18th IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 25-27 2018, Torres Vedras, Portugal

# Heterogeneous Multi-Agent Planning Using Actuation Maps

Tiago Pereira*†‡, Nerea Luis§, António Moreira*†, Daniel Borrajo§, Manuela Veloso‡ and Susana Fernandez§
*Faculty of Engineering, University of Porto, Portugal
Email: {tiago.raul, amoreira}@fe.up.pt
†INESC TEC, Porto, Portugal
‡Carnegie Mellon University, Pittsburgh, USA
§Universidad Carlos III de Madrid, Spain

*Abstract*—**Many real-world robotic scenarios require performing task planning to decide courses of actions to be executed by (possibly heterogeneous) robots. A classical centralized planning approach that considers in the same search space all combinations of robots and goals could lead to inefficient solutions that do not scale well. Multi-Agent Planning (MAP) provides a good framework to solve this kind of tasks efficiently. Some MAP techniques have proposed to previously assign goals to agents (robots) so that the planning effort decreases. However, these techniques do not scale when the number of agents and goals grow, as in most real world scenarios with big maps or goals that cannot be reached by subsets of robots. In this paper we propose to help the computation of which goals should be assigned to each agent by using Actuation Maps (AMs). Given a map, AMs can determine the regions each agent can actuate on. They help on alleviating the effort of MAP techniques knowing which goals can be tackled by each agent, as well as cheaply estimating the cost of using each agent to achieve every goal. Experiments show that when information extracted from AMs is provided to the Multi-Agent planner, goal assignment is significantly faster, speeding-up the planning process considerably. Experiments also show that this approach greatly outperforms classical centralized planning.**

## I. Introduction

Real-world robotic scenarios in which a set of robots need to solve a certain amount of tasks usually require the combination of path-planning and motion-planning techniques. An example of this type of scenarios is the coverage problem, which consists of distributing the space among the set of robots, so that each one explores a certain region of the environment. The problem is to plan and find a route for each robot so that all the feasible space is covered by the robots' actuators, while minimizing the execution time. Vacuum cleaning robots can be potential candidates for this type of planning problem. We assume we have a team of heterogeneous robots with different sizes. While the smallest robot can reach more areas, a bigger robot can clean more while traveling a smaller distance.

Nevertheless, other similar problems can also be solved with our contributed technique, such as heterogeneous robots executing vigilance tasks. As long as there exist some navigation graph where we can extract information to help the planner and agents with similar or different capabilities it will be a potential domain to solve with our approach.

We have encoded our problem as a Multi-Agent Planning (MAP) task. Automated planning is the field of Artificial Intelligence which deals with the computation of plans. The problem is modeled with the standard PDDL language [1]. For that purpose, we use a discrete representation of the map, i.e., a 2D grid of waypoints. The robots can move from one waypoint to another as long as they are grid neighbors and do not collide with obstacles. Moreover, robots can actuate other waypoints if their distance to the robot's current position is less than their actuation radius. The planner outputs a plan that accomplishes all feasible actuation tasks, by moving the robots to all the reachable locations from where they can actuate the goal waypoints. Given the robot heterogeneity, some tasks might only be feasible for a subset of the robots.

From a MAP point of view, the multi-robot problem we propose to solve forces us to deal with two issues regarding performance of the planning process: (1) the size of the search space grows with the number of waypoints and goals; and (2) some goals are not feasible for some robots. On one hand, real-world scenarios are big enough to make almost impossible for a planner to solve this problem in a reasonable amount of time by just assigning all goals to all agents (following a centralized planning approach). On the other hand, some Multi-Agent planners invoke a goal-allocation phase before starting to plan, computing smaller individuals plans [2]. However, during goal allocation, a relaxed plan is computed per goal and robot to either return an estimated cost or identify unfeasibility.

Therefore, to obtain path-planning related information we contribute a methodology that combines Actuation Maps (AMs) [3] with a multi-agent planner, using these maps as a pre-processing step to speed-up the goal assignment phase. These maps are built only once before the beginning of the planning process, one per robot, at a very low cost in comparison to the impact on time savings observed later in goal assignment. Following the idea of the relaxed plan computation during planning, our architecture computes an estimated cost before-hand regarding the information that can be directly extracted from AMs. As a result, the planner receives the estimated cost information as input, and saves time by avoiding the relaxed plan computation.

