A Multi-robot Auction Method to Allocate Tasks with Deadlines ⋆

José Guerrero, **Gabriel Oliver**

Universitat de les Illes Balears,
Mathematics and Computer Science Department, Palma de Mallorca,
Spain, e-mail:{jose.guerrero, goliver}@uib.es

Abstract: Task allocation is one of the main problems in multi-robot systems, very especially when the robots form coalitions and the tasks to execute have to be carried out before a deadline. In general, the time required by a coalition to finish a task can be very difficult to find because it depends, among other factors, on the physical interference. This paper presents an extension of our previous auction method using a new concept called two round auction. In this framework the robots learn the interference and therefore, the coalition's utility, from their past experience using an on-line support vector regression method (SVR). We will show how the performance of the system can be improved if the interference impact is included in the model, and how the second round auction increases the total utility. This method has been tested using transport like tasks.

Keywords: multi-robot, cooperation, interference, auction, learning

1. INTRODUCTION

Multi-robot systems can provide several advantages over single-robot systems: robustness, flexibility and efficiency among others. To benefit from these potential aspects, several problems have to be solved. Among all these problems, we focus on task allocation issues, that is, selecting the best robot or robots to execute a task. Some tasks require that two or more robots cooperate to execute them creating coalitions. In this case, we have to calculate the utility of the coalition. Moreover, when the tasks must be executed before a deadline we also need to know how much time will the coalition need to finish the task. As it happens in a lot of real environments, like in a rescue scenario, when a task can not be executed before its deadline, the utility decreases following a time function.

A lot of research has been done to solve the task allocation and coalition formation problems but they are still open problems. One of the most used and well studied task allocation solutions are auction methods Gerkey and Mataric (2002); Dias and Stentz (2003). Only a few auction strategies, like Vig (2006), allow to allocate several robots to the same task, but these methods don't take into account the interference effect. E. Jones proposed in Jones (2009) a set of auction based methods to assign tasks to robots with intra-path constraints, that is, when the actions of a robot affect the planning of others robots. Anyhow, these systems don't take into account the physical interference between agents.

As it has been demonstrated in different studies Lerman and Galstyan (2002); Rosenfeld et al. (2005), the physical interference has an important impact on the system performance. This effect has a dramatic impact when deadlines are given, because the interference modifies the time needed by the coalition to finish the task, and therefore the coalition utility. Here, we extend our previous auction method Guerrero and Oliver (2007) using a new concept called two round auctions. This new concept helps us to calculate the appropriate number of robots in a coalition to meet the task's deadline. We also use in this work for the first time an on-line support vector regression method (SVR) Ma et al. (2003) to predict the coalition's utility and therefore the time needed to execute a task. This learning method has been used by other authors Jones et al. (2006) within an auction process, but in a very different framework. As far as we know, this is the first time that an auction method models the utility of a robots coalition when they have to maximize whatever time utility function associated to a task. Moreover, The the method can also determine the coalition size.

Finally, another problem that this paper analyzes is how does the monitoring of the task progress affects the system performance. The monitoring process can be a complex task which requires sophisticated sensorial and communication resources. As it will be seen, if the SVR models correctly the interference, then we can predict the execution time and, therefore, the monitoring process is not needed.

The rest of this paper is organized as follows: section 2 presents a formal definition of the problem to solve; section 3 explains the two round auction task allocation algorithm; section 4 explains the SVR technique that has been used; section 5 presents how to use SVR to model the physical interference effect; section 6 shows the results of the experiments and, finally, section 7 explains some conclusions and the future work of our research.

⋆ This work has been partially supported by project DPI2008-06548-C03-02 and FEDER funding.

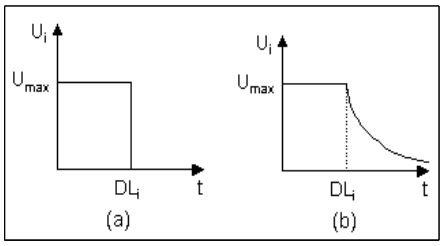

Fig. 1. Examples of utility functions

2. PROBLEM STATEMENT

In this section, we will formalize the task allocation problem previously sketched and we will explain the main problems that it presents.

