A Distributed Tasks Allocation Scheme in Multi-UAV Context

Thomas Lemaire, Rachid Alami, Simon Lacroix

A Distributed Tasks Allocation Scheme in Multi-UAV Context

Thomas Lemaire, Rachid Alami, Simon Lacroix
LAAS-CNRS
7, av du Colonel Roche
31077 Toulouse Cedex 4
Email: firstname.name@laas.fr

***Abstract*— This paper deals with the task allocation problem in multi-robot systems. We propose a completely distributed architecture, where robots dynamically allocate their tasks while they are building their plans. We first focus on the problem of simple "goto" tasks allocation: our approach involves an incremental task allocation algorithm based on the *Contract-Net protocol*. We introduce a parameter called *equity coefficient* in order to equilibrate the workload between the different robots and to control the triggering of the auction process. Then, we address the problem raised by temporal constraints between tasks, by dynamically specifying temporary hierarchies among the tasks. Tests run in simulation quantify the benefits of our improvements.**

I. Introduction

A. Context

The coordination of multiple robots gathered in a single system is a big challenge in the robotics area. A team of robots can achieve tasks faster, safer, better than a single robot, and of course a team can accomplish operations that a single robot cannot execute alone. This kind of system targets applications which range from mapping missions of buildings or in natural environment, rescue or intervention missions in hazardous area, to planetary exploration or deployment of equipment without human intervention. An exemplary project in this domain is *Centibots*[1], which aims at demonstrating a hundred robots system mapping, exploring and monitoring a large building in a coherent fashion during a period of 24 hours.

Our work takes place in the context of the EEC funded project *COMETS*[2], which aims at designing, developping and demonstrating a distributed control system for cooperative activities using heterogeneous UAVs. Targeted applications are surveillance and monitoring, the project being more specifically oriented towards forest fire monitoring applications. In this context, the tasks the UAVs must achieve are essentially "goto" tasks, that are generated either directly by an operator or a central system that analyses all the information acquired by the various UAVs. Some tasks are intrinsically *cooperative tasks*, such as the 3D monitoring of a fire front, that requires the simultaneous perception of the fire from at least two distinct positions. Other constraints, such as maintaining a communication link between the UAVs and the central station, can generate cooperative behaviors, *e.g* by defining communication relays.

[1] `http://www.ai.sri.com/centibots/`

[2] COMETS stands for "Real-time Coordination and Control of Multiple Heterogeneous UAVs - see `http://www.comets-uavs.org/`

B. Problem statement

The development of multi-robot architectures implies the development of three main decisional abilities: mission planning, task allocation and coordinated task achievement [1], [2]. Analysis is restricted here to task allocation issues. The mission is a partially ordered set of tasks. This may be the result of a previous mission decomposition phase. Coordination and resource-conflict issues that are involved when the robots effectively achieves their tasks once they are allocated [3] are not addressed here. Given a system of several robots, and given that set of tasks, we want the robots to allocate all the tasks to each other and build their plans accordingly in order to complete the mission. They should also be able to dynamically modify the allocation, and consequently their plans, to adapt to changes in the environment or to new requests issued by the operator. The system must have dynamic allocation and planning abilities, and must satisfy constraints on energy resources and communication ranges, that are both limited. Also, all the robots must ensure sufficient energy to go back to their starting point when their tasks are achieved.

One main goal of this work is to keep the architecture distributed among the robots so as to gain scalability, robustness and reactivity, with the drawback that the solution's optimality is not guaranteed.

In the context of *COMETS*, the tasks the system of robots must achieve consist of (*i*) navigation tasks (*i.e.* reach a given position), and (*ii*) perception tasks. The latter tasks can be achieved while the UAV hovers at a given position, or while it follows a predefined path (*e.g* circling around a given target), but they can also imply the simultaneous presence of several UAVs, hovering or moving according to predefined geometric patterns. Therefore, two distinct problems must be solved:

1) Allocation and planning of navigation tasks is concidered.
2) The problem of constraints on tasks schedules which enables the system to deal with complex tasks that need synchronization of robots activities is addressed.

The first problem is a multi Traveling Salesmen Problem (often referred as *m-TSP*). The second problem is an extension of the previous one, in which constraints on the execution dates of the tasks must be satisfied: it is possible to stipulate for a task the date when it must be started with respect to the start date of another task.