Domain-independent planning might not scale up well. However, in most real-world domains, practitioners can benefit from some domain-dependent knowledge and the challenge is how to efficiently use this knowledge to allow planners

18th IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 25-27 2018, Torres Vedras, Portugal

to scale up. We take advantage of the fact that most robotic tasks deal with a map where robots have to carry out several activities, and that computing AMs can be done very cheaply. If we provide that knowledge as input to a domain-independent MAP system that performs task allocation, the integrated system can scale up to real hard tasks.

## II. Planning with Actuation Maps

As previously mentioned, this work combines Actuation Maps (AM) with Multi-Agent Planning (MAP). The contributed architecture can be seen in Figure 1.

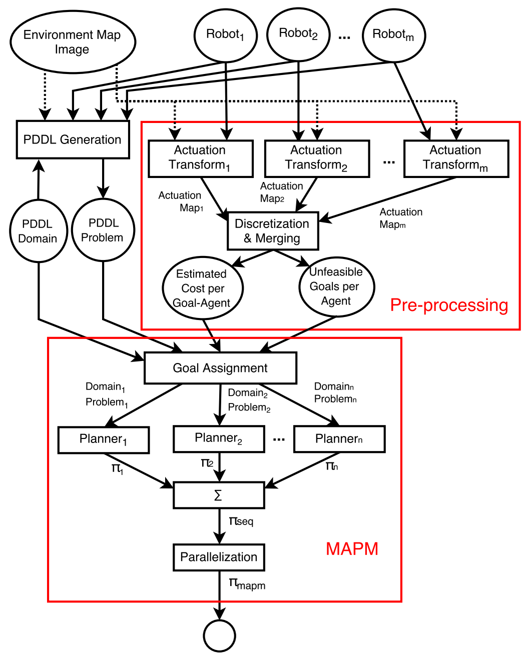


Figure 1. Architecture for combining AMs and Multi-Agent Planning

Our system receives as input a *Map Image* which represents a 2D environment (e.g. building floor plan), $m$ *Robot* models with the agent's features, and a *PDDL domain* that represents the type of planning task being solved (e.g. coverage problem). According to the *PDDL domain* description, the *PDDL generation* node takes as input the image file with the map and the robot models in order to generate a *PDDL Problem*. The *PDDL generation* could use the default pixel resolution when building the respective 2D grid of waypoints, but we assume it can also sample it down for a more light-weight representation of the environment.

By using the MAP technique from [4], the planning architecture would be equivalent to the bottom part of Figure 1. The planner would take the PDDL problem and domain files, and execute *Goal-Assignment* (GA). In this first step, a relaxed problem is computed for all agent-goal pairs in order to obtain an estimate cost. Then, goals are assigned to each robot using that estimated cost and following a pre-determined GA strategy. Agents plan individually with their own problem, returning a potential solution (plan) in the end. All plans are first merged into one single plan and then the solution is parallelized, resulting in plan $\pi_{mapm}$.

With that architecture, the time spent on planning depends highly on the GA efficiency. Given that it has to solve a relaxed problem for each agent-goal pair, repeating similar searches multiple times, this method does not scale well with big maps and a high number of robots. Therefore, we contribute a pre-processing step, which uses the *Actuation Maps* to speed-up GA. AMs are generated once before planning, determining which regions are actuation-feasible for each one of the robots, and also providing a cheap estimate of the cost of using each agent to achieve each goal. The *Discretization* node converts the pixel-based cost estimate into a grid-based cost, with the same sampling rate used by the *PDDL generation* node.

Using the low cost estimates from the individual AMs, the *Merging* node identifies the pairs robot-goal that are unfeasible, compiling lists of both *Unfeasible Goals per Agent* and the overall list of unfeasible goals. The *Merging* also compiles a list with the *Estimated Cost per Goal-Agent*, saving time by alleviating the efforts of MAP during GA. Because the AMs are determined only once before planning, a greater impact from pre-processing is expected for large sets of goals.

In this work we consider teams of circular robots that actuate in a 2D environment, where the world is represented by a 2D image that can be downsampled to a grid of waypoints. The AM gives information about the actuation capabilities of each robot, as a function of robot size and initial position [3]. In the vacuum cleaning robots example, AM represents the regions of the world each robot can clean.

### A. Problem Formulation

The kind of problems that can be solved using our approach are identified by the following features: these problems always have a navigation graph and a set of potential tasks to be executed by an agent on each node of the graph (e.g. Rovers or Transport domain). Thus, in these domains we can generate the AMs and extract specific information to help the planner solve more quickly the planning problems.

