

# Simulated annealing for multi-robot hierarchical task allocation with flexible constraints and objective functions

Alejandro R. Mosteo and Luis Montano

*Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza*  
*Instituto de Investigación en Ingeniería de Aragón, I3A*  
*c/ María de Luna 1, E 50018 Zaragoza, Spain*  
*{amosteo, montano}@unizar.es*

**Abstract**—In this paper we study a novel approach in networked robotics for optimal allocation with interchangeable objective functions, from minimizing the worst-case cost of any agent in a multi-robot team in time-critical missions, to minimizing the team usage of resources. We propose a general model for flexible mission planning, using hierarchical task networks as descriptive framework, the multiple traveling salesmen as optimization model, and distributed simulated annealing for solution search in very large solution spaces. This proposal does not discard viable solutions, hence the optimal one for the model may be eventually found. We briefly comment on the feasibility and usefulness of full replication of data in critical missions. We present a working implementation in simulation and preliminary plans for urban environment experiments, and also an implementation using adapted market-based techniques for comparison against our proposal. For our simulations we use a coverage problem, where our algorithm enables us to determine the best starting point for the robots, exploiting the flexibility of HTNs.

**Keywords:** multi-robot, task allocation, hierarchical tasks, robot coordination, replanning, networked environment, sensing-actuation.

## I. INTRODUCTION

Cooperative networked robotics is a hot topic in current research, since the use of several robots allows to tackle new problems unfeasible for a single robot and quicker, more reliable, failure-proof solutions in general. Fields that can benefit from multi-robot cooperation are civil aid (USAR<sup>1</sup>), exploration in structured [1] or unstructured [2] environments, autonomous patrolling of known [3] or unknown areas [4], victim detection [5] and sensor networks [6] for quick response in emergency situations.

However, multi-robot teams introduce new challenges and a fundamental one is the optimal use of available resources. We can summarize it as expressed in [7] by the two questions:

- What do we do?
- Who does what?

For clarity, the first question refers to *planning*, while the second refers to *allocation*.

We can find theoretical studies [8]–[10] aiming to systematize the principles behind the problem, which is in general recognized as a NP-Hard one [11]. There is also a large body of experimental work, being auction-based techniques [2], [12], [13] one of the most fruitful. These techniques can often find suboptimal solutions by the nature of the algorithms employed, which can not always guarantee the eventual finding of the optimal solution or give quality bounds. Recent work addressing these shortcomings can be found in [11], [14], [15]. *Hierarchical Task Networks* (HTN) [16] are also used to expand planning capabilities. We find them in task allocation contexts in, for example, [7], [17], [18].

Another new aspect introduced by the multi-agent context is that of the optimization criterion. Cited solutions tend to polarize around *TotalSum* minimization, that is, minimizing the sum of all the individual agent costs<sup>2</sup>. Other useful criterion is the *MinMax* one, where the goal is to minimize the worst agent cost. In terms of time this would imply finding the shortest mission timespan [20].

In this paper we propose a solution for the simultaneous solving of the planning and allocation problems. We believe it has several key advantages: it allows flexible problem modeling using HTNs and arbitrary task ordering constraints. There are no explicit restrictions on the quality of the solutions found, because the complete solution space is potentially reachable (although in unbounded time). This is achieved by using the simulated annealing optimization technique, used before in robotic path planners [21]. It is applicable to heterogeneous robots by using a general multiple traveling salesmen model [22]. It is applicable without adaptation to both described optimization criteria and also to other mixed criteria via *criterion descriptors*. Our simulations back these claims and also show that it is a viable solution for real-time use in physical robots. We outline the experiments we are preparing

<sup>2</sup>A problem of maximization can be conversely seen as a problem of minimization by making the appropriate transformations [19]. It is common in market based approaches to talk about maximizing the utility, whereas in the *Traveling Salesman Problem* realm the common talk is of minimization. Following these conventions, we will use the respective nomenclature in sections dealing with each one. The comparisons resulting from simulation will be made in terms of costs.

<sup>1</sup>Urban Search And Rescue

right now to field test our method. In short, we propose it as a general solution for the planning and allocation problems that can be applied to many particular robotic contexts.

We also present an implementation of recent market-based allocation in the line of [17] for comparison, modified to use the *MinMax* criterion. To the best of our knowledge, this particular modification is an original work.

The paper is structured as follows. In section II we present the general problem to solve and the particular example used for our simulations. Section III reflects our thoughts on data replication. Sections IV and V describe our main proposal and the market-based implementation respectively. We follow with simulation results in section VI and finally present our plans for outdoor experimentation and conclusions in sections VII and VIII.