The task allocation is defined as follows: We want to allocate a set of tasks to a set of robots. Each task t_i has a workload ($taskWorkLoad_i$) that represents the amount of work required to finish the task. For example, if the robots must transport an object, this value will be the weight of that object, if the robots must clean a surface, the workload will be the area to clean. Moreover, each task has associated a utility, and a deadline time DL_i. DL_i is the time instant before which the task must be finished. If the execution time of a task exceeds its deadline, its utility decreases following a certain function, called U_i. Figure 1 (a), show an example of a hard deadline environment, where the utility function drops to zero if the task cannot be executed on time. On the other hand, 1 (b) shows a situation where the task always has a positive utility, whatever the execution time is. Therefore, the goal of the system will be maximize the sum of utilities of the tasks regardless of which function is used, that is, it maximizes the following equation:

$$\sum_{1 \leq i \leq M} U_i(t_i) \tag{1}$$

where M is the number of tasks and t_i the time required to finish the task i. As it can be seen, equation 1 doesn't maximize the number of finished task but the total utility. Besides, each robot (r_i) has an individual work capacity ($workCapacity_i$) that represents the amount of work that the robot can process per time unit. The tasks can be executed by a group of robots, when they form a coalition. Thus, we have to know if the group of robots can fulfil the deadline, and therefore the utility that these robots will generate at the end of the task. To that end, we need to calculate the work capacity of the group as a whole ($groupCapacity$), that is, the amount of work that the group can perform per time unit. In general, it is not true that the work capacity of the group is the sum of the work capacity of each single robot. Thus, the real value of the group capacity is:

$$groupCapacity = idealCapacity - I \tag{2}$$

Where $idealCapacity$ is the model of the group capacity without interference and I is the interference factor. Although this is a simple model, it's very efficient, as it will be shown later. The $idealCapacity$ can be easily represented when the individual capabilities of the robots are known. As it has been explained before, the sum of the individual utilities of each robot is often used. Usually, these utilities are independent of the interference factor and therefore they can be added. On the other hand, I is not easy to calculate because it depends on some dynamic factors like the number of robots or the geometry of the environment.

3. TASK ALLOCATION

The task allocation mechanisms, including the groups' formation, membership policy and task assignment is described in the following paragraphs. This new mechanism extends and improves our previous work Guerrero and Oliver (2007) to take into account whatever utility function is used. To achieve this goal, the two round auction method, is executed. The two round auction algorithm splits a classical auction process into two steps: "Auction for a Task" and "Auction for a Robot".

3.1 Auction for a Task

During the first auction round a classical auction method has been modified to select which robots, and very specially, how many of them are needed to execute a task. Here, in an initial stage, each robot is looking for a task. When a robot finds a new task, it will try to lead it. There is only one leader for each task. The details about how a robot can be promoted to leader, can be found in Guerrero and Oliver (2004). If a robot is promoted to leader, it will create, if necessary, a work group; that is, a set of robots that will cooperate to execute a specific task. In that case, the leader must decide which the optimum group size is and what robots will be part of it. To take this decision, the leader uses an auction like mechanism. During this process the leader will be the auctioneer and the other robots will bid using their work capacity. Thus, this value is the utility function of the auction method or the price that the robots want to pay to participate in the task. The leader selects the robots with the highest work capacity using a greedy algorithm, until it detects that the group is able to reach its deadline, that is, until this condition is verified:

$$DL_g = \frac{taskWorkLoad}{groupCapacity} \leq DL_i \tag{3}$$

where DL is the deadline of the task i. As it can be seen, DL_g is the expected time required to finish the task. Thus, the expected utility of the task can be calculated using the utility function and the expected time. The leader doesn't include any selected robot in its group until it receives the confirmation from these robots. This confirmation will be produced during the auction for a robot round, that will be explained in the next section. Besides, the leader will collaborate on carrying out the task with the other robots of the group. The complexity of the auctioneer algorithm is $O(log(n))$ for sorting the received bids plus $O(n)$ for selecting the best robots, where n is the number of bids. The bidder only have to calculate its bid value, thus its algorithm complexity is $O(1)$.