A single-agent planning task can be formally defined as a tuple $\Pi = \{F, A, I, G\}$, where $F$ is a set of propositions, $A$ is a set of instantiated actions, $I \subseteq F$ is an initial state, and $G \subseteq F$ is a set of goals. We consider a MAP formalization where a set of $m$ agents, $\Phi = \{\phi_1, \ldots, \phi_m\}$, has to solve the task $\Pi$. We define the Multi Agent Planning (MAP) task as a set of planning subtasks, one for each agent, $M = \{\Pi_1, \ldots, \Pi_m\}$ where $M$ refers to the MAP task. Each planning subtask $\Pi_i$ includes only the facts, actions, goals and initial state related to the agent $\phi_i$. For representation convenience, an equivalent lifted representation of each single-agent planning task in PDDL would be a pair (domain, problem): $\Pi_i = \{D_i, P_i\}$, as it is reflected in Figure 1 as output after the goal assignment (GA) step.

In order to use MAP techniques, we modeled the domain and problem using PDDL. The domain has two types of objects: robots, which act as agents; and waypoints, which represent positions in the discretized world. The goal of the problem is a list of waypoints to actuate on (positions that need to be covered). The PDDL domain has four predicates:

- **At** (robot, waypoint): defines the robot position;
- **Connected** (robot, waypoint, waypoint): establishes connectivity between waypoints;
- **Reachable** (robot, waypoint, waypoint): shows waypoints actuated by robot located on specific waypoint;
- **Actuated** (waypoint): indicates which waypoints were already actuated; this predicate is used to specify goals.

Robots have to actuate every waypoint in the goal list of the PDDL problem, as long as the task is feasible. Robots can navigate through two connected waypoints or just actuate another waypoint from their current waypoint position. Each robot has its individual navigability graph of waypoints.

Finally, the two actions that are defined in the domain are called *navigate* and *actuate*. The first one moves a robot from its current waypoint location to a neighbor waypoint. The second action is used to mark a waypoint as actuated if it is ***reachable*** from the robot's current waypoint location.

*Navigate* and *actuate* are the two actions that can be executed by an agent when it is placed on a waypoint. Both *navigate* and *actuate* have as effect the predicate ***actuated***.

We do not address any kind of collision between robots or capacity for the path between two waypoints. We assume collisions can be resolved with post-processing replanning.

## B. Multi-Agent Planner

In order to run the problem we have developed the Multi-Agent planner called MA-Plan Merger (MAPM). The main sequence of steps of this algorithm follows the structure presented in [4]. However the structure has been improved to use information from AMs. MAPM processes inputs from AMs to skip the computation of relaxed plans during GA.

The pre-processing inputs to MAPM is the list of estimated costs $EC = \{(g, \phi_i, c) \mid g \in G, \phi_i \in \Phi, c = C(g, \phi_i)\}$ such that $c$ is computed as the number of steps for an agent to reach the goal position $g$ from its initial position, shown in Figure 1 as the *Estimated Cost per Goal-Agent* node. The cost of navigating between two neighbor grid waypoints is 1 unit. MAPM also receives as input a MAP task, which consists of a PDDL domain and problem files (Figure 2). The PDDL problem file includes the list of agents. In addition a goal-assignment strategy (GAS) needs to be chosen to define the way goals are assigned to agents by the system.

```
Algorithm MAPM ($\Pi, GAS, EC, P$)
Inputs: $M$, $GAS$, $EC$, $P$
Output: $\pi_{mapm}$

1  $\Phi', M'$ = goal-assignment($M, GAS, EC$)
2  Forall $\phi'_i \in \Phi'$ do $\pi_i$ =plan($\Pi_i, P$)
3  $\pi_{seq}$ =merge($\pi_1, \dots, \pi_n$)      /* where $n = |\Phi'|$ */
4  If valid($\pi_{seq}$)
5  Then return $\pi_{mapm}$ =parallelize($\pi_{seq}$)
6  Else return no plan
```

Figure 2. High level description of the MAPM algorithm. Inputs: MAP task ($M$), Goal Assignment Strategy ($GAS$), Estimated Cost per robot-goal ($EC$), Single-Agent Planner ($P$). Output: resulting plan ($\pi_{mapm}$).