## II. PROBLEM STATEMENT

The general problem we want to solve is the choosing of executable plans in a fully expanded HTN and the allocation of its primitive tasks to robots in such a fashion that the solution is optimal according to the minimization criterion, that can be *MinMax*, *TotalSum* or a combination of both, bearing in mind the following difficulties:

- 1) The HTN can imply an exponential number of executable plans, caused by the use of OR nodes [7].
- 2) The optimal assignment of any of these plans is an instance of the MTSP<sup>3</sup> NP-Hard problem [9].
- 3) Introduction of new tasks can occur on-line, causing replanning and reallocation of tasks.
- 4) The full mission is considered critical, that is, its completion depends on completion of all tasks and no task loss should occur.

The particular problem used in this paper as simulation example is the complete exploration of a parking and urban area in minimum time by a robotic team, including to find the best entry point in the area to be explored for each one of the available robots. Real life applications include vehicle finding, vehicle cataloging, explosive detection, as examples.

The approach we propose in this paper is the use of HTNs for the description of the mission and generation of executable plans, the MTSP as the cost model for primitive tasks, and the use of simulated annealing to perform the minimization in real time. We will also provide a market-based solution for comparison. Our solutions, like those in [7], [17], aim to solve both problems of plan generation and task allocation simultaneously.

For the problem at hand, we are making the following assumptions and simplifications:

- We have a floorplan of the area to be explored. This could be obtained from an aerial photography, a database of critical targets or the national body in charge of buildings and roads.
- Entry points are not arbitrary, but we have a predefined set of these to choose from.

<sup>3</sup>Multiple Traveling Salesmen Problem

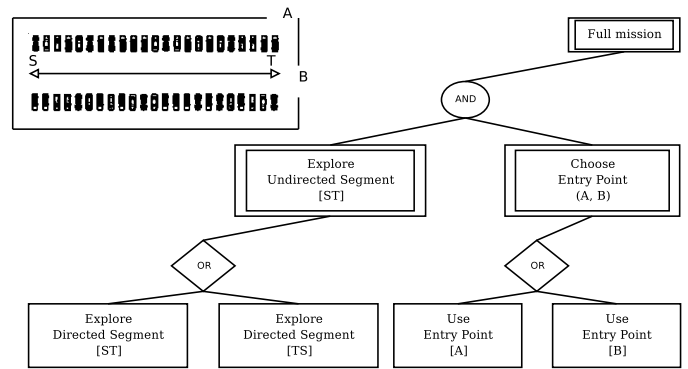


Fig. 1. Task tree for a simplified example with a single robot. In the left we see a parking with two rows of cars that can be explored in one pass (segment  $ST$ ). There are two entry points to choose from, A and B. In the tree, double-line boxes are abstract tasks and single-line boxes are primitive tasks. There are four possible executable plans, caused by the two OR nodes.

- The robots are always connected forming a MANET<sup>4</sup> [23]).
- The robots are capable of localization and navigation and carry appropriate sensors for the mission at hand.

### A. Hierarchical tasks

We believe HTNs are a powerful and expressive tool for high level thinking, making easy for humans the high level description of missions. In figure 1 we show an schema of the tasks used in our simulations. We have a first set of tasks related to the exploration problem:

- *Main exploration*: This represents the exploration of the entire area. This task can be expanded using coverage algorithms (though for our purposes we have done it by hand) in a set of straight paths [24], [25] or, in the case of streets, each street will equal a task. The rationale is that a robot, while walking down such a straight path, is capable of detecting with its sensors some relevant features of a nearby vehicle.
- *Explore Undirected Segment*: These are each of the segments result of the decomposition. Since we do not want to discard any possibilities, we assume that a robot can explore them in any of the two ways. Each of these tasks are expanded as two OR-related *Directed Segment* tasks. That is, each segment has to be visited once, in any of the two ways.
- *Explore Directed Segment*: This primitive task means that a robot must cover the segment in an explicit direction.

We also have tasks whose purpose is to find the best entry point for the robots if the area has several of them, so they can be deployed optimally.

- *Choose Entry Point*: This compound task is a collection of all the allowed entry points. There are as many tasks of this kind as robots in the team.
- *Use Entry Point*: This primitive task simply means that the robot has to be deployed at the given Entry Point.

<sup>4</sup>Mobile ad-hoc network

Each compound *Choose Entry Point* is trivially expanded as an OR node with an underlying *Use Entry Point* for each allowed entry point (see figure 1).

The entry tasks carry a constraint so they must be scheduled as the first one in any robot. For this reason, since there are as many tasks of this kind as robots, each robot will have assigned to it no more and no less than one entry point, as is to be expected. Entry points can be shared by several robots.

