well as the individual discharges) does not have only a global capacity limit (hydraulic capacity), but it is extended with many thresholds, one for each contaminant type. Then the goal of the auctioneer is not only to not exceed the hydraulic capacity of the plant, but also to have each of the contaminant levels under its thresholds. To this end the discharge flow as well as each of the contaminants are considered as separated resources.

With these adjustments, the coordinating scenario described in the previous section can be easily adapted to be applied to this problem, so the robustness mechanism can also be used. Here the disobeying probability can be defined as a function of the characteristics of the discharge (or the industry), for example:

- The flow of the discharge.
- Duration.
- Volume (amount of liters of the discharge).
- Contaminant levels.
- Retention tank occupation.

4.2. Implementation

To evaluate the coordination and the robustness mechanisms we have implemented a prototype of the system that reproduces the coordination process and the communication between the plant and the industries performing discharges. We have created an agent to represent the plant and another one for each one of the industries. So far we have only considered the hydraulic capacity.

To calculate the bid, each industry agent takes into account the urgency for performing the discharge, based on the retention tank occupation. Thus, the bid value of agent i, v_i, is computed as:

$$v_i = \frac{\text{tank occupation}_i}{\text{total tank capacity}_i} \tag{3}$$

In case an industry agent has to reschedule its discharges, its behavior is the following: it first tries to store the rejected discharge into the retention tank; the discharge of the tank is then scheduled as the first activity of the agent after the current conflict finishes. Otherwise, if the industry has its tank already full, the discharge will be performed anyway.

The objective function to maximize in the auction clearing is the sum of the winning bids values. The free linear programming kit GLPK [6] has been used to solve the winner determination problem appropriately modeled as a mixed integer programming problem (MIP). The robustness constraint is added to the constraint optimization problem as an additional constraint.

The trust models of the industries have been implemented using only one characteristic of the discharges: the flow. The models of the industries are learned during the execution by storing, for each different value of flow of a discharge from an industry, two counters for the total number of lies and truths (that is, disobedient and obedient actions), and another value to compute the lie magnitude. These values are updated after each performed discharge in the following way: if the respective industry was authorized by the plant, then the number of truths of the corresponding flow is incremented; alternatively if the performed discharge was not authorized, then the number of lies is incremented. Independently, the value regarding the average lie magnitude (of this concrete flow) is updated with the lie magnitude of the current discharge computed as the division between the used capacity in relation with the requested capacity.

4.3. Experimentation results

In order to evaluate the results we have considered some quality measures based on different characteristics of the solution:

- **number of overflows (NO)** occurred during the simulation
- **maximum flow overflowed (MFO)**, measured in m^3/day
- **total volume overflowed (VO)**, in liters
- percentage of discharge denials **obeyed** by the industries **(%IO)**

The experiments consisted of simulations using a set of real data provided by the Laboratory of Chemical and Environmental Engineering (LEQUIA). This data is composed of the discharges of 5 industries in two weeks. The first one is a pharmaceutical industry; it is increasing its discharge flow during the week and does not discharge during the weekend. The second one is a slaughterhouse that discharges a constant flow, except at the end of the day when it increases. The third one is a paper industry that discharges a constant flow during the seven days of the week. The fourth one is a textile industry, whose discharges flow oscillates during the day. The fifth one is the waste water coming from the city, whose flow is fixed. The hydraulic capacity of the plant is 32000 m^3/day.

We have tested the mechanism in different scenarios and situations. We have experimented with and without coordination among the industries (without coordination the treatment plant does never unauthorise a discharge), activating and deactivating the robustness mechanism, and also changing the obeying behavior of the industries. There are scenarios where all the industries always obey the decisions of the plant (as long as they have enough tank capacity), and other scenarios where the industries have some probability of disobeying the outcome of the coordination mechanism (this probability depends on the occupation of the tank; the higher the occupation, the higher the chances of disobeying), and also scenarios where there is one industry (the textile, chosen randomly) that is always disobeying the decisions of the plant.

		NO	**MFO**	**VO**	**%IO**
No coordination		80	9826	$15.21{\cdot}10^6$	-
Obey	No Rob	28	4996	$3.74{\cdot}10^6$	98.95
	Rob	28	4996	$3.74{\cdot}10^6$	98.95
Disobey	No Rob	113.40 (7.55)	14357 (1077.02)	$13.4{\cdot}10^6$ (319429)	97.23 (0.21)
	Rob	121.3 (7.94)	14233 (1358)	$13.2{\cdot}10^6$ (374673)	96.58 (0.41)
TEXTILE INDUSTRY ALWAYS DISOBEYING					
No coordination		80	9826	$15.21{\cdot}10^6$	-
Obey	No Rob	112	6523	$6.89{\cdot}10^6$	90.84
	Rob	58	6590	$5.47{\cdot}10^6$	96.77
Disobey	No Rob	119.70 (4.72)	14819 (1373.74)	$14.3{\cdot}10^6$ (263955)	89.96 (0.28)
	Rob	109.50 (3.95)	14150 (1310)	$13.6{\cdot}10^6$ (242619)	95.19 (0.17)

Table 1. Simulation results.