The first step of the MAPM algorithm is to allocate the feasible goals to the agents (line 1). This step uses the information of estimated costs received from AMs. Goal assignment phase (GA) returns as output (1) a subset of $\Phi'$ agents, $\Phi' = \{\phi_1, \dots, \phi_n\}$, that will be the only ones who will plan to solve the problem; and (2) a new MAP task $M' = \{\Pi_1, \dots, \Pi_n\}$. As a result, a specific PDDL domain and problem will be generated for each $\phi'_i$ agent which only includes the goals each agent has to achieve. If a goal is unfeasible for all the $\Phi'$ agents, MAPM will discard it from the new MAP task $M'$. In order to perform task allocation [5] some strategy has to be determined or implemented, as the aim is to improve the efficiency of the planning process afterwards. Load-Balance strategy, defined in [6], assigns each goal $g \in G$ to the best agent $\phi_i \in \Phi$, but it does not assign to an agent more goals than the ratio of total goals divided by number of agents. The Load-Balance assignment strategy is used when minimizing the maximum number of actions per agent (equivalent to minimizing makespan). As a second option, we also took the Best-Cost strategy also defined in [6], which assigns each goal to the agent that can achieve it with the least cost. The Best-Cost strategy is used when minimizing the total number of actions over all robots (plan length).

After the GA phase, as in [4], each agent in $\phi'$ builds its plan individually (line 2). Then the agents' plans are merged by a simple concatenation of plans in one resulting plan (line 3). If the merged plan is valid, we parallelize the plan (line 5) generated by the merging step. Parallelization is performed in two steps: converting the input total-order plan into a partial-order one by a similar algorithm to [7]; and parallelizing this partial-order plan by ordering actions in the first time step that satisfies all ordering constraints in the partial-order plan.

To set up MAPM, an off-the-shelf planner had to be chosen. Our configuration of MAPM uses LAMA-UNIT-COST as the planner $P$ of the algorithm. LAMA-UNIT-COST corresponds to the first search that LAMA performs, using greedy-best-first with unit costs for actions [8]. The merged plan is validated using VAL [9], from the International Planning Competition[1].

## C. Actuation Space

We briefly summarize here the process of building the Actuation Space [3]. We assume there is an occupancy grid map. In this image each pixel has a value with the probability of the corresponding world position being occupied by an obstacle. This occupancy grid map is first transformed into a binary image using a fixed threshold.

We define $\mathcal{G}$ as the set with all pixel positions from the input binary image. This input image (*Environment Map Image*

[1]http://icaps-conference.org/index.php/main/competitions

in Figure 1) is represented by $\mathcal{M}$, the set with the obstacle pixel positions. We define the structuring element as an image that represents the robot shape. Thus $\mathcal{R}_i$ is the set with pixel positions from a circle with radius equal to the robot size. The morphological dilation on the obstacle set $\mathcal{M}$ by $\mathcal{R}_i$ is:

$$\mathcal{M} \oplus \mathcal{R}_i = \bigcup_{\mathbf{r} \in \mathcal{R}_i} \mathcal{M}_{\mathbf{r}} \tag{1}$$

where $\mathcal{M}_{\mathbf{r}}$ is the translation of $\mathcal{M}$ by vector $\mathbf{r}$.

The visual output of applying this dilation operation to a map of obstacles is the inflation of obstacles by the robot size. The free configuration space, $\mathcal{C}^{free}$, is then defined as:

$$\mathcal{C}_i^{free} = \{\mathbf{p} \in \mathcal{G} \mid \mathbf{p} \notin \mathcal{M} \oplus \mathcal{R}_i\} \tag{2}$$

where $\mathcal{G}$ is the set with all the pixel positions. The free configuration space represents the feasible robot positions.

In order to determine the Actuation Space instead, the partial morphological closing operation is used. The partial morphological closing was introduced in order to consider the initial robot position when determining the Actuation Space. In order to use the partial morphological closing, the algorithm needs to find the navigable regions first. The set of navigable regions from a starting robot point $\mathbf{r}_i^0$ is

$$\mathcal{L}_i(\mathbf{r}_i^0) = \{\mathbf{p} \in \mathcal{G} \mid \mathbf{p} \text{ connected to } \mathbf{r}_i^0 \wedge \mathbf{p} \in \mathcal{C}_i^{free}\} \tag{3}$$

The navigable set $\mathcal{L}_i(\mathbf{r}_i^0)$ is the set of points that are connected to the initial position $\mathbf{r}_i^0$ through a path of adjacent cells in the free configuration space.