### B. Cost model

We are using a sequential task execution model, in which a robot can only execute a task at a time but can have a list of assigned tasks of arbitrary length. The analogy with the MTSP is as follows: Cities represent the ending of execution of a task, and arcs represent the cost of executing it. Thus, the cost of traveling from a city to another is the cost of performing a task, given the previously finished one. Solving this MTSP gives in effect the solution to the task allocation. To define the problem, the robots have to publish a cost-matrix of size  $|T_p|^2$  to the solvers, being  $|T_p|$  the number of primitive tasks. This is enough to evaluate any allocation. We are using the most general variety of the problem: m-cost [22] (since robots can be heterogeneous or have different sensed data) with asymmetric costs.

A well known criterion for the MTSP is the *TotalSum* one

$$A_{\text{Opt}} = \arg \min_A \left( \sum C_r(A) \right), r \in \{r_1, \dots, r_n\} \quad (1)$$

where  $A$  is an allocation of tasks to robots and  $C_r$  is the cost incurred by a robot  $r$  of the robot team  $R$  given the task allocation.

In this criterion, we are interested in minimizing the sum of all agent costs. An example domain is transportation (fuel optimization). The *homogeneous* instance of the MTSP, where all agents have a same cost matrix, can be solved transforming it to a single-traveler TSP [26]. In [22] we learn that the heterogeneous case is not solvable this way and is furthermore a very hard problem. It presents nonetheless big interest in robotic applications where robots can be heterogeneous. Our framework support this latter case.

Other notable optimization criterion is the *MinMax* one:

$$A_{\text{Opt}} = \arg \min_A \left( \max (C_r(A)) \right), r \in \{r_1, \dots, r_n\} \quad (2)$$

In this criteria we are interested in having a best worst case. Note that, in a pure *MinMax* optimization, the agent plans that are not the worst one can be suboptimal, since they are completely “hidden” by the worst case. This means that, using this criterion, we can end with a plan that is inefficient for the non-critical agents. Our proposal can explicitly deal with this circumstance, as follows.

To choose between *MinMax* and *TotalSum* strategies, we use what we call *criterion descriptors*. These are a 2-tuple  $(\omega_M, \omega_T)$ , with  $\omega_M$  being the weight factor for the *MinMax* cost, and  $\omega_T$  the weight factor for the *TotalSum* cost. The final optimization cost is computed simply as

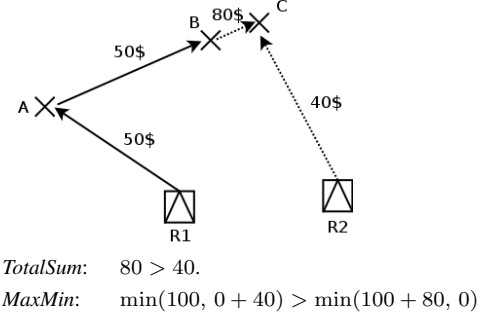


Fig. 2. The *TotalSum* criterion would give task C to R1, even if R2 is idle, whereas the *MaxMin* criterion will award C to R2. Note that the *MaxMin* criterion is the equivalent in terms of profit to the *MinMax* in terms of cost.

$$C = \omega_M C_M + \omega_T C_T \quad (3)$$

where

$$C_M = \max C_r(A), r \in \{r_1, \dots, r_n\} \quad (4)$$

$$C_T = \sum C_r(A), r \in \{r_1, \dots, r_n\} \quad (5)$$

correspond to *MinMax* ( $C_M$ ) and *TotalSum* ( $C_T$ ) costs, respectively, and  $A$  is some assignment. It is straightforward that the  $(1, 0)$  descriptor corresponds to a pure *MinMax* optimization, while  $(0, 1)$  corresponds to a *TotalSum* one. The usage of mixed descriptors like  $(1, 0.001)$  will correspond to a *MinMax* optimization where lower *TotalSum* costs are favored, in fact alleviating the problem of suboptimal non-critical agent plans, since they are now actively taken into account during the minimization. Hence we achieve a dominating *MinMax* cost, where the non-worst agent costs are *TotalSum* optimized. We plan to further study the influence of different descriptors and their possible applications and benefits in future work. In this paper we have focused on pure *MinMax* optimization for our simulations. We consider that criterion descriptors are a generalization of the *equity coefficient* used in [20] and help to clarify the issues behind their use.

It is important to note that many market-based allocation schemes are implicitly implementing a *TotalSum* optimization in terms of utility. This is usually justified saying that a robot, by trying to maximize its profits, is maximizing the whole team profit [2], [17]. However, this is not a sensible optimization criterion when total mission timespan is the critical objective. This manifest itself in situations like the one shown in figure 2 where two robots are visiting spots of interest. Let us suppose R1 is already committed to tasks A, B when C is auctioned. Even if R2 is idle, R1 profit for C is greater than that of R2, and R1 will win the task.