We can observe in Table 1 the outcome of all these scenarios. First of all we can notice that the scenario without any coordination produces the worst results regarding volume overflowed, therefore this is the worst possible scenario as this concrete indicator is the most important one. When adding the auction-based system the results are highly improved, principally when all the industries obey the decisions of the plant. The obeying scenario reflects the best possible circumstances for the problem, however the problem has been tested with more adverse situations in order to better evaluate the robustness mechanism.

When all the industries disobey as a function of its tank's occupation we can notice a subtle improvement by using the robustness mechanism in both the volume and the maximum flow overflowed. However the difference is not much relevant, and the number of overflows is higher within the robust approach.

On the other hand, in the environment where there is one industry always disobeying the plant's decisions regardless of the disobeying function, the robustness mechanism seems to mark differences, both in the obeying and the disobeying scenarios. All the indicators are significantly improved when using it, specially regarding the volume overflowed and percentage of obedience.

5. Conclusions and Future Work

In this paper we have presented a robustness mechanism for auction-based resource allocation problems. Through this mechanism, the system finds a solution that is robust, i.e. it is able to remain applicable even if there are changes in the environment. Changes involve both modifications on the resources capacities requests and the use of the resource even when the user is not authorized to. The core of the robustness mechanism consists in a trust model that is learned during the execution and a risk function associated to the auctioneer of the resources, that are used together in order to produce a robust allocation.

The results obtained through simulation using real data show that the robustness mechanism improves the results over the non-robust approach. However, further work has to be made in the different risk attitudes of the auctioneer as we have not noticed significant changes when varying its risk attitude from risk-averse to risk-proclive. Also the trust model needs to be improved as it considers tasks with different characteristics independently, therefore in problems where the tasks characteristics were too dynamic it would be useless as there would not be two identical tasks.

It should be noted that the robustness mechanism may induce the agents to disobey, as doing so they are going to be always authorized by a risk-averse auctioneer. To avoid such a situation another mechanism should be incorporated to the system. Different mechanisms to achieve that have already been studied, as for example the addition of *fines* (or *penalties*) to be paid whenever an agent does not obey the decision of the coordinator; another method would be to stipulate a deposit to be paid for the participants before beginning the coordination, and returned later only to the ones that have obeyed the coordinator. However, the price of these fines or deposits should be studied in more detail in order to make it not too cheap so an agent would prefer to pay it instead of obeying the coordinator, neither too expensive so that a poor agent would have more problems than a rich one to pay it.

References

[1] A. Ben-Tal and A. Nemirovski, 'Robust solutions of uncertain linear programs', *Operations Research Letters*, **25**, 1–13, (1999).

[2] D. Bertsimas and M. Sim, 'The price of robustness', *Operations Research*, **52**(1), 35–53, (2004). 8.

[3] Jürgen Branke and Dirk Christian Mattfeld, 'Anticipation and flexibility in dynamic scheduling', *International Journal of Production Research*, **43**(15), 3103–3129, (2005).

[4] Y. Chevaleyre, P.E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J.A. Rodríguez-Aguilar, and P. Sousa, 'Issues in multiagent resource allocation', *Informatica*, **30**, 3–31, (2006).

[5] A.J. Davenport, C. Gefflot, and J.C. Beck, 'Slack-based techniques for robust schedules', in *Proceedings of the Sixth European Conference on Planning (ECP-2001)*, pp. 7–18, (2001).

[6] GLPK. GNU Linear Programming Kit, http://gnu.org/software/glpk.

[7] Emmanuel Hebrard, Brahim Hnich, and Toby Walsh, 'Super solutions in constraint programming', in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Lecture Notes in Computer Science, 157–172, Springer, (2004).

[8] Alan Holland and Barry O'Sullivan, 'Truthful risk-managed combinatorial auctions', in *Proceedings of IJCAI'07*, pp. 1315–1320, (2007).

[9] J. Kalagnanam and D. Parkes, *Handbook of Supply Chain Analysis in the E-Business Era*, chapter Auctions, bidding, and exchange design, Kluwer Academic Publishers, 2005.

[10] Juong-Sik Lee and Boleslaw K. Szymanki, 'A novel auction mechanism for selling time-sensitive e-services', *Proc. 7th International IEEE Conference on E-Commerce Technology (CEC'05), Munich, Germany, July 2005, pp. 75-82.*, (2005).

[11] Víctor Munoz, Javier Murillo, Dídac Busquets, and Beatriz López, 'Improving water quality by coordinating industries schedules and treatment plants', in *Proceedings of the Workshop on Coordinating Agents' Plans and Schedules (CAPS)*, ed., Mathijs Michiel de Weerdt, pp. 1–8. IFAAMAS, (may 2007).

[12] Tuomas Sandholm and Victor Lesser, 'Leveled commitment contracting: A backtracking instrument for multiagent systems', *AI Magazine*, **23**(3), 89–100, (2002).