Applying a second dilation operation to the navigable set (subset of $\mathcal{C}^{free}$) instead of applying it to the free configuration space, we obtain the partial morphological closing operation. The Actuation Space is the dilation of the navigable space:

$$\mathcal{A}_i(\mathbf{r}_i^0) = \mathcal{L}_i(\mathbf{r}_i^0) \oplus \mathcal{R}_i \tag{4}$$

In Figure 3 we show a simulated map with 2 robots with different sizes, and the respective Actuation Spaces.

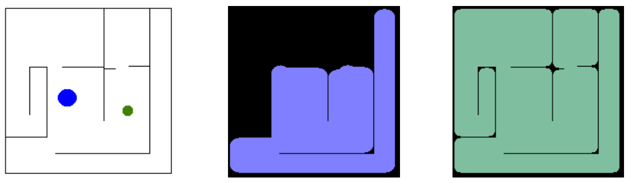

(a) Map & 2 Robots (b) Actuation Space 1 (c) Actuation Space 2

Figure 3. Simulated map and two heterogeneous robots with different sizes in (a); colored regions in (b) and (c) represent actuation spaces for respective robots, i.e. the points in the environment that each robot can actuate, depending on their size and initial position shown in (a).

## D. Actuation Map and Grid Downsampling

When converting the original map and the Actuation Space to the PDDL description, it is possible to consider each individual pixel as a waypoint in a grid with the size of the whole image. However, that approach results in a high density of points that makes the planning problem excessively complex. We reduce the set of possible locations by downsampling the grid of waypoints. The downsampling rate $s_r$ is set manually. If the original pixel resolution is used, the resulting grid of waypoints $\mathcal{G}'$ contains all pixels and is equivalent to $\mathcal{G}$, otherwise it represents the grid after downsampling.

Using the Actuation Space it is possible to find the Estimated Cost (EC) list. For that purpose, we contribute the following extension. We build the navigable space $\mathcal{L}_i$ in an iterative procedure, from the starting position $\mathbf{r}_i^0$. In the first iteration we have $\mathcal{L}_i^0(\mathbf{r}_i^0) \leftarrow \{\mathbf{r}_i^0\}$, and then:

$$\begin{aligned}\mathcal{L}_i^j(\mathbf{r}_i^0) = \{\mathbf{p} \in \mathcal{G} \mid \exists \mathbf{q} \in \mathcal{L}_i^{j-1}(\mathbf{r}_i^0) : \mathbf{p} \text{ neighbor of } \mathbf{q} \\ \wedge \mathbf{p} \in \mathcal{C}_i^{free} \wedge \mathbf{p} \notin \mathcal{L}_i^a(\mathbf{r}_i^0) \quad \forall a < j\}\end{aligned} \tag{5}$$

When using this recursive rule to build the navigable space, we guarantee that any point in the set $\mathcal{L}_i^j(\mathbf{r}_i^0)$ is exactly at distance $j$ from the initial position $\mathbf{r}_i^0$. If we build the actuation space sets with the intermediate navigable sets $\mathcal{L}_i^j(\mathbf{r}_i^0)$,

$$\mathcal{A}_i^j(\mathbf{r}_i^0) = \mathcal{L}_i^j(\mathbf{r}_i^0) \oplus \mathcal{R}_i \tag{6}$$

then the intermediate actuation set $\mathcal{A}_i^j(\mathbf{r}_i^0)$ represents the points that can be actuated by the robot from positions whose distance to $\mathbf{r}_i^0$ is $j$. The actuation space defined in the previous section can also be alternatively defined as

$$\mathcal{A}_i(\mathbf{r}_i^0) = \{\mathbf{p} \in \mathcal{G} \mid \exists a : \mathbf{p} \in \mathcal{A}_i^a(\mathbf{r}_i^0)\} \tag{7}$$

The Actuation Map is defined for $g \in \mathcal{A}_i(\mathbf{r}_i^0)$:

$$\mathcal{AM}_i(\mathbf{r}_i^0, g) = \min\{j \mid g \in \mathcal{A}_i^j(\mathbf{r}_i^0)\} + 1 \tag{8}$$

The Actuation Map $\mathcal{AM}_i(\mathbf{r}_i^0, g)$ represents, for each $g \in \mathcal{A}_i(\mathbf{r}_i^0)$, the minimum number of actions needed for the robot to actuate the grid waypoint $g$ if starting from the initial position $\mathbf{r}_i^0$, measured in the pixel-based grid $\mathcal{G}$. In equation 8, the minimum $j^*$ represents the minimum distance (number of *navigate* actions) needed to travel from $\mathbf{r}_i^0$ to some point from where $g$ can be actuated. The added one in equation 8 accounts for the *actuate* action needed to actuate $g$, after the $j^*$ *navigate* actions to reach a place where the robot actuates $g$.