In our simulations, the cost for each primitive task are:

- *Explore Directed Segment*: The time required to travel the segment plus the time needed to arrive to the starting segment pose from the current pose.
- *Use Entry Point*: Zero if the first task in the agent plan, infinite otherwise.

### III. DATA REPLICATION IN CRITICAL MISSIONS

Purely distributed algorithms like market-based allocation can incur in data loss if the robot owning a task is lost. To avoid this, it has been suggested [27] that robots should track tasks they have sold to reclaim them if circumstances deem it necessary. This, ultimately, means that a chain of ownership is formed to some initial seller so a kind of centralization is introduced into an otherwise distributed solution.

For small teams like ours, and taking advantage of the broadcast nature of the wireless medium, we believe that a total replication of the critical data (like pending tasks and best solution known) amongst all the connected devices is a solid answer to avoid any data loss (also suggested in [27]). The goal is to have a copy of each shared information in each node; copies are synchronized during normal operation. In abnormal circumstances, nodes can have old non-updated data but this only means that, temporarily, sub-optimal solutions could be computed due to missing updates.

We obviously can face bandwidth and latency constraints with this solution. This has not been a problem for now in our experimentation. Results in geographic routing [28] suggest that efficient broadcast in large networks can be achieved with latency costs  $O(n \log n)$  and resource costs  $O(n)$  being  $n$  the number of nodes. To ensure integrity of the shared copies we have implemented a lightweight object-oriented database. Further details are given in the section on implementation architecture.

To summarize, we enumerate the perceived advantages and disadvantages of this total replication:

Advantages:

- No dependence on some central authority.
- Transparency for the data user, who simply sees a regular database.
- Immediate availability of all required data.

Drawbacks:

- Bandwidth and latency constraints.
- Would require specialized broadcast/clustering techniques for “large<sup>5</sup>” teams.

### IV. SIMULATED ANNEALING SOLUTION

In this section we describe our main proposal. Simulated annealing [29] is an optimization technique commonly used for problems which are intractable by traditional means and which fits well with combinatorial problems. It has been successfully used in the solving of traditional NP-Hard problems like the Traveling Salesman Problem [30] and also in robotic path planners [21].

Simulated annealing mimics the metallurgy process of annealing, which involves heating and cooling cycles. In simulation, the current solution is replaced by some variation of it, which will be accepted or rejected with a certain probability  $p(C_{old}, C_{new}, temperature)$ . This probability depends on the

difference of quality of the solutions and on the current temperature. When temperature reaches zero, only better solutions are accepted (greedy behavior), while during hot stages worse solutions can replace the current one. This allows to escape from local minima.

A fundamental element of the annealing problem is the generation of neighbor solutions. We can leverage this point not only to solve the MTSP problem, but also to explore alternative plan expansions; hence we are simultaneously solving the planning and allocation problems. As a consequence, this is also the key ingredient of our approach to determine the best entry point for each robot in our simulation. Another advantage of the annealing approach is that we can address the most complex and general m-cost asymmetric MTSP model with arbitrary constraints on the task ordering or execution time windows. This is easily handled avoiding the generation of invalid solutions when feasible or, failing that, by giving infinite cost to any solution violating constraints.

The generality of our approach has two principal drawbacks: we lack solution quality bounds, unless the particular problem at hand could provide these; and although the optimal solution is reachable, it can take any amount of time to find it given the probabilistic nature of the annealing process and the exponential size of the solution space.

While the annealing process is centralized in principle, we think that in our current project this can be used to our advantage by running an identical algorithm with different random seeds in every robot (or connected computing device). The required information for the algorithm is propagated using the shared database.

In our approach, the execution of the mission is divided in two stages: a start-up stage, where an initial solution is computed during a reasonably short period compared to the expected total mission time, and the execution stage.

#### A. Start-up stage

While we could count on a precalculated database of relevant urban locations (for example, public parkings in principal cities) for initial precomputed solutions, the start-up stage will account for a lack of any precalculated data. Once the relevant blueprints of the working area are obtained, the human operator must provide the possible entry points for the robots. Meanwhile, a set of *Explore Undirected Segment* tasks will be created. The task tree is submitted to the robots, which in turn compute their cost-matrix for the primitive tasks. With this information, the planning can begin and the algorithm is run in every computing device available.

#### B. Execution stage

Once a prudential time has elapsed (in our experiments we give one minute for the start-up stage), robots are committed to the best solution found and start task execution. However, the annealing algorithm can continue, taking into account the current robot commitments. This way, better plans can be found at any moment. We can also benefit from the decreasing size of the remaining problem, which means that the final

<sup>5</sup>We have not actively researched this point to properly characterize this “large” size.

stages will be likely solved to optimality. Also any deviation in the sensed environment from the *a priori* data can be integrated and taken into account.