3.2 Auction for a Robot

Whether the deadline has been fulfilled or not, after the auction for a task round, the leader starts the "auction

for a robot" process, thus multiple auction processes are performed in parallel. To get the selected robots, the leader will bid for them sending the expected utility of the task. Thus, now the leader becomes the bidder and the selected robots are the auctioneers. When a robot receives a bid from a leader, it waits a fixed amount of time for more bids from other leaders. After this time, the robot selects to become a member of the coalitions with the greatest bid and sends a confirmation message to the corresponding leader. On the one hand, if the leader doesn't receive any answer from a robot after a time or receives a rejected message, then it removes it from the list of selected robots. On the other hand, a robot is definitely added to the group, when the leader receives a confirmation message from it. Followig the same reasoning used in the auction for a task stage, now the complexity of the bidder algorithm is $O(log(n))$ for sorting the "bids" recieved from the autioneer, plus $O(n)$ for selecting the best group. The autioneer only have to recieve the message from the bidder, and therefore its complexity is $O(1)$.

If during the task execution the leader detects that the deadline (DL_g) can not be fulfilled, it starts a new two auction process to get new robots. From now on, we will consider the robot utility and the group utility synonymous of robot's work capacity and work capacity of the group, respectively.

4. SUPPORT VECTOR REGRESSION

This section describes the process to predict or to learn the value of the interference using support vector regression (SVR) C.Chang and Lin (2001). The goal of this method is to find a function, $f(x)$, as flat as possible that fits a given set of training data (x_i, y_i), $i = 1..l$ where $x_i \in R^n$ represents the input data of $f(x)$ and $y_i \in R$ are the results. From this data, the function $f(x)$ is:

$$f(x) = \sum_{1 \leq i \leq l} (\alpha_i - \alpha_i^*) K(x, x_i) + b \quad (4)$$

where α_i, α_i^* are the lagrange multipliers, $K(x, x_i)$ is a kernel function and b is the bias. During our experiments the radial basis function, shown in equation 5, is used as kernel:

$$K(x, x_i) = e^{-\gamma \|x - x_i\|^2} \quad (5)$$

where γ is a parameter of this kernel.

As it is shown in subsection 5.2, the x_i vectors represent the coalition characteristics including both coalition robot and environment features. From this point forward, we will call to x_i the coalition vectors. The SVR learning can be off-line (batch) or on-line, both strategies have been implemented in our system, as it will be explained during the following subsections.

4.1 Batch Support Vector Regression

During off-line or batch SVR, all the samples of the training set are known before the execution and the set can be trained at once. The model created with these samples will be used during all the execution. To implement the batch support vector regression we have used the *libsvm* library developed by National Taiwan University C.Chang and Lin (2001). This library creates a model file from a given set of training data. We use the $\varepsilon - SVR$ method and a radial basis function as the kernel function.

E. Jones in Jones et al. (2006) also applies the *libsvm* library to allocate tasks in the Robocup Rescue Simulation League but in that case, only a single robot can be assigned to the same task. Moreover, Jones' method only tries to find the individual utility function for each agent taking into account the expected incoming tasks and their deadline.

4.2 On-line Support Vector Regression

In an on-line SVR learning, new data can be added or removed during the execution of the task. Each time that a new sample (x_i, y_i) is added, the SVR is trained again using this new data. J. Ma et al proposed in Ma et al. (2003) an accurate on-line SVR method, where each time that a new data is added or removed, the SVR function's lagrange multipliers are modified to ensure that the full system verifies the Karush-Kuhn-Tucker (KKT) conditions. Thus, if the KKT conditions are verified, we can ensure that the error of the SVR function is at a minimum. In our system, we have used the implementation of Ma's method done by J. Parrella in Parrella (2007). The experiments show that this method is faster than the *libsvm* when the samples arrive one by one.