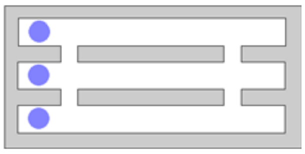
Finally, the *Estimated Cost per Goal-Agent* list $EC$ is

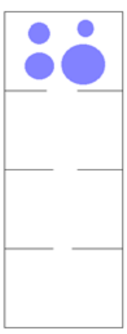
$$\begin{aligned}EC = \{\langle g, \phi_i, \text{cost}\rangle \mid g \in \mathcal{G}' \wedge \phi_i \in \Phi \\ \wedge g \in \mathcal{A}_i(\mathbf{r}_i^0) \wedge \text{cost} = \text{ceil}\left(\mathcal{AM}_i(\mathbf{r}_i^0, g)/s_r\right)\}\end{aligned} \tag{9}$$

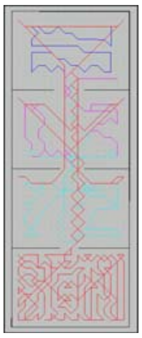
where $s_r$ is the downsampling rate. The division by $s_r$ transforms the estimated cost of actions measured in the pixel-based grid $\mathcal{G}$, $\mathcal{AM}_i(\mathbf{r}_i^0, g)$, to the respective cost in the downsampled grid of waypoints $\mathcal{G}'$. The *ceil* function rounds the result to the smallest integral that is not less than $\mathcal{AM}_i(\mathbf{r}_i^0, g)/s_r$.

## III. EXPERIMENTS AND RESULTS

In this section we show some experiments that test the impact of the pre-processing on the MAPM performance. We have modeled three different scenarios that include up to four agents with different sizes, and thus different actuation capabilities. Planning results are shown using as metrics the time in seconds, the length of the resulting plan and the makespan. The makespan is the length of the parallel plan (number of execution steps, where several actions can be executed at the same execution step). We designed three different scenarios, shown in Figure 4, each one with two levels of waypoint density (H, the higher, and L, the lower).


(a) Mutual Exclusive (450×220)


(b) Corridor (500×200)


(c) Corridor paths (500×200)

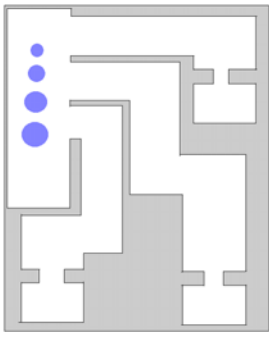

(d) Extremities (600×500)

Figure 4. Maps of the three scenarios used in the experiments. Grey regions represent out-of-reach regions that can nevertheless contain goal waypoints, which are unfeasible for all the robots. Robots are represented with blue circles positioned in the region of their starting position. Solution for the Corridor scenario is also presented, with the paths for the four robots.

Four different configurations of MAPM have been set up:

- MAPM Load-Balance (LB) with estimated-cost information (EC). EC refers to the configuration that combines Actuation Maps and MAP.
- MAPM Best-Cost (BC) with estimated-cost information (EC), also combining Actuation Maps and MAP.
- MAPM LB, same as before but without EC information.
- MAPM BC same as before but without EC information.

Furthermore, the following state-of-the-art planners have been chosen as a comparison baseline:

- ADP [10], a planner that automatically detects agents.
- LAMA [8], winner of IPC 2011.
- YAHSP [11], a state-of-the-art centralized planner.

The maximum time spent on the pre-processing for any scenario was 170 milliseconds, for the Extremities problem with 4 robots. We included the pre-processing times (to generate the AMs) in the GA column of Table I, and in the total time in Table III. Hardware used for running the planner was IntelXeon 3,4GHz QuadCore 32GB RAM. AMs were computed using a 2.5GHz DualCore 6GB RAM.

Tables I and II are shown to prove the remarkable impact that information from Actuation Maps (AMs) has in combination with MAPM. Goal assignment (GA) times in Table I are minimal in comparison with the ones from Table II when the planner needs to compute the relaxed plans for every goal-agent pair before starting to plan for the solution.

