2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

S+T: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation

Antidio Viguria
School of Electrical and
Computer Engineering
Georgia Institute of Technology
30332 Atlanta, USA
antidio@gatech.edu

Ivan Maza and Anibal Ollero
Robotics, Vision and Control Group
University of Seville
41092 Seville, Spain
{imaza,aollero}@cartuja.us.es

Abstract— **In this paper, we present a distributed market-based algorithm called S+T, which solves the multi-robot task allocation (MRTA) problem in applications that require the cooperation among the robots to accomplish all the tasks. If a robot cannot execute a task by itself, it asks for help and, if possible, another robot will provide the required service. In the paper, tasks consisting in transmitting data in real-time that could require communication relay services are considered. On the other hand, the parameters of the algorithm can be adapted to give priority to either the execution time or the energy consumption in the mission. The potential generation of deadlocks associated to the relation between tasks and services is studied, and as an original result, a distributed algorithm that prevent them is proposed. The algorithm has been tested in simulations that illustrate the main features of the S+T algorithm.**

I. INTRODUCTION

An important issue in distributed multirobot coordination is the *multi-robot task allocation* (MRTA) problem that has recently become a key research topic. It deals with the way to distribute tasks among the robots and requires to define some metrics to assess the relevance of assigning given tasks to the robots. In the last decade, different approaches has been used to solve this problem: centralized ([1], [2]), hybrid ([3], [6]) and distributed ([5] and [13]). Within the distributed approaches, the market-based approach [4] has been the most successful one which is based on the *Contract Net Protocol* ([10], [11]).

Usually these market-based approaches assume that each task can be executed completely by a single robot. But this could not be the case for example in a surveillance or exploration scenario, in which a task consisting in transmitting images in real-time could require another robot to act as a communication relay. Our approach to solve this problem is based on the concept of service. If a robot cannot execute a task by itself, it asks for help and, if possible, another robot will provide the required service. Required services are generated dynamically and are necessary to successfully complete their associated task. Other possible scenarios, where this approach is useful, could be the box-pushing problem and the cooperation among various robotic arms. In the first one, assuming that we know the weight of the box and how much weight a robot can push, one or more services could be required until the pushing capacity of the team of robots is equal or greater that the weight of the box. In the second scenario, it is supposed that we have several robotic arms with a limited set of tools and some overlapping of their workspaces. When a robot has to perform a task, it will need a group of tools. If these tools are not within its workspace, the robot will ask for a service to get the desired tool from another robot.

It is widely accepted that one of the main advantages of multi-robot systems w.r.t. a stand-alone robot is their capability to perform tasks that can be impossible for a single robot. In this paper a new task allocation protocol (called S+T), designed to exploit this characteristic, is described. This protocol is based on a distributed market-based approach and could be considered an extension of the SIT algorithm [12]. The basic idea is that a robot can ask for services when it cannot execute a task by itself. The cost of the task will be the sum of the costs of the task and the service or services required.

A similar idea is presented in [7], where soft temporal constraints were considered using master/slave relations, and also in [14], where the efficiency of the solution is increased considering at the same time the decomposition and allocation of complex tasks in a distributed manner. However, the potential execution loops associated to the relation between tasks and services that could lead to deadlock situations, is addressed in our paper. To the best of our knowledge, there is no other paper dealing with this problem in a distributed manner within the MRTA area. Moreover, the parameters of our algorithm can be adapted to give priority to either the execution time or the energy consumption (i.e., the sum of the distances traveled by each of the robots) in the mission.

The paper is organized as follows. In the next section the S+T algorithm is described and illustrated with a simple example. In the same section, the changes on the costs that allows the algorithm to prioritize between the execution time and the energy spent on the mission is also explained. In Section III, the deadlock problem is stated, and a distributed algorithm to solve it is explained. Simulation results that illustrate the main characteristics of the S+T algorithm are shown in Section IV. Finally, conclusions and future work are discussed in Section V.