In our online-SVR method, each time a task is finished, the leader gets the difference between the "real" execution time and the execution time calculated only with the $idealCapacity$ of equation 2. The interference value I can be calculated from this difference, using the coalition and environment characteristics as the new training vector. This vector will be sent to the other robots, creating a global SVR model. The complexity of this algorithm is, in the worst case, $O(v^3) * O(kernel)$, where v is the number of support vectors and $kernel$ is the time needed to calculate the kernel value. In our SVR implementation v is always lower or equal to 100.

5. PHYSICAL INTERFERENCE IN TRANSPORT TASKS

In this section we will study how to model the physical interference using SVR in a specific transport like task. This task is described as follows: some randomly placed robots must locate objects, randomly placed too, and carry them to a common delivery point. Each object to gather has a weight and each robot has a load capacity. This weight is the $taskWorkLoad$ of equation 3. The robot load capacity is the amount of weight that it can carry at once. Thus, if a robot cannot carry the entire object at once, it takes a part of it, goes to the delivery point and comes back to the object for more bits. Of course, this is a very simple environment but it allows us to isolate the interference effect from other factors that can appear in more complex tasks.

In the next subsections we will describe how to get the individual utility for each robot and how the $idealCapacity$ of equation 2 is calculated.

5.1 Individual Utility Function

We will now describe how to find the individual utility or the individual work capacity of each single robot. The transport task explained earlier will be used as an example. The work capacity of a robot is the amount of object's weight that this robot can transport to the delivery point per time unit. Under ideal conditions, that is, assuming an open environment without any obstacle or robot between the object and the delivery point, the robot's work capacity is easy to calculate. Thus this capacity is:

$$workCapacity_i = \frac{C_i V_i}{2(C_i V_i + d)} \quad (6)$$

where C_i is robot's load capacity, V_i its maximum velocity and d is the distance between the object and the delivery point. In Guerrero and Oliver (2007) the reader can find details about how to get these values. Thus, the *idealCapacity* of equation 2 will be equal to the sum the $workCapacity_i$ of each member of the group.

5.2 Interference Effect: off-line and on-line SVR

In this section we will analyze how to create the off-line SVR model to learn the physical interference between robots, that is the I value of equation 2. To analyze the interference effect we have executed a task where several robots must transport parts of a single object and the total weight transported by the robots after 40.000 time units is calculated. All the robots have the same load capacity (2 weight units) and the same velocity (3 distance units/time unit) and therefore, they all have the same work capacity. Moreover, the environment doesn't have any obstacle but the robots, the object and the delivery point. Eight different distances between the object and the delivery point have been tested and the number of robots varied from 1 to 8. All these experiments have been executed using a multi-robot simulator called RoboCoT. RoboCoT is a realistic simulator developed by the authors at our university. The details of this results can be found in Guerrero and Oliver (2007). Knowing the *idealCapacity* we can calculate the value of interference I for each situation. This value only takes into account the interference between robots of the same working group. All this information is now included in the training data of the SVR method. The coalitions vectors (x_i values) will include 2 features: the *idealCapacity* value, calculated using equation 6, and the number of robots of the group. The calculated values of I are used as the expected results of $f(x)$ (y_i). A total of 80 training data pairs (x_i, y_i) has been used. All the information is used by the *libsvm* library to create the model of the system.

During the on-line SVR learning process, the robots start the execution with the training set used during the off-line SVR. When a new task is finished, its leader adds the information of this task to the set of current samples and executes the on-line SVR algorithm Ma et al. (2003). The number of SVR's samples is increased through the execution and, therefore, the time needed by the online SVR is greater. To avoid an excessive number of samples its has been limited to 120. The experiments show that this number is a good tradeoff between execution time and system performance.