Total time results for each planner in each one of the six problems are shown in table III. A maximum of two hours was given to each planner to solve each scenario. YAHSP results do not appear in the tables because it could not solve any of the scenarios. Regarding total time, the fastest configuration is MAPM-LB-EC if all total times are summed up. The impact of combining information from actuation maps with MAP can be easily appreciated by comparing MAPM-LB-EC and MAPM-LB.

Table I
TIME RESULTS IN SECONDS FOR MAPM LOAD-BALANCE CONFIGURATION WITH ESTIMATED COST INFORMATION. TOTAL TIME, GOAL ASSIGNMENT, INDIVIDUAL PLANNING AND PARALLELIZATION TIMES.

| MAPM-LB-EC | | | | |
|---|---|---|---|---|
| Name | TOTAL(s) | GA | Planning | Parallel |
| CorridorH | 33.58 | 0.64 | 24.37 | 8.57 |
| CorridorL | 6.18 | 0.26 | 4.62 | 1.30 |
| ExtremH | 602.68 | 3.06 | 428.28 | 171.34 |
| ExtremL | 58.32 | 0.92 | 40.93 | 16.47 |
| MutExH | 7.39 | 0.34 | 5.03 | 2.02 |
| MutExL | 1.39 | 0.12 | 1.04 | 0.23 |

Table II
TIME RESULTS IN SECONDS FOR MAPM LOAD-BALANCE CONFIGURATION WITHOUT ESTIMATED COST INFORMATION. TOTAL TIME, GOAL ASSIGNMENT, INDIVIDUAL PLANNING AND PARALLELIZATION TIMES.

| MAPM-LB | | | | |
|---|---|---|---|---|
| | TOTAL(s) | GA | Planning | Parallel |
| CorridorH | 1232.97 | 1204.20 | 20.88 | 7.89 |
| CorridorL | 128.78 | 123.59 | 4.10 | 1.09 |
| ExtremH | timeout | | | |
| ExtremL | 3870.00 | 3823.75 | 32.89 | 13.36 |
| MutExH | 903.65 | 896.82 | 4.81 | 2.02 |
| MutExL | 69.41 | 68.19 | 0.98 | 0.24 |

Table IV shows results regarding the plans' length and also results regarding makespan. As ADP does not return a makespan metric, we have applied our parallelization algorithm to transform its total ordered plans into partial order plans so that we can fairly compare the results.

The best configuration overall regarding Plan Length is MAPM-BC-EC (called M-BC-E in the table). Regarding makespan in Table IV, the Load-Balancing strategy is better, as expected. Moreover, MAPM-LB-EC (M-LB-E) configuration is the best one for problems with higher density of waypoints, while MAPM-LB (M-LB) proves to be better for reducing makespan in low density problems. This can be explained by the discretization errors from equation 9, which are greater when the down sampling rate is bigger. These errors result in slightly inaccurate cost estimates that change goal assignment.

## IV. RELATED WORK

A MAP approach that als uses a pre-processing step is the automated agent decomposition for multi-robot task planning [12]. In that work the pre-processing step exploits decompositions of the problem in domains with a lower level of interaction, boosting the final performance. ADP [10] is also

Table III
TOTAL TIME RESULTS IN SECONDS. MAPM WITH ESTIMATED-COST INFORMATION IN LOAD-BALANCE; MAPM WITHOUT ESTIMATED COST INFORMATION IN LB; MAPM WITH ESTIMATED COST INFORMATION IN BEST-COST; MAPM WITHOUT ESTIMATED COST INFORMATION IN BC; ADP AND LAMA.

| | **Total Time including Pre-Processing (s)** | | | | | |
|---|---|---|---|---|---|---|
| | MAPM-LB-EC | MAPM-LB | MAPM-BC-EC | MAPM-BC | ADP | LAMA |
| CorridorH | **33.58** | 1232.97 | 41.53 | 2839.79 | *mem.limit* | *timeout* |
| CorridorL | 6.18 | 128.78 | 9.07 | 135.58 | 104.47 | **5.75** |
| ExtremH | **602.68** | *timeout* | 1788.69 | *timeout* | *mem.limit* | *timeout* |
| ExtremL | **58.32** | 3870.00 | 112.03 | 3929 | *mem.limit* | *timeout* |
| MutExH | 7.39 | 903.65 | 7.27 | 910.18 | **5.54** | 6.29 |
| MutExL | 1.39 | 69.41 | 1.38 | 72.24 | **0.84** | 1.12 |