In essence, each robot and connected resource can continually work in the search of a better solution. It is not critical to communicate complete solutions until a robot is about to finish its current task, so network usage can be minimized by just broadcasting the cost of the best solution known.

### C. Neighborhood generation

In our implementation we have several kinds of solution mutators that produce a new neighbor, as follows:

- Random reinsertion.  
This movement will randomly choose a task from any agent, remove it from the owner plan and insert it again at a random point in the plan of a random agent.
- Random swap.  
Two tasks are chosen and its positions swapped.
- Random plan switch.  
In this movement, an OR branch is chosen and replaced randomly by one of its siblings.
- Guided movements.  
Very much like the previous movements, but in this case we choose a task or branch from the most costly agent plan, in a guided attempt to improve the worst case.
- Heuristics.  
In this class of movements we have two more expensive TSP heuristics, based in greedy insertion [26].

We have found that this mix of purely random movements and heuristic movements works well. In one hand, pure random movements allow the exploration of far apart areas in the solution space. In the other hand, guided and heuristic movements quickly find good solutions in a given local area. With the use of appropriate data structures, random and guided movements can be performed with cost  $O(\log \max(N_r, N_t))$ , being  $N_r$  the number of robots and  $N_t$  the number of tasks. Heuristics are computationally more expensive and can cause the search to “sink” to a fixed solution state from a large number of states, so we give them a very small probability of being performed. Our implementation using the Gnat Ada [31] compiler explores around 2000-6000 solutions per second in a standard PC a year old, depending on the problem size.

Another advantage of this approach is that mutations probabilities can be tweaked at will. This allows to have a large stock of mutations, maybe some highly specialized ones, that can be enabled or disabled if the particular problem requires it.

### D. On-line replanning

We manage the apparition of new tasks the following way:

1. Publish the new task to all robots.
2. Robots publish new costs for the task.
3. Quickly assign task via auction.
4. Set the new task allocation as best known solution.
5. Proceed with annealing.

Changes in costs are similarly managed:

1. Robot publishes the updated costs.
2. Reevaluate the best solution cost.
3. Proceed with annealing.

This mechanism ensures quick response and seamless integration with the annealing algorithm.

## V. MARKET-BASED SOLUTION

We will use a solution akin to that in [17] for comparison with our annealing proposal, with a notable modification to use the *MinMax* criterion. This modification, in practice, turns the marketplace into a heuristic for the *MinMax* criterion. We give it in terms of costs:

- 1) Seller broadcasts tuple  $\{t, C_a\}$
- 2) Bidders reply with  $C_b$  if  $C_b < C_a$
- 3) Bidder with minimum  $C_b$  is awarded  $t$

where meanings are:

- $t$ : auctioned task.
- $C_a$ : auctioneer plan cost *without* removing  $t$  from its plan.
- $C_b$ : bidder plan cost *including*  $t$  in its plan.

Since  $C_b < C_a$  and also  $C_a$  will decrease after removing  $t$  from the auctioneer plan, in practice we are reducing the maximum cost.

In our implementation, robots auction subtrees entirely owned by them. Since in this particular problem all OR nodes consist of single primitive tasks (see figure 1), the auctions will always be for OR nodes and never for primitive tasks. A bidder will consider each branch of the OR node, and bid using the best one for his current plan. The strategies tried to insert a new task are

- Best insertion point.
- Optimal TSP solution when the plan has less than 20 tasks.
- Complete greedy replanning of all owned nodes.

Finally, the *Choose Entry Point* subtrees are not used in our market simulations, since robots need to know where they are to properly bid. Hence, this solution does not address the choosing of entry points.

## VI. SIMULATIONS

In this section we will present the results obtained by the two proposed algorithms. As mentioned, we are aiming for pure *MinMax* optimization, both in the annealing and the market-based implementations. We have used three simulated scenarios of different complexity. In every one of them, the tasks for the market-based execution are initially auctioned in random order. After completing the auctioning, enough full rounds of auctions are run by each robot until no plan improvements occur. In the annealing planner, we simulate the complete running period, with the restriction that a started task cannot be removed from the robot that is executing it. Multiple planners are not run in parallel, but a single one is used for the simulations. The cooling schedule is a exponential one, which is cyclically applied over the complete simulation time. The

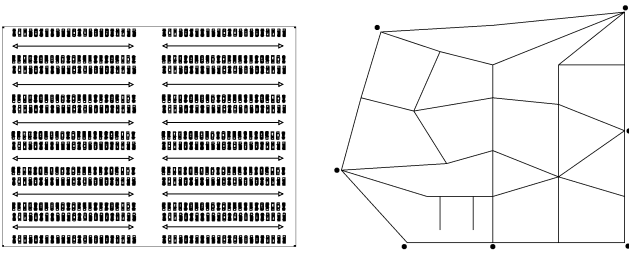


Fig. 3. Left: The small parking scenario, with *Explore Undirected Segment* tasks shown as arrows. Right: The city schema used for the large scenario, with dots signaling entry points.

initial solution is constructed using a greedy allocation after a random executable plan is selected.