This paper describes a task allocation scheme based on *Contract-Net*, and is innovatory in two ways :

For the first problem, we aim at minimizing a global criteria (the longest trip) while *Contract-Net* only takes into account local data. We introduce in *Contract-Net* a global parameter which helps the optimization the criteria. It also helps to control the auctions generation in the system.

For the second problem, the robots must share up-to-date data that describe the constraints on the tasks and need to plan the tasks that are linked together accordingly, without disrupting the bidding process of *Contract-Net*. This will be done with the temporary assignment of master/slave role to the robots depending on the tasks that have been allocated. The challenge lies in avoiding the use of a centralized planning algorithm.

C. Related work

Rabideau *et al.* made a comparison of several methods for tasks allocation in [4]. They emphasize algorithms with three degrees of distribution, the most distributed one is based on the *Contract-Net* protocol. In [5] Bellingham *et al.* successfully implement in simulation an algorithm for the *optimal fleet coordination problem*, their algorithm does not address the problem of synchronization between tasks and can be classified in level 2 from [4]. Dias and Stenz also studied different approaches to the task allocation problem with multiple robots in [6] and came to the point that distributed algorithms based on *Contract-Net* suit the needs. Note that a number of distributed schemes for task allocation in multi-robot domains have been proposed in the literature. One of the first is [7]. ALLIANCE [8] is a distributed behavior-based architecture, which uses motivations that enable/inhibit behaviors, resulting in tasks (re)allocation.

Contract-Net has been introduced by Smith in [9] and further developed by Sandholm [10]. Since 1999 *Contract-Net* have widely been used in multi-robot applications [11], [12], [13]. Stenz and Dias work on an architecture called *TraderBots* in which *leaders* can optimize the plan of several other robots [14], [15], Mataric' explored various strategies for the *Contract-Net* protocol in [16]. Several studies dealing with concrete mission such as buildings or planetary exploration [17], [18], [19] or emergency handling [16], have shown the feasibility and the performance of the *Contract-Net* architecture in real world situations.

To our knowledge, on the problem of allocation and planning of non-independent tasks in a distributed multi-robot system, only one paper from Kalra and Stentz [20] presents preliminary results on the *sweeping perimeter problem*. In this work, the temporal window taken into account is very small, the coordination is explicit between a limited number of robots (one robot and his two neighbors), and the market-based approach is not fully exploited since the auctions implies three agents only.

D. Outline

The next section introduces an *equity coefficient* that is used in the bids evaluation and to control the auction generation process within *Contract-Net*. Quantitative simulation results obtained on the m-TSP problem illustrate the improvements brought by the consideration of this coefficient with respect to a plain *Contract-Net* approach. Section III deals with the introduction of time constrained tasks in *Contract-Net*. It shows that the introduction of simple execution date contraints can help to cope with cooperative tasks, that are either requested by an operator or automatically generated within the system, to establish communication relays for instance. Finally, a discussion concludes the paper.

II. Contract-Net with equity

In the classic market based approach, each agent (for us robots) can make a public auction for one of its tasks, and then the other robots can bid on that task using a given cost function. The winner of the bidding process gets the task and must insert it in its plan. In order to drive the process toward an optimal solution, one agent can sell a task only if the bidden cost for the execution of that task is at least less than a certain amount of its own execution cost (generally 10% less). The cost function we will use here is simple and is calculated from the distance the robot will travel.

A. Equity factor

The aim is to obtain an allocation that minimizes the length of the longest trip, which also can be seen as minimizing the duration of the mission. Our idea is to adress this global optimization problem by considering two aspects: first, *Contract-Net* is used to assign the tasks to the robots at a low cost so as to keep the total distance traveled by the team of robots not too far from optimality, and second, equity is enforced between the robots so as to really distribute the tasks among them and obtain a mission which is as short as possible.

For this purpose, we introduce a measure of equity called *equity coefficient* (C_{eq}). Each robot can compute its own workload (wl) using a cost function: the workload is the cost of the whole plan of the robot. The robots broadcast the value of their workload to the others and each one can compute its C_{eq}. For the robot A the formula is :

$$C_{eq}^{A} = \frac{wl(A) - \overline{wl}}{\overline{wl}} \tag{1}$$

Where $\overline{wl}$ is the mean of $wl(.)$ over the robots for which A knows the workload. Indeed, since we consider limited communication range, A may have only a partial knowledge of the workloads. The meaning of this coefficient is :

- $C_{eq}^{A} < 0$: robot A has a too small plan with respect to the other robots.
- $C_{eq}^{A} > 0$: robot A is overloaded with respect to the other robots.
- $C_{eq}^{A} > C_{eq}^{B}$: robot A has more work than robot B.