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

II. SERVICES AND TASKS: S+T ALGORITHM

As any other market-based algorithm, there are two roles (bidders and auctioneer) that are played dynamically by the robots. The auctioneer is the agent in charge of announcing the tasks and selecting the best bid from all the received bids. The algorithms associated to each role are detailed in Algorithms 1 and 2. In the bidding process, when a robot needs a service to execute a given task, it will bid initially with just the cost of the task (because it still does not know the cost of the required services) labelling the message to the auctioneer as "provisional". The auctioneer will evaluate all the bids, and if the best bid requiring a service is better than the best bid without the need of a service, the robot requiring the service will start another auction in order to find which robots can perform that service. When this second auction is finished, the robot will send to the auctioneer the complete cost of the task, including the cost of the associated services. Afterwards, the auctioneer will decide which robot executes the task based on the updated costs. If a task is allocated to a robot requiring a service, that service will be allocated also at the same time.

Algorithm 1 S+T auctioneer algorithm

```
if there is any task to announce then
  announce task
  while timer is running do
    receive bids
  end while
  calculate best bid (lowest cost)
  if best bid is lower than the auctioneer bid then
    if best bid requires a service then
      allow robot to start a new auction in order to find
      a robot who can execute that service
    end if
    wait until the second auction is finished and the total
    cost of the task (including the service cost) is sent
    send task to best bidder taking into account the
    updated bids
  end if
  delete task from announcement list
  if task has an associated service then
    send a message to the robot that will execute the
    service in order to delete it from its local plan
  end if
end if
```

It should be pointed out that both the protocol used to allocate the services and the algorithm to allocate the tasks are based on the SIT algorithm presented in [12]. The only differences are:

- Services cannot be reallocated dynamically.
- When a robot that will execute a service changes its local plan, it has to report the new cost of the service to the robot which required it (that can start another auction to check if a different robot has a lower cost now for that task).

Algorithm 2 S+T bidder algorithm

```
a new message is received
if new message is a task announcement then
  compute the optimal insertion point for the task in the
  local plan
  calculate bid (marginal cost)
  if the task requires a service then
    send initial bid to the auctioneer and indicate that a
    service is needed
  else
    send bid to the auctioneer
  end if
else if new message allows to ask for a service then
  start a new auction in order to find a robot that can
  execute the service
  receive all the bids for the service
  calculate the complete cost for the task including the
  cost for the service
  send the new cost to the auctioneer
else if new message is a task award then
  insert task in the local plan in the position calculated
  before
  add task in the announcement list
  if the task needs a service, allocate the service to the
  robot that won the auction
  if the cost of any allocated service (in case it exists) has
  changed because of the insertion of the new task in the
  local plan then
    send the new cost of the service to the robot with the
    task
  end if
end if
```

A relevant feature of the protocol is that services can be allocated recursively, i.e., a robot that executes a service could also require another service to accomplish the first one and in this way to any number of recursive services. Therefore, the algorithm takes full advantage of the possibilities that a team of robots can offer (it is even possible to execute missions with a task involving the whole team).

In order to illustrate this characteristic, a surveillance mission will be considered. The mission consists in transmitting information from a certain area to a base station in real-time. The robot has to be within the communication range of the base or in the range of another robot acting as a communication relay. As it can be seen in Figure 1, the transmission to the base requires two robots acting as communication relays. The most relevant messages involved in the negotiation process are represented in the diagram depicted in Figure 2.

It should be pointed out that when a robot announces a service required for a certain task, the robot that will execute that task cannot take part in the auction process for the service.