Table IV
PLAN LENGTH AND MAKESPAN: MAPM WITH ESTIMATED-COST INFORMATION IN LOAD-BALANCE; MAPM WITHOUT ESTIMATED COST INFORMATION IN LB; MAPM WITH ESTIMATED-COST IN BEST-COST; MAPM WITHOUT ESTIMATED COST INFORMATION IN BC; ADP AND LAMA.

| | **Plan Length** | | | | | | **Makespan** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M-LB-E | M-LB | M-BC-E | M-BC | ADP | LAMA | M-LB-E | M-LB | M-BC-E | M-BC | ADP | LAMA |
| CorridorH | 1289 | 1268 | 1226 | **1136** | | | **403** | 408 | 461 | 734 | | |
| CorridorL | 605 | 598 | 588 | 475 | 1403 | **470** | 219 | **189** | 303 | 458 | 1313 | 286 |
| ExtremH | 3428 | | **3116** | | | | **1453** | | 1929 | | | |
| ExtremL | 1490 | 1587 | 1365 | **1233** | | | 556 | **511** | 928 | 1140 | | |
| MutExH | **642** | **642** | **642** | **642** | 748 | **642** | **116** | **116** | **116** | **116** | 162 | **116** |
| MutExL | **277** | **277** | **277** | **277** | 278 | **277** | **59** | **59** | **59** | **59** | 60 | **59** |

related with that decomposition work. Our approach factorizes the problem regarding goals and agents, creating independent subtasks for each agent, but the domain is never changed.

The methodology that uses morphological operations in order to build the Actuation Maps was previously introduced [3]. It has been shown that similar transformation can be used to obtain Actuation Maps for any-shape robots as well [13]. Other relevant multi-robot planning problem in robotics is inspection, which searches for paths that can perceive a set of targets, where neural network solutions have been proposed [14].

There are similar environments to our problem defined in previous planning domains, like *VisitAll* and *Rovers* domain.

## V. CONCLUSIONS AND FUTURE WORK

In this paper we showed how to combine information from Actuation Maps with Multi-Agent Planning to solve a multi-robot problem more efficiently. We used a pre-processing step to determine feasibility of robot-goal pairs and to extract an estimated cost, used later to avoid computing relaxed plans during goal assignment. Total solving times were significantly improved when pre-processing information was provided to MAPM. As future work, we would like to extend the pre-processing technique to other domains and robot models.

## ACKNOWLEDGMENT

This work is financed by the ERDF through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by FCT (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013, and FCT grant SFRH/BD/52158/2013, and partially by MICINN projects TIN2008-06701-C03-03, TIN2011-27652-C03-02 and TIN2014-55637-C2-1-R.

## REFERENCES

[1] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *JAIR*, vol. 20, 2003.

[2] D. Borrajo and S. Fernández, "MAPR and CMAP," in *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*, Jerusalem (Israel), 2015. [Online]. Available: http://agents.fel.cvut.cz/codmap/results/CoDMAP15-proceedings.pdf

[3] T. Pereira, M. Veloso, and A. Moreira, "Multi-robot planning using robot-dependent reachability maps," in *Robot 2015: Second Iberian Robotics Conference*. Springer, 2015, pp. 189–201.

[4] N. Luis and D. Borrajo, "Plan Merging by Reuse for Multi-Agent Planning," in *Proceedings of the 2nd ICAPS Distributed and Multi-Agent Planning workshop (DMAP)*, 2014.

[5] V. Conitzer, "Comparing multiagent systems research in combinatorial auctions and voting," *AMAI*, vol. 58, no. 3-4, pp. 239–259, 2010.

[6] D. Borrajo, "Plan sharing for multi-agent planning," in *Proceedings of the ICAPS'13 DMAP Workshop*, R. Nissim, D. L. Kovacs, and R. Brafman, Eds., 2013, pp. 57–65.

[7] M. M. Veloso, M. A. Pérez, and J. G. Carbonell, "Nonlinear planning with parallel resource allocation," in *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*. Morgan Kaufmann, 1990, pp. 207–212.

[8] S. Richter and M. Westphal, "The LAMA planner: Guiding cost-based anytime planning with landmarks," *JAIR*, vol. 39, pp. 127–177, 2010.

[9] R. Howey, D. Long, and M. Fox, "