B. Equity factor and task evaluation

In *Contract-Net* a task is allocated to the robot which can insert it in its plan for the lowest cost; also the robot should not be too overloaded. For that the evaluation the robot makes for a task is modified by taking into account its C_{eq}. The utility the robot A computes for the task T_1 ($ut^A(T_1)$) is corrected in $ut'^A(T_1)$ by :

$$ut'^A(T_1) = ut^A(T_1) - C_{eq}^{A} \times |ut^A(T_1)|$$

This correction is applied to the utility computed by both the auctioneer and the bidder. By this mean *Contract-Net* is influenced the way we want :

- A robot with a high workload will more easily reallocate its tasks and will get new tasks with more difficulty because its utility for the tasks is lowered.
- On the contrary, a robot with a low workload will be more easily allocated new tasks but will give up its own tasks with more difficulty because its utility for the tasks is increased.

C. Control of the auctions generation

The problem here is we do not want several auctions being launched at the same time. Basically, the *Contract-Net protocol* does not provide any details when the agents of the system can start an auction, and other papers do not emphasize this point either. Our need is to keep the system entirely distributed, so we do not want an authority which would give the right to the robots in turns, and we want to keep the system dynamic so we do not want to give to each agent a static list which would define the turns for the auctions.

Our solution is inspired by the *token-ring* networks in which a token passes from one computer to another to give them the authorization to send their data over the network. Here the token allows the robot to make an auction.

1) Token circulation: The robot that has got the token is the auction leader. If another robot is willing to make an auction, it can ask for the token to the current auction leader. It sends his request along with its C_{eq}. The owner of the token collects all the requests, it is also allowed to request for the token. It then randomly chooses the next owner of the token, using a random distribution based on the collected C_{eq} (the more a robot is overloaded, the higher chance it has to get the token). This is done to help overloaded robots to reallocate their tasks.

2) Token creation: When a robot wants to make an auction, but nobody has the token, it then creates a token and uses itself to make an auction, the process is started spontaneously ! Because of communication delay, it may happen that several robots create a token at the same time, this is why we specify the following behavior :

- If a robot that is not currently an auctioneer receives several auctions at the same time, it then bids on the auction which has the higher priority *ie* the higher C_{eq} (the auctioneer gives its C_{eq} along with the auction). The other auctions are ignored.
- If a robot that is currently an auctioneer receives other auctions, it keeps on his own auction only if it has the higher C_{eq}, else it cancels its auction and can bid on the auction with the higher priority.

D. Results

A typical mission allocation and execution goes this way: (1) A set of tasks is given to the system (either directly by an operator, or issued from a decomposition process). (2) The base is a simple *Contract-Net* agent except that it has a high priority (an artificially high equity factor) and will never make a bid. The base starts making auctions with the tasks of the mission and goes on until all the tasks are allocated to the robots. (3) The base does not keep the token any more and the robots can start making auctions. (4) The process stops when none of the robots ask for the token. A robot stops making auction when it has already auctioned all its tasks and no reallocation has occurred. If not, the robot auctions again all its tasks. This stop criterion is quite different from what has been done until now (usually a fixed number of auctions turns). (5) The mission starts being executed by the robots. The auction process starts again when new tasks are requested by the operator, or when a robot fails to achieve its plan.

scenario	$\bar{l}$	$\sigma(l)$	$\min(l)$	$\max(l)$	n
grouped/no equity	5248	674	3035	6150	133
grouped/equity	2195	264	1844	2954	156
scattered/no equity	2481	481	1724	4631	133
scattered/equity	1895	162	1581	2343	160

Fig. 1. This table summarizes the statistical results over 100 runs of the simulation of four scenarii, considering a grouped or scattered start, and with or without the use of the equity coefficient. l is the length of the maximal tour, and n is the number of auctions of the allocation process.

In the tests, we focus on steps 2 to 4. The results were obtained with our simulator *SiMuRob*[3].

We based all our tests on the same mission: 50 points picked up in the environment have to be visited by a team of 4 robots. The points have been uniformly randomly generated once in this environment. In order to show the interest of our *equity coefficient*, we run scenarii with the coefficient disabled (we give it a fixed value so as to mimic a plain *Contract-Net protocol*). Another important point is how the robots are firstly distributed in the environment. If they are scattered (the usual situation when *Contract-Net* is used) each robot is implicitly attributed a different area, the area surrounding its initial position, because of the cost function which is based on traveled distance. If they are initially grouped around a same point (which is mostly the case in operational situation), the problem is more difficult.