The use of services increments the cooperation among robots and allows to achieve missions that could be impos-

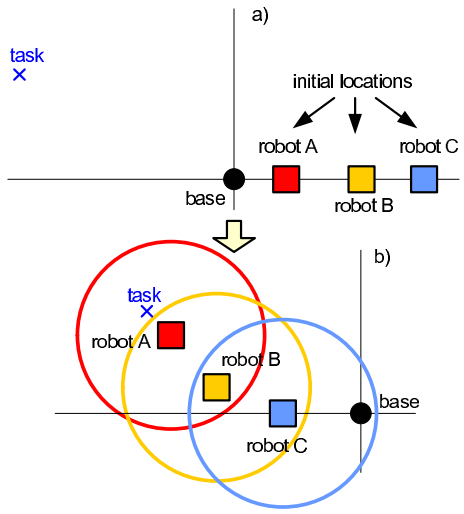

Fig. 1. Example of multiple recursive services required to accomplish one task. Figure a) shows the initial positions of the robots and the base station and b) shows the final assignment of tasks and services that allows robot A to transmit images to the base station using robots B and C as communication relays.

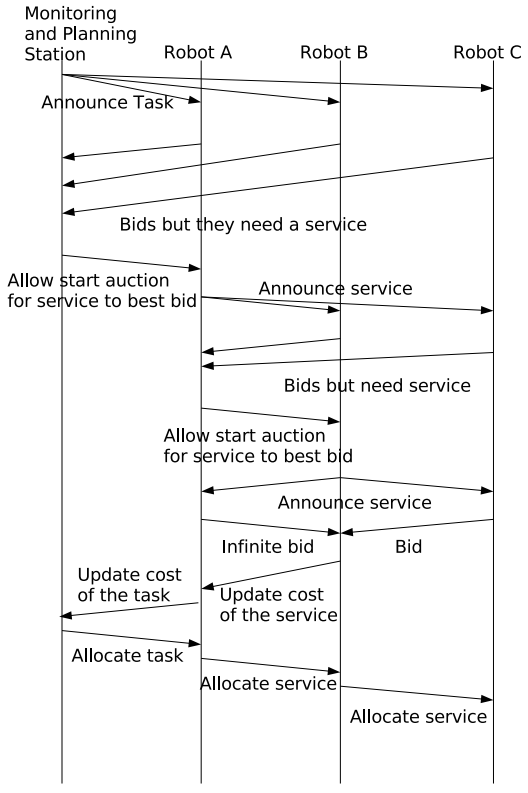

Fig. 2. Messages interchanged in the negotiation process using the S+T algorithm for the example illustrated in Figure 1 (one task requiring two services to be executed).

sible using a regular task allocation algorithm, for example, transmitting images in a surveillance mission from a position that does not have direct coverage with the base of operations. However, services can also increment the total time of the mission since more than one robot could be used to execute one task and, therefore, less tasks can be executed "in parallel". In this context, if a robot can execute a task by itself with a bigger cost than another robot using services, it should be decided which option is better. From our point of view, the answer to this question depends on the specific application and two different approaches have been developed to tackle with different scenarios:

- In our first approach, tasks have a higher priority than services, and therefore, it should be applied to scenarios where the goal is to minimize the total execution time of the mission. Basically, when an auctioneer receives bids from robots and, at least one of them does not require a service, the task will be directly allocated to it. This approach also needs less communication messages since services will be only considered when they are totally necessary for the success of the mission.
- In the second approach, the priority between the total time of the mission and the energy consumed by the team can be adjusted with a parameter α defined as follows:

$$\alpha = \frac{P}{1-P} \tag{1}$$

where $P \in [0,1]$ is the priority to minimize the total time of the mission. This parameter is used in the computation of the cost for the service:

$$C_s = C_o \cdot (1 + \alpha \cdot L) \tag{2}$$

where C_o is the original cost of the service, C_s is the new cost of the service and L is the level of the service, i.e., if it is the first service that depends on a task, L is equal to 1. If it is a service that depends on the first service, then L is equal to 2 and so on. This second parameter is used to penalize the use of more than one robot to execute one task. Moreover, when the use of services is unavoidable, L allows to increase the priority of services that need less robots.