The first scenario is a small parking roughly inspired in the one of our workplace, which has 12 lanes to explore as seen in Fig. 3. This gives  $2^{12}$ , that is 4096, possible executable plans. Two entry points are considered for this parking, one at the center top and another one at the center bottom. The second scenario is a bigger model with the same structure, with a total of  $2^{32}$  possible plans, which we consider big enough to be regarded as intractable by full enumeration of solutions. The third scenario (Fig. 3) is a model of the streets in a small area of our city, with a total of  $2^{43}$  possible plans, and some additional difficulties like dead-ends and a more irregular disposition of the segment tasks. This scenario has seven possible entry points. We shall call these three scenarios *small*, *medium* and *large* for short. Note that even if the two parking scenarios have apparent symmetries in the disposition of tasks, there are small differences that suppress them. We have done this in purpose to avoid multiplicity of solutions and problem simplification, although on a real implementation we would of course want to leverage these when possible.

Fig. 4 shows the minimum, maximum, median and quartile costs over 10 runs of each algorithm with a team of four robots, which is the size of our real robotic team. The *SA* column gives the cost computed by the annealing planner having to chose entry points; the *SA<sub>Fix</sub>* gives the cost computed by the annealing planner using from the start the best entry points previously found. The *Market* column gives the *MinMax* cost computed by the market simulation with the same entry points

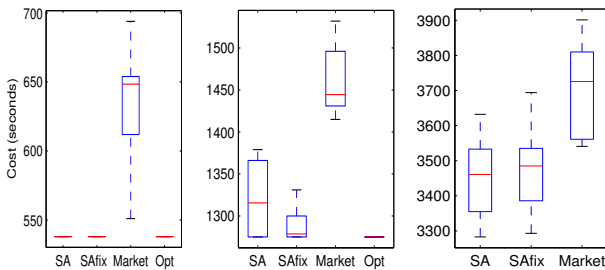


Fig. 4. Costs of solutions for the 12-, 32- and 43-lane problems.

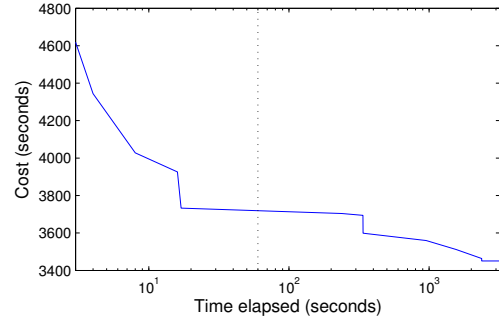


Fig. 5. Evolution of the best solution found over a complete run of the algorithm in the large parking problem with 4 robots. The vertical line indicates execution stage start at second 60. Better solutions found after this point can still be applied.

as *SA<sub>Fix</sub>*. Finally, the *Opt* column gives the optimal solution cost, which we know for the parking scenarios. These costs do not include the time given for the start-up stage or the auction rounds, but are the final solutions computed by the algorithms.

As we can see, the simulated annealing solutions are consistently better than the market-based ones, which is to be expected from a centralized theoretically complete<sup>6</sup> algorithm over an heuristic one. *SA* solutions are within a 0%-4% of the optimal known solution, whereas the market ones are about 15%-18% off which is, nonetheless, a good result, specially for problems of this complexity. Note that the small scenario has been solved to optimality in all runs by the annealing algorithm, while only a few times in the medium scenary and probably none in the large one.

Since we don't know the optimal solution of the urban scenario, we can not give assurances on quality solution, but again we see a better performance from the annealing solver. In table I we give the ratio mean/best for each algorithm and problem size. These ratios also indicate a decrease in the quality of solutions when the problem grows in size for the annealing solver, whereas the gap with the market-based solution shrinks. Further experimentation should be carried to see if this trend reaches a point where greedy heuristic like first-round auctions are as good as our annealing proposal.

TABLE I

RATIO OF MEAN TO BEST SOLUTION

Scenario	SA	SA <sub>Fix</sub>	Market
Small	1.00	1.00	1.18
Medium	1.04	1.02	1.15
Large	1.05	1.06	1.13

Table II shows the quality of solutions for the different scenarios when considering different number of robots for the *SA* algorithm. The values are the ratio mean/best solution cost. Columns show the already noted decrease in quality of solutions as the problem gets larger, whereas rows highlight

<sup>6</sup>with infinite running time.

that adding robots increases the number of solutions and so the performance degrades over a larger solution space.