Whatever method is used, on-line or off-line SVR, the leader selects the best robots (robots with the highest bids) until equation 3 is verified or until no other robot can increase the group capacity. During the execution of a task the leader can periodically receive information about the remaining weight of the object ($taskWorkLoad$) to be transported. If available, the leader of the task uses this information to make a guess about the actual $taskWorkLoad$.

5.3 The Monitoring Process

During the execution of a task the leader can periodically receive information about the remaining weight of the object ($taskWorkLoad$) to be transported. If available, the leader of the task uses this information to make a guess about the actual $taskWorkLoad$ using a simple linear equation:

$$WL(t) = WL(t_m) - groupCapacity * (t - t_m) \quad (7)$$

where t_m is the time when the leader receives information about the task progress and $WL(t)$ is the expected $taskWorkLoad$ at instant t.

During a continual monitoring task progress execution, the leader knows in each moment the exact value of the $taskWorkLoad$, and therefore, it can start another auction process if inequality 3 is not verified. In a non-monitoring task process execution, the leader only knows the $taskWorkLoad$ at the beginning of the task, and then it uses equation 7 to predict the $taskWorkLoad$, with a unique constant $WL(t_m)$ during the whole process.

6. TASK ALLOCATION EXPERIMENTS

In this section we will show the results of several experiments carried out to study the impact on the system performance of: the two round auction process, the physical interference model, the monitoring process and the on-line SVR learning. During all the experiments, the simulator RoboCoT has been used. The robots must execute the transport task explained in the section 5. The main objective is to maximize the equation 1. All the objects must be transported even if they are over their deadline and their utility have become zero. The time to deadline starts when the object appears in the environment. To simplify the analysis, the robots know the situation of each object in the environment.

6.1 Two rounds auction experiments

In this subsection we will explain the experiments carried out to show how two rounds auction methods can improve the system performance.given by equation 1. Six different kind of experiments have been carried out:

- Monitoring and interference (M_I): the continual monitoring strategy is used together with our interference model.
- No Monitoring and interference (NM_I): in these experiments our interference effect is used, but without monitoring.

- Monitoring and no interference (M_NI): the continual monitoring strategy is used but without the interference method. That is, the expected execution time is calculated using only the ideal capacity.
- No Monitoring and no interference (NM_NI): neither continual monitoring or our interference model are used.
- Single round auction (SR): in this strategy the leader only uses the "auction for a task" method. Then, the robots select the first agreement message received from any leader.
- Greedy selection: all the leaders try to create a working group as great as possible without taking into account the deadline value or the task characteristics. Thus, this strategy is like a SR auction but now the leader sends an agreement message to all the bidders.

Both Greedy and SR methods only use the auction to robot round, and M_I, NM_I, M_NI, NM_NI implement the two rounds method. During the experiments we used 10 robots and 3 objects to gather. All the tasks have a weight equal to 40 weight units. Robots must transport 400 objects and the total time needed to transport each one of these objects is calculated. Despite having only 3 objects in the environment, when an object is fully transported to the delivery point, it immediately appears another one in a random place. All the tasks have a deadline equal to 900 time units, but its maximum utility value can change. Finally, two utility functions have been used: a hard deadline and the following soft deadline function:

$$U_i = \begin{cases} U_{max} & \text{if } t < DL_i \\ U_{max} \dfrac{0.07DL_i}{(t_i - DL_i) + 0.07DL_i} & \text{otherwise} \end{cases} \quad (8)$$

where U_{max} is the maximum utility of the task, DL_i is the deadline of this task and t_i is the time required to finish.

In all theses cases we have used the off-line SVR strategy. The off-line SVR model only considers the intra-group interference, that is, the physical interference between members of the same group. The interference, produced between members of different groups, has not been included. The analysis of the results shows that the inter group interference in this specific environment increases about 8% the modelled one. Thus, during the experiments carried out to get the results of the following figures, the interference obtained from the model has been increased this percentage. This overhead percentage has been calculated from the difference between the simulations results and the expected time of equation 2.