The value of the parameter P should be selected depending on the type of mission. If it is more important to minimize the energy spent on the mission and the total time is not important for us, we should select $P = 0$, which means $\alpha = 0$. On the other hand, if we want to minimize the total time of the mission without considering a complete execution of all the tasks, we should select $P = 1$ which means $\alpha \to \infty$. In this case, services will not be considered and the algorithm will behave as the SIT market-based algorithm with local plans and reallocations [12].

III. DEADLOCK SITUATIONS

Until now, the allocation process of tasks and services has been presented, but not the synchronization issues related

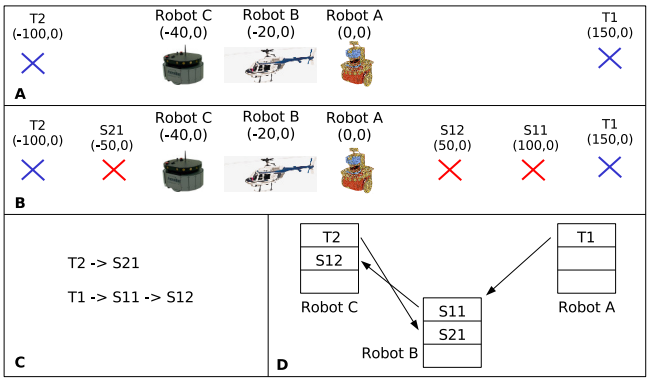

Fig. 3. Example where a deadlock is generated since the execution of the tasks depends on the execution of services. Figure A shows the initial position of the robots and the tasks to be allocated. Assuming a radius of communication of 50 units, Figure B shows the services needed to execute the tasks. Figure C shows the relation of execution between the tasks and the services and Figure D presents the relation in terms of execution in the final allocation using the S+T algorithm and the order of execution of the tasks in each robot. TI represents the task number I and SJL means a service associated to task J and level L.

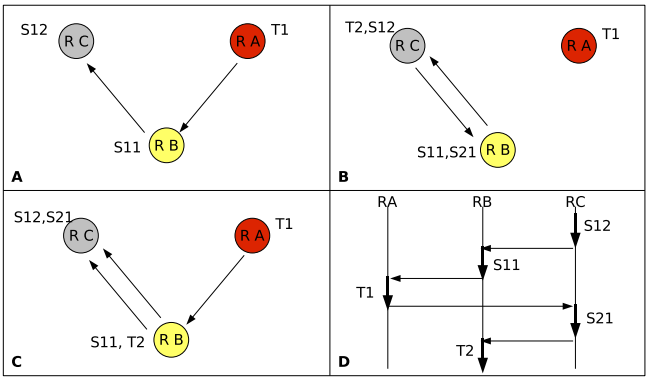

Fig. 4. Considering the initial configuration presented in Figure 3, Figure A shows the allocation after the announcement of the first task and the path followed by the "check loop" messages. Figure B presents the allocation of the second task and the path of the "check loop" messages that detects the execution loop. Figure C shows the allocation after the reallocation of the task and how the execution loop has been removed. Figure D presents the execution of the different tasks and services.

with the relation between tasks and services during the execution. From a general point of view, when the execution of tasks depends on others, the generation of deadlocks must be considered, and even more when the process is distributed. It has been noticed in simulation that this problem appeared frequently since each robot only has local information and there is no direct way to know if its particular local plan will generate a deadlock in the execution of all the tasks and services by the team of robots. For example, in Figure 3, it is shown how an execution loop can be generated using the S+T algorithm for a particular example with data transmission tasks and communication relay services.

This problem has not an easy solution since robots only have knowledge of their own plans. It is also important to find an algorithm to solve this problem in a distributed way since the key idea is to have a whole functional robotic system that works without the presence of a centralized entity. Our solution is based on the use of "check loop" messages, i.e., every time a robot wins a task, it will broadcast a message indicating the service associated to the new task (if it exists). The robot which has won that service will process the message and will send a message for every task or service that appears in its local plan before the mentioned service and has also a service associated to it. As it is shown in Figure 4, when a robot receives back a "check loop" message with its id, it will sell the task that provokes the loop and it will introduce it in a black list in order to avoid biding again for it. The use of a black list has the purpose to prevent the generation of allocation loops when the best two robots for a task are involved in an execution loop when they integrate the task in their local plans (i.e., they start to reallocate the task to each other and in both cases an execution loop is formed). Finally, the complete algorithm is shown in Algorithm 3.