The results presented in table 1 show that the solution obtained with the equity coefficient is improved by a factor of 2.4 over the standard *Contract-Net*, if the robots are scattered, the improvement factor drops to 1.3. This is due to the fact that the solution found by the standard protocol is already a good one. The interest of our method is that it works well even if the initial situation is not favourable. On the other side the allocation process is about 20% longer (more auctions are done) with the equity coefficient enabled.

III. Time-constrained tasks in a distributed environment

The problem of constrained tasks allocation and planning for a system of multiple robots is commonly addressed with a centralized planner such as *GRAMMPS* [21]. We sketch here how we deal with simple time constrained tasks in our distributed environment.

[3] *SiMuRob* (Simulation Multi-Robot) is a Java application developed at LAAS-CNRS and can be freely downloaded at `www.laas.fr/~tlemaire/`.

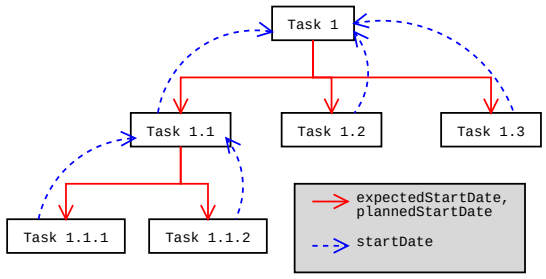

Fig. 2. This tree shows up the hierarchical links between tasks.

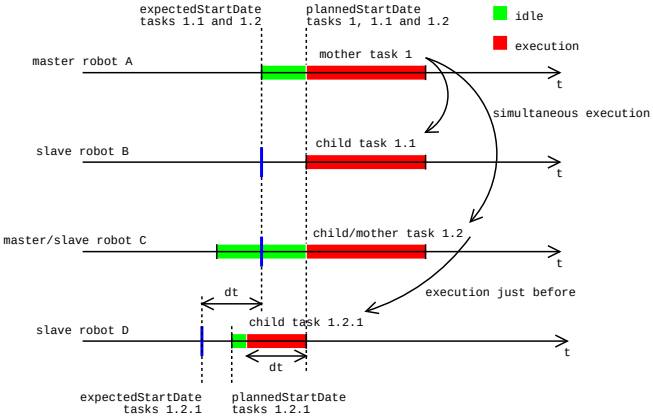

Fig. 3. This is an example of the plans of four robots after the allocation of four constrained tasks.

A. Execution around date d

This constraint means that we will try to have a given task executed more or less at a given date. This constraint enables the system to deal with constraints on relative date of execution of several tasks expressed numerically: T_1 and T_2 simultaneously or T_1 n seconds before of after T_2. We choose to put the constraint *execution around date* d on the tasks for several reasons :

- This constraint is *soft*, which means that there is an infinite number of solutions that satisfy it, the distributed allocation algorithm will find even a bad solution and will not end to dead-lock.
- The quality of satisfaction for that constraint is easily *measurable*, and then we are able to take into consideration this measure when we evaluate the utility the robot will gain in executing that task. The quality of satisfaction for the constraint can be directly included in the bid of our *Contract-Net protocol*.
- The information needed to plan such constrained tasks is very limited and will not overload the communication bandwidth between the robots.

B. Constrained tasks tree

The constraint can be used for example to enforce simultaneous execution of two tasks. One task T_1 which is planned for execution at date d_1 puts on task T_2 the constraint *execution around date* d_1. The task T_1 is said to be the mother task, and task T_2 the child task.

Temporally T_1 is defined by a $startDate$ d_1 (the date when it can be executed at the earliest), and a $plannedStartDate$ (the date when it will actually be executed). T_2 has the same attributes plus an $expectedStartDate$ d_1 (the preferred date for its execution).

The allocation process must allocate T_1 before T_2 (because we need to know d_1 for bidding on T_2), but the tasks can be reallocated later. It is important to note that only T_2 is constrained, T_1 is allocated and firstly planned as usual. After the allocation process, the *master* robot R_A (the one which will be executing T_1) will choose $plannedStartDate$ for both T_1 and T_2, the robot R_B (the one which will be executing T_2) is called the *slave* robot. Since the system is dynamic, changes can be made to the plans of R_A and R_B. If it happens, the slave robot only informs its master of the changes in its plan, it sends the new $startDate$ for that task and then the master robot computes a new $plannedStartDate$ for the execution of the two tasks which is acceptable by both R_A and R_B.