Figure 2 shows the total utility when the soft deadline function is used. This utility has been calculated using the equation 1. As it can be seen, the M_I and NM_I strategies present very similar results, therefore, it can be stated that our interference model correctly predicts the evolution of a task without any monitoring process. Moreover, in both cases the total utility has been increased about 40% with regard to the greedy method. In addition, this figure shows the impact of the interference on the system. Thus, the results of the methods that don't use the interference model (M_NI and NM_NI) and the results of the greedy system are very similar. Finally, the utility produced by the robots that use the single round auction

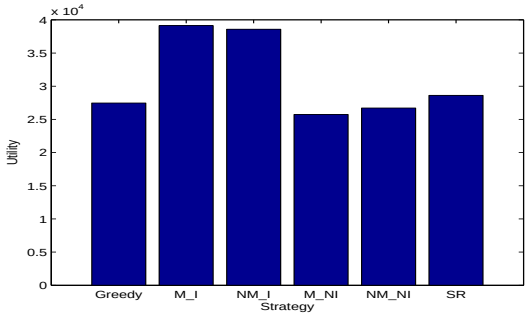

Fig. 2. Total utility with the soft deadline function.

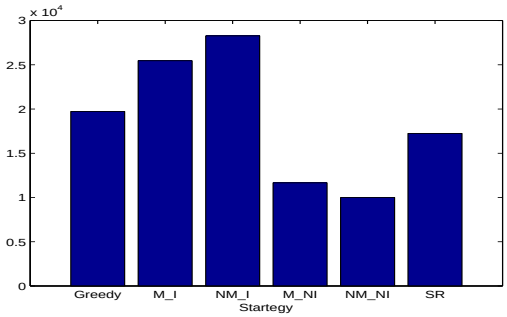

Fig. 3. Total utility with the hard deadline function.

strategy is also very similar to the greedy experiments results. Therefore, the "Auction for a robot" round clearly increases the system performance.

The results of the experiments with the hard deadline function are shown in figure 3. The NM_I presents the best results, increasing about 43% the total utility of the system when compared to the greedy strategy. Moreover, the non-monitoring strategy outperforms the monitoring methods.

Figure 2 also shows that the SR method only increases around 5% the total system's utility compared to the greedy strategy. To analyze this situation in more detail, a second set of experiments has been executed. During these experiments the robots carry out the transport mission during 30.000 time units and the soft utility function has been used. The objects are homogeneous, that is, all of them have the same utility value and function. After 30.000 time units, we get the time required to transport each object and the number of objects gathered. The figures 4 and 5 show the time needed to finish each task using the greedy and the single round strategies. The bar with a label 0.6 represents the percentage of tasks whose execution time exceeds 60% of the deadline. Comparing figures 4 and 5 it is remarkable that in the SR experiments the percentage of tasks highly exceeding the deadline has been significantly reduced. For example, during the experiments with the greedy strategy about 17% of tasks need more than 60% of the deadline time to finish, but during the SR experiments this percentage is zero.

6.2 On-line SVR experiments

In this subsection we will analyze the impact of the on-line learning on the total utility generated by the system. During these experiments 7 homogeneous robots

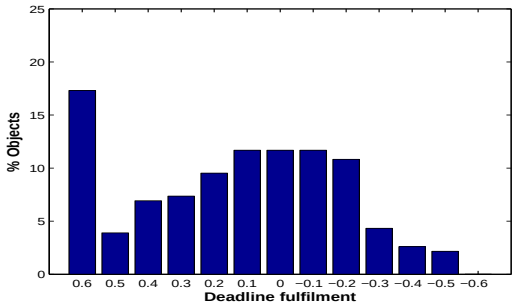

Fig. 4. Time required to finish the tasks during greedy robot selection.

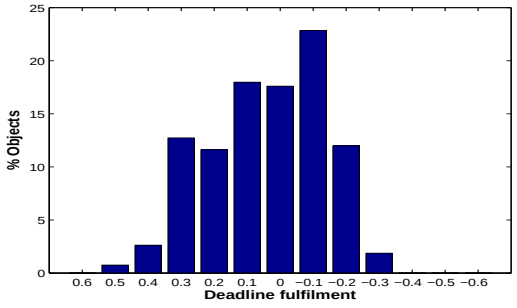

Fig. 5. Time required to finish the tasks during the single round experiments.

Table 1. Total Utility of the on-line SVR experiments

	off-line SVR	on-line SVR
Without errors	10802	11851
With errors	6290	10226

must transport 200 objects using the soft deadline utility function. Two kinds of experiments have been carried out:

- Experiments without errors: in this case the robots know without any error all their parameters: velocity, distance to delivery point, etc. Moreover, the 8% of inter-group interference overhead is not added to the ideal capacity.
- Experiments with errors: in these experiments the robots don't know their maximum velocity. The real maximum velocity of the robots is 3.0, but they suppose it's equal to 0,8. This wrong velocity is used to get the ideal capacity of the system. Thus, we can see what happens when there are errors in the kinematic model of the robot.

Table 1 shows the total utility of the system with both strategies, on-line and off-line SVR. As it can be seen, when there are no errors in the system, the on-line SVR algorithm only increases 9,7% the execution's utility with regards to the off-line strategy. By contrast, the on-line method significantly increases the utility of the system, 62,5%, when there are errors in the model of the robot. Thus, it can be stated that our system can model both, physical and kinematic interference.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a new auction method to fulfil the deadlines of a set of tasks using multi-robot coalitions. The results show that our two round auction method increases the total utility, both using hard and soft deadline utility functions. Moreover, the SVR method used to predict the expected utility of the coalition improves these results. Our method has been tested with transport tasks using both off-line and on-line SVR methods.

The work presented is in progress and has some challenging aspects to add and to improve. We are working to use a preemption auction method, that is a method that allows the exchange of robots between working groups. We will extend these experiments using real robots and other kind of tasks, like exploration, cleaning, etc. Moreover, we will study other on-line SVR sample substitution strategies. For example, our first results show that the clustering techniques proposed by W. Wang and Z. Xu in Wang and Xu (2004) are not useful for on-line learning because they require a lot of time.

REFERENCES

C.Chang and Lin, C. (2001). *LIBSVM: a library for support vector machines (http://www.csie.ntu.edu.tw/ cjlin/libsvm).*

Dias, M. and Stentz, A. (2003). Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. Technical Report CMU-RI-TR-03-19, Carnegie Mellon University.

Gerkey, B.P. and Mataric, M. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on robotics and Automation, Special Issue on Multi-robot Systems*, 18(5), 758–768.

Guerrero, J. and Oliver, G. (2004). Multi-robot task allocation method for heterogeneous tasks with priorities. In *7th. International Symposium on Distributed Autonomous Robotic Systems*. Tolouse (France).

Guerrero, J. and Oliver, G. (2007). Interference modelization in multi-robot auction methods. In *6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV'07)*.

Jones, E., Dias, M., and Stentz, A. (2006). Learning-enhanced market-based task allocation for disaster response. Technical Report CMU-RI-TR-06-48, Carnegie Mellon University.

Jones, E.G. (2009). *Multi-Robot Coordination in Domains with Intra-path Constraints*. Ph.D. thesis, The Robotics Institute Carnegie Mellon University.

Lerman, K. and Galstyan, A. (2002). Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2), 127–141.

Ma, J., Theiler, J., and Parkins, S. (2003). Accurate on-line support vector regression. *Jurnal Computation*, 15, 2683–2703.

Parrella, J. (2007). *Online Support Vector Regression*. Master thesis, University of Genoa, Italy.

Rosenfeld, A., Kaminka, G.A., and Kraus, S. (2005). A study of scalability properties in robotic teams. In *In*, 27–51. Springer-Verlag.

Vig, L. (2006). *Multi-Robot Coalition Formation*. Phd thesis, Gratuate School of Vanderbilt University.

Wang, W. and Xu, Z. (2004). A heuristic training for support vector regression. *Neurocomputing*, 61, 259–275.