Algorithm 3 Distributed loop detection algorithm

```
wait until receive a "check loop" message with a task or
service that the robot has in the local plan
if id message == robot id then
  if task has an associated service then
    send a cancel service message
  end if
  delete task from won-tasks list (loop detected)
  insert task in black-tasks list
  insert task in announcement-tasks list
else
  move to the initial position of the local plan
  repeat
    if task or service has a service associated to it then
      send "check loop" message
    end if
    next task or service in the local plan
  until task or service != task received in the "check loop"
  message
end if
```

IV. SIMULATION RESULTS

A multi-robot simulator has been used to test decentralized algorithms. This simulator is based on an architecture designed for heterogeneous robots and divided into three layers [8]. The highest layer is independent from the type of robot and is the one aware of the existence of other robots. Thus, the task allocation algorithm is implemented in this layer. Moreover, the communication among robots is based on IP using BBCS [9], so it can be also used as an interprocess communication method for simulations. The other two layers are used to execute the different tasks allocated to the robot and to make easier the simulation of new algorithms by using a modular and component-based architecture.

In the simulations, surveillance tasks where robots have to send back images in real-time to a base station from a

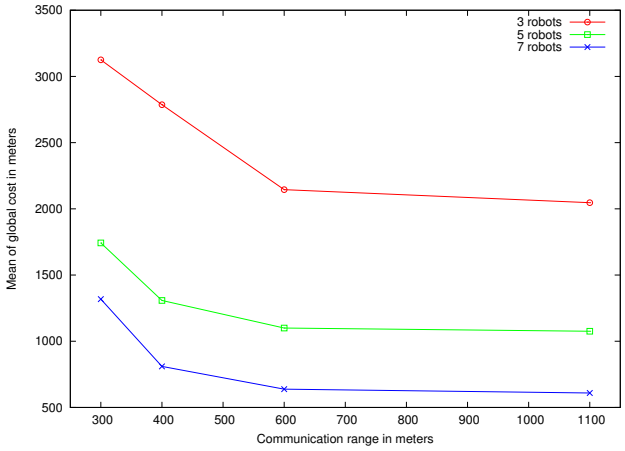

Fig. 5. Mean of the total distance traveled by all the robots over one hundred missions with different communication ranges, number of robots and five tasks.

Robots	Comm. range (m)	Total distance (m)	Messages received	Services
3	600	2145.15	47.96	0.56
	400	2786.52	80.32	2.44
	300	3125.23	150.45	4.36
5	1100	1075.23	48.06	0.0
	600	1099.43	52.3	0.30
	400	1307.97	85.66	1.36
	300	1742.34	164.87	3.45
7	1100	609.14	45.06	0.0
	600	638.42	45.8	0.24
	400	810.23	79.76	1.24
	300	1318.31	142.96	2.76

TABLE I

RESULTS WITH FIVE TASKS, DIFFERENT NUMBER OF ROBOTS AND VALUES FOR THE COMMUNICATION RANGE. THE MEAN OF THE VALUES FROM ONE HUNDRED RANDOM MISSIONS ARE SHOWN WHERE TOTAL DISTANCE MEANS THE DISTANCE TRAVELED BY ALL THE ROBOTS, MESSAGES RECEIVED IS THE NUMBER OF MESSAGES RECEIVED BY ONE ROBOT IN THE S+T ALGORITHM AND NUMBER OF SERVICES IS RELATED TO THE ONES EXECUTED BY ONE ROBOT.

certain point were considered. Therefore, a robot transmitting images have to be within the communication range of the base station using its own communication device or using one or more robots as communication relays (services). For this particular scenario, the execution synchronization between tasks and services has been implemented using preconditions, i.e. a task cannot start until all the services associated to it have been executed. Moreover, the robot or robots that execute a service cannot start the next task or service in their local plan until the associated services have been completed.