TABLE II  
TEAM SIZE & SA ALGORITHM

Scenario	2 robots	4 robots	8 robots
Small	1.00	1.00	1.00
Medium	1.00	1.04	1.05
Large	1.03	1.05	1.06

In Fig. 5 we can see the progress of the best solution cost obtained by the SA algorithm in a typical run. We observe that in the first seconds there is a quick evolution, which is to be expected since the first plan was chosen at random and can be highly suboptimal. After that, slow improvements can be obtained over all the mission duration, and specially when the problem size reduces in the final period.

## VII. EXPERIMENTS WITH REAL ROBOTS

Our work is part of the EXPRES<sup>7</sup> project, which is framed around the USAR principles and aims to provide robotic help in distress situations where human intervention is dangerous. The objective is to build a cooperative robot team which can perform complex tasks in civil aid operations with a high degree of autonomy. Examples of missions that are being studied in the EXPRES project are victim location in road tunnel accidents (with emphasis in maintaining a completely connected mobile MANET) and the one which has been examined in this paper: complete exploration of a parking or, more generally, urban area for quick location of interesting vehicles, objects or people. The motivation for such a task is that it is a common tactic of some terrorist groups in our country to advertise the planting of a bomb car at some location, often supplying critical data about the vehicle such as the plaque or the model. The rationale seems to be that this *modus operandi* creates distress amongst the civil population, but only police lives are put at risk in the deactivation mission.

We are implementing the algorithm for use with our team of outdoor Pioneer3 AT robots (figure 7) within the next weeks. We intend to demonstrate it in the parking of our lab for the final version of this paper.

### A. Software architecture

The pivotal concept behind our robotic architecture is modularity. Each desired functionality is implemented as a plugin that can be selectively enabled. Plugins can be chained (see figure 6) if their input/output datatypes match. The most relevant modules for this experiments are:

- *Core*: Handles the communication with the robot hardware.
- *Network*: Provides access to the wireless medium.
- *Database*: Manages the shared database.
- *Annealer*: Performs the optimization presented in this paper.
- *Executer*: Carries out the task list assigned to the robot.

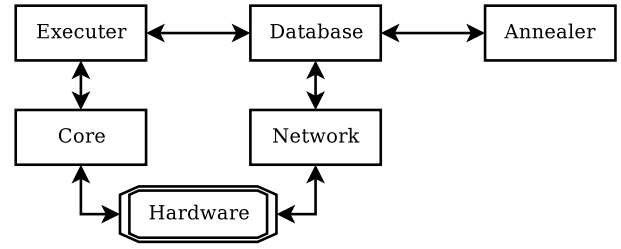


Fig. 6. Relevant plugged modules in each robot using the EXPRES software architecture.

### B. Planned experiments

Real robots bring in interesting problems directly caused by the interaction with the environment. These include self-localization, for which we will use other research already done in our group in the field of scan-matching, and replanning when facing unexpected or new sensed data.

We expect optimal solutions since the parking is roughly in the size range of the small simulation. The planned experiments are as follows:

1) *Complete exploration with different criterion descriptors*: This experiment is basically a replica of the simulations already presented, but we will also try the *TotalSum* criterion descriptor to properly illustrate the difference of the resulting plans and allocations. The objectives are to show the proper working in a static environment and to highlight the different results achieved by each criterion.

2) *Exploration with unexpected events and replanning*: This second experiment will show the reactivity and plausibility of this proposal when facing the apparition of new tasks and costs. The setup will include some unexpected obstacles in the lanes to be explored by the robots, causing the on-line creation of new primitive tasks. Objectives are thus the verification of proper working with dynamic plans (even if the environment is still static).

## VIII. CONCLUSIONS AND FUTURE WORK

The proposed simulated annealing algorithm simultaneously solves the planning and task allocation problems. It performs well in the tried examples, often finding the optimal solution for small-medium sized problems. In larger problems, it still outperforms the market-based solution for the examples studied. We believe that a replicated centralized algorithm is desirable in a critical mission, where we can not afford to distribute the information in the system so that losing a part of it would mean an irreparable loss —and failure.

We have also shown that the annealing solver is well fit for tackling the general MTSP variety, including constraints in the task model, such as tasks that need to be performed at certain points of the mission. This is naturally handled in the neighbor solution generation by means of infinite costs or avoiding the generation of bad solutions when feasible.

We also believe this is a general method applicable to a variety of real applications with tight interaction with the environment: i.e. urban applications, transportation, surveillance,

<sup>7</sup>EXPloration & REScue.



Fig. 7. Robot team in the testing scenario.

searching, guiding. All of these can benefit from features we can use with this solution: HTN modeling, heterogeneous agents (robots and infrastructure devices) and costs, task constraints, different optimization criteria and replanning.