The relation master/slave between the robots is local in time (only for the execution of the considered tasks), and temporary because the tasks can be reallocated to other robots. So this is quite different from the *TraderBots* architecture [15].

Figure 2 presents a tree of tasks and focus on the data that are exchanged in order to plan the tasks, and figure 3 sketches the allocated tasks from the robot point of view. The synchronization between robots is actually accomplished with the introduction of *idle* periods in the robots plans.

C. Evaluation of the utility of a task

Now the quality of satisfaction for the constraints which weight on the tasks of the plan is to be taken into account. Previously we computed the cost of a plan with its length, now we add a term for each task which reflects the constraint satisfaction quality. We call this term $deltaDate$, for the constrained task T_i the formula is :

$$deltaDate_i = |startDate_i - expectedStartDate_i| + \sum_j deltaDate_{ij}$$

where $deltaDate_{ij}$ comes from the children tasks T_{ij} of task T_i. These children tasks are either allocated to the same robot or to another one.

The utility of a plan can now be computed by the formula :

$$planUtility = -\left[movingCost + k \times \sum_{task_i \in plan} deltaDate_i\right]$$

The robot bids on a task with the value $(planUtility' - planUtility)$ where $planUtility$ and $planUtility'$ are respectively the utility of the plan *before* and *after* the insertion of the task.

The factor k is here to normalize the sum. In fact we add two quantities $movingCost$ and $deltaDate$ which are not of the same nature. This becomes false if the $movingCost$ is computed with the time needed by the robots to go from one point to another. One can understand k as a scale factor, $k = 0.1$ (it is the typical value we use.) means that we find the

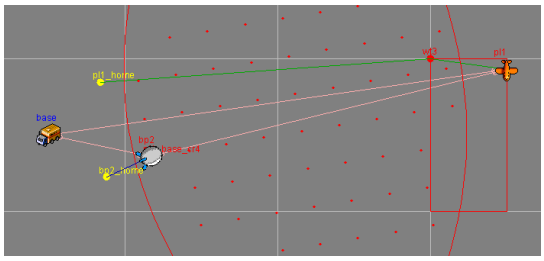

Fig. 4. This screenshot presents a plane `pl1` which is watching out an area and a blimp `bp2` which is a communication relay between `pl1` and the base. The pink lines represents the communication links which are available (the link `pl1`→base is not available). The red circle arcs enclose the area where the blimp can serve as a communication relay, and the dots represents the discrete positions, the planner of the blimp has chosen one of these positions.

periods when the robot is idle 10 times less important than when the robot is active.

D. Time consistency of the plans

We must ensure that the time constrained tasks are planned correctly to prevent the system from ending into deadlock. Here again we use a very basic planner, not really efficient, but very easy to implement and which clearly maintains consistency of the plans. Each child task is tagged with an $expectedStartDate$, the planner will insert the task into the plan so as to respect the local chronology between children tasks of this plan.

When a modification occurs in the plan, we use a simple but rough process to maintain this local consistency: if two children tasks are not in the chronological order, they are swapped.

Assuming that the plans are incrementally built, the local consistency ensures the global consistency. Indeed, the synchronization is reached by inserting idle periods into the plans of the robots so a robot waits even a long time for synchronization rather than trying to swap tasks.

E. Implemented tasks

Two new tasks have been implemented in our simulation to illustrate constrained tasks.

The *watch-out* task consists for the robot to travel around a rectangular area to be monitored and the robot must keep communication with the base. If the communication link between the base and the robot cannot be maintained during the execution of the task, then the robot should generate a *com-relay* task between it and the base which is to be executed by another robot at the same time the watch-out task is executed. The com-relay task can be recursive, which means that several robots can be needed to effectively maintain communication between the base and the robot which will be watching out the area. Figure 4 illustrates these two tasks.

F. Results

Here the results are more qualitative. The simulator shows that even with very simple planning algorithms in a distributed environment, we manage to allocate and plan a mission correctly. Figure 5 presents what we obtain with our simulator on some examples. It illustrates the strategy found by the team : since for the watch-out tasks two robots are needed (one for the given task, and one more for the com-relay), we can see that the four robots are split into two teams of two robots and each team takes care of a part of the environment. When there remains only three robots, the solution is more complex and less structured, but is valid and does not appear to be very sub-optimal.