Numerous simulations with different number of robots were performed for the surveillance missions mentioned above with several communication range values in a scenario of 1000x1000 meters. In Figure 5, it can be observed that the total distance traveled by all the robots decreases when the communication range increases as far as the probability to require a service decreases. The total distance traveled by all the robots is considered as a good measurement of the energy spent during the mission. Moreover, the mean of the total distance traveled decreases when the number of robots increases due to the fact that a constant number of tasks is used in all the missions.

Table I shows the resulting mean values of some parameters in missions with five tasks, different number of robots and values for the communication range. The number of services executed increases when the communication range of the robots decreases and, as a logical consequence, the number of messages received by one robot and the total distance traveled by all of them also increases, as it was mentioned above. This means that the communication requirements and the energy needed to execute the mission will be higher when the number of services increases.

On the other hand, simulations have been run with different values of the α parameter that depends on $P \in [0, 1]$ (see Section II). As it can be seen in Figure 6, one hundred random simulations have been executed for different values of P. $P = 0$ is an extreme value applied when the user wants to minimize the total distance traveled by all the robots in the mission in terms of energy, and therefore, the cost of the services is not modified. Also in Figure 6, it can be observed how the maximum distance traveled by one robot decreases when P increases, and therefore, the time of the mission will be smaller (assuming that all the robots move at the same speed) because of the penalization of the costs associated to the services. However, if the execution time is critical, with $P = 1.0$ the S+T algorithm services are not considered and some tasks could be undone (mission partially accomplished). In Figure 7, it is shown the mean of the number of tasks executed over 100 missions with different values for the communication range and with $P = 1.0$. Up to six hundreds meters, it can be seen that a significant number of tasks cannot be accomplished for the group of robots if the use of services is not considered. Therefore, we have to be careful when the parameter P is equal to 1.0 and a given mission needs services to execute most of the tasks. In that case, the time of the mission will be minimized but many tasks will not be executed. Then, it is advisable to only use $P = 1.0$ when most of the tasks can be executed without services and the execution time of the mission is very critical.

V. CONCLUSIONS AND FUTURE WORK

A distributed task allocation algorithm called S+T and based on a market-based approach has been presented. In order to execute tasks that need more than one robot, the concept of service has been introduced. The basic idea is that a robot can ask to others for services when it cannot execute a task by itself. Two approaches for the algorithm has been developed. In the first one, services are only considered on the allocation process when none of the robots can execute a particular task by itself. The second approach can be adapted to the type of application with a parameter α prioritizing

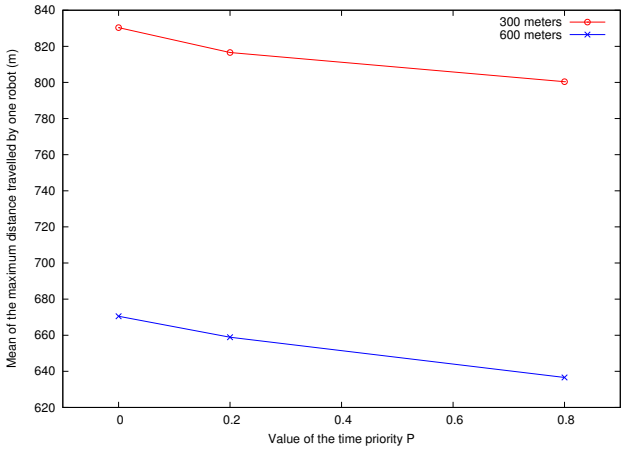

Fig. 6. Mean of the maximum distance traveled by one robot over one hundred missions with $300m$ and $600m$ as the communication range and five number of robots and tasks.