Our market-based implementation illustrates that the *Min-Max* criterion on auction-based solutions is a viable alternative when we want to retain the advantages of market approaches. This kind of solutions can be used when we are constrained on CPU resources due to other computations being carried in the robots, or for quick update of plans when there is little time to react, since the annealing algorithm is slower in giving its best solutions.

Future plans include the research of the *MinMax* criterion in problems with reduced a priori information, the influence of other criterion descriptors, the annealing performance with more complex plan trees and larger numbers of primitive tasks and also with tightly constrained plans such as those needed in coordinated surveillance missions.

#### ACKNOWLEDGMENTS

The authors thank Norbert Ascheuer and Christoph Helmberg for his insightful help with the MTSP problem.

This project is partially funded by the Spanish MCYT-FEDER project DPI2003-07986.

#### REFERENCES

- [1] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–378, 2005.
- [2] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *ICRA'02*, 2002.
- [3] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," in *National Conference on Artificial Intelligence (AAAI 2004)*, July 2004.
- [4] S. Sachs, S. LaValle, and S. Rajko, "Visibility-based pursuit-evasion in an unknown planar environment," *The International Journal of Robotics Research*, vol. 23, no. 1, pp. 3–26, January 2004.
- [5] S. Burion, "Human detection for robotic urban search and rescue," Master's thesis, Carnegie Mellon University, 2004.
- [6] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive Computing*, pp. 24–33, October 2004.
- [7] R. M. Zlot and A. T. Stentz, "Complex task allocation for multiple robots," in *ICRA'05*. IEEE, April 2005.
- [8] B. P. Gerkey and M. J. Mataric, "Multi-robot task allocation: analyzing the complexity and optimality of key architectures," in *ICRA*, 2003, pp. 3862–3868.
- [9] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Intl. J. of Robotics Research*, vol. 23, no. 9, pp. 939–954, September 2004.
- [10] D. V. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *Journal of Intelligence Research*, vol. 16, pp. 389–423, June 2002.
- [11] M. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *IROS'04*, IEEE/RSSJ, Ed., vol. 1, 2004, pp. 698–705.
- [12] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, Oct. 2002.
- [13] M. B. Dias, R. M. Zlot, N. Kalra, and A. T. Stentz, "Market-based multirobot coordination: a survey and analysis," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-13, April 2005.
- [14] F. Zhang, W. Chen, and Y. Xi, "Improving collaboration through fusion of bid information for market-based multi-robot exploration," in *ICRA'05*, IEEE, Ed., 2005, pp. 1169–1174.
- [15] M. B. Dias and A. T. Stentz, "Opportunistic optimization for market-based multirobot control," in *IROS 2002*, September 2002.
- [16] K. Erol, J. A. Hendler, and D. S. Nau, "Htn planning: Complexity and expressivity," in *AAAI*, 1994, pp. 1123–1128.
- [17] R. M. Zlot and A. T. Stentz, "Market-based multirobot coordination using task abstraction," in *International Conference on Field and Service Robotics*, July 2003.
- [18] P. Stone and M. M. Veloso, "Task decomposition and dynamic role assignment for real-time strategic teamwork," in *Agent Theories, Architectures, and Languages*, 1998, pp. 293–308.
- [19] B. P. Gerkey, "On multi-robot task allocation," Ph.D. dissertation, University of Southern California, August 2003.
- [20] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *ICRA*, May 2004.
- [21] B. Brummit and A. T. Stentz, "GRAMMPS: A generalized mission planner for multiple mobile robots," in *ICRA'98*, May 1998.
- [22] C. Helmberg, "The m-cost ATSP," in *7th International IPCO Conference on Integer Programming and Combinatorial Optimization*. London, UK: Springer-Verlag, 1999, pp. 242–258.
- [23] I. Chlamtac, M. Conti, and J. J.-N. Liu, "Mobile ad hoc networking: imperatives and challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13–64, 2003.
- [24] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon decomposition," in *International Conference on Field and Service Robotics*, 1997.
- [25] D. T. Latimer, IV, S. Srinivasa, V. L. Shue, S. Sonne, H. Choset, and A. Hurst, "Towards sensor based coverage with robot teams," in *ICRA'02*. IEEE, May 2002.
- [26] G. Reinelt, *The traveling salesman: computational solutions for TSP applications*, ser. Lecture notes in computer science. Springer, 1994, vol. 840.
- [27] M. B. Dias, M. B. Zinck, R. M. Zlot, and A. T. Stentz, "Robust multirobot coordination in dynamic environments," April 2004.
- [28] I. Stojmenovic, "Geocasting with guaranteed delivery in sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 29–37, December 2004.
- [29] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 4598, no. 220, pp. 671–680, May 1983.
- [30] V. Cerny, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.
- [31] "Ada libre site." [Online]. Available: <https://libre2.adacore.com/>