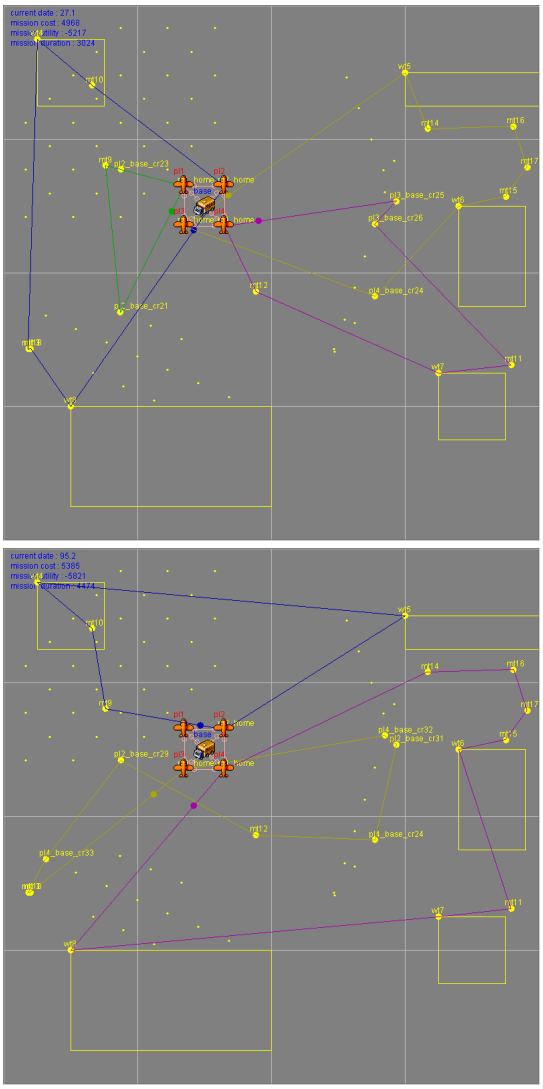

Fig. 5. Top: some watch-out and goto tasks are allocated to the team of robots. Bottom: The plane `pl4` (top-left) has fallen out of order and the tasks have been reallocated to the remaining robots.

IV. Discussion and perspectives

A. A simple planner

The planner we use to insert the tasks into the plans of the robots is very simple, it adds a task by choosing the best index inside the current tasks list to execute the new task. We also added an improvement procedure for the plans which is based on an *r-opt* algorithm from [22]. This algorithm tries to locally improve the plan by swapping r-tasks. We implemented the algorithm for $r = 2$ which eliminates most of the loops we could observe in the plans of the robots before.

Even if the two blocks (planner and plan optimization) we use are not very efficient, we think this architecture is interesting for a planner used in collaboration with *Contract-Net*.A simple but fast planner can rapidly compute a bid to participate to an auction. And a more elaborate algorithm can improve the current plan to alleviate the limitations of the fast planner. This algorithm could compute a whole new plan or try to modify the current plan. It can be run periodically and/or when important changes occur in the current plan.

B. Types of constraints

In our current implementation we deal with a relaxed problem of temporal contraints. Tasks may be *numerically* partially ordered: the constraint between T_1 and T_2 cannot be "T_1 before T_2" but must be "T_1 n seconds before T_2". As far as we know, the problem of distributed constrained tasks allocation is still unsolved.

C. Various metrics for task evaluation

Mataric' *et al.* introduces in [13] a metric for task evaluation depending on the kind of the task which is being auctioned. This enables the system to work with a great variety of tasks. In our case, we have a few number of tasks but they can be used in different contexts. We think about a metric which could depend on the *context* of the mission rather than on the tasks themselves, this metric could be chosen by an operator or automatically determined by the system when an alarm is detected for example. Such metrics could be :

- A metric based on the *energy* necessary to achieve a given task. It could be used when the robots have to watch-out an area, must detect alarms,... In these cases one wants the system to save energy so as the mission could be executed a longer time.
- A metric based on the *time* necessary to achieve a given task. Rescue missions for example can benefit from a reduced execution time.