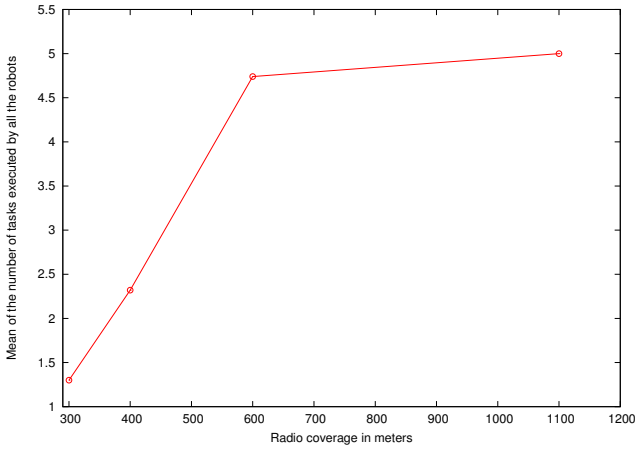

Fig. 7. Mean of the number of tasks executed by all the robots over one hundred missions with different values of the communication range and five number of robots and tasks. The use of services are not considered in this simulations, i.e., $P = 1.0$ or $\alpha \rightarrow \infty$.

between the execution time and the energy consumption in the mission. Regarding the execution of tasks and services, the creation of deadlocks has been studied and a distributed algorithm that avoids them has been introduced.

Finally, the use of an algorithm such as S+T could increase the probability of completing a mission when tasks need more than one robot to be executed. But, this advantage entails an overhead in the allocation process. Moreover, not all the missions might need services to be completed successfully. Therefore, we plan to study how to create an algorithm that can be adapted dynamically to the needs of the specific situation switching between the two approaches presented in Section II or modifying the parameter α using learning techniques.

ACKNOWLEDGMENTS

This work has been partially funded by the European Union AWARE Project (IST-2006-33579) and the AEROSENS Project of the Spanish Research and Development Program (DPI2005-02293).

REFERENCES

[1] B. Brumitt and A. Stenz. GRAMMPS: A generalized mission planner for multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.

[2] P. Caloud, W. Choi, J. Latombe, C. Le Pape, and M. Yim. Indoor automation with many mobile robots. In *Proceedings of the IEEE International Workshop on Intelligent Robotics and Systems (IROS)*, 1990.

[3] M. B. Dias and A. Stenz. Opportunistic optimization for market-based multirobot control. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2714–2720, Lausanne, Switzerland, 2002.

[4] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE Special Issue on Multirobot Coordination*, 94(7), 2006.

[5] B.P. Gerkey and M.J. Matarić. Murdoch: Publish/subscribe task allocation for heterogeneous agents. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 203–204, Barcelona, Spain, 2000.

[6] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical decision-theoretic approach to multi-robot mapping and exploration. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2714–2720, 2003.

[7] T. Lemaire, R. Alami, and S. Lacroix. A distributed tasks allocation scheme in multi-UAV context. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[8] I. Maza, A. Viguria, and A. Ollero. Networked aerial-ground robot system with distributed task allocation for disaster management. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, Gaithersburg, MD, USA, 2006.

[9] V. Remuss, M. Musial, and U.W. Brandenburg. BBCS robust communication system for distributed system. In *Proceedings of the International Workshop on Safety, Security, and Rescue Robotics*, Bonn, Germany, 2004.

[10] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, 1993.

[11] G. Smith. The Contract Net Protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12), 1980.

[12] A. Viguria, I. Maza, and A. Ollero. SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007.

[13] B. B. Werger and M. J. Matarić. Broadcast of local eligibility for multi-target observation. In *Distributed Autonomous Robotic Systems 4*, pages 347 – 356. Springer-Verlag, 2000.

[14] R. M. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research, Special Issue on the 4th International Conference on Field and Service Robotics*, 25(1):73–101, 2006.