V. Conclusion

In this paper we have successfully developed a totally distributed architecture for multi-robot tasks allocation. The *Contract-Net protocol* has been enhanced with a token based system which manages the turns for the auctions independently of any extern authority and is associated with an innovatory stop criterion. Moreover, to our knowledge it is the first time that global synthetic information is taken into account during the bidding process of *Contract-Net*. Our algorithm takes into account both a cost computed by the planner of the robot and a global attribute called *equity coefficient*. This kind of simple attribute is well-adapted to distributed platform since its value is indicative and thus does not require ideal communication between the robots to be computed, and we demonstrate that it can substantially improve the performance of the whole system.

On another side, an open problem is pointed out: the design of planning algorithms that can be run on a distributed platform with limited communication bandwidth such as a system of robots. Another difficult point is the robustness of the algorithm when communication is temporally lost. We demonstrate with our simulation that this is feasible, but with several limitations such as the kind of available constraints that can weight on the tasks.

References

[1] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *IEEE Int. Conf. on Robotics and Automation (ICRA'99)*, 1999.

[2] ——, "Robots that cooperatively enhance their plans." in *Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS)*, vol. 4, 2000, pp. 55–65.

[3] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, "Multi-robot cooperation in the martha project," *IEEE Robotics and Automation Magazine, Special Issues: Robotics and Automation in Europe*, 1997.

[4] S. Chien, A. Barrett, T. Estlin, and G. Rabideau, "A comparison of coordinated planning methods for cooperating rovers," in *Proceedings of the Fourth International Conference on Autonomous Agents*, C. Sierra, M. Gini, and J. S. Rosenschein, Eds. Barcelona, Catalonia, Spain: ACM Press, June 2000, pp. 100–101, poster announcement.

[5] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, ch. Multi-task allocation and path planning for cooperating UAVs.

[6] M. B. Dias and A. T. Stentz, "A market approach to multirobot coordination," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI -TR-01-26, August 2001.

[7] H. Asama and K. Ozaki, "Negotiation between multiple mobile robots and an environment manager," in *IEEE Int. Conf. on Robotics and Automation (ICRA'91)*, 1991, pp. 533–5382.

[8] L. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 2, pp. 220–239, 1998.

[9] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," in *IEEE Transaction on Computers*, ser. C-29, no. 12, 1980, pp. 1104–1113.

[10] T. Sandholm, "An implementation of the contract net protocol based on marginal cost calculations," in *Proceedings of the 11th National Conference on Artificial Intelligence*. Menlo Park, CA, USA: AAAI Press, July 1993, pp. 256–263.

[11] A. T. Stentz and M. B. Dias, "A free market architecture for coordinating multiple robots," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-99-42, December 1999.

[12] M. B. Dias and A. T. Stentz, "A free market architecture for distributed control of a multirobot system," in *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, July 2000, pp. 115–122.

[13] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," in *IEEE Transaction on Robotics and Automation*, vol. 18, 2002, pp. 758–768.

[14] M. B. Dias and A. T. Stentz, "Enhanced negotiation and opportunistic optimization for market-based multirobot coordination," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI -TR-02-18, August 2002.

[15] ——, "Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI -TR-03-19, August 2003.

[16] M. J. Mataric', G. S. Sukhatme, and E. Ostergaard, "Multi-robot task allocation in uncertain environments," *Autonomous Robots*, 2003.

[17] M. J. Mataric' and G. Sukhatme, "Task-allocation and coordination of multiple robots for planetary exploration," in *10th International Conference on Advanced Robotics*, August 2001, pp. 61–70.

[18] R. M. Zlot, A. T. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *IEEE International Conference on Robotics and Automation*, May 2002.

[19] M. B. Dias, D. Goldberg, and A. T. Stentz, "Market-based multirobot coordination for complex space applications," in *The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, May 2003.

[20] N. Kalra and A. T. Stentz, "A market approach to tightly-coupled multi-robot coordination: first results," in *CTA (Collaborative Technology Alliance) robotics program*, 2003. [Online]. Available: http://www.frc.ri.cmu.edu/~axs/doc/cta03.pdf

[21] B. L. Brumitt and A. Stentz, "GRAMMPS: A generalized mission planner for multiple mobile robots in unstructured environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*. Piscataway: IEEE Computer Society, May 16–20 1998, pp. 1564–1571.

[22] E. L. Lawler, J. K. L. A. H. G. R. Kan, and D. B. Shmoys, *The Traveling Salesman Problem*. New York: John Wiley & Sons, 1985, ch. Empirical analysis of heuristics, Heuristics for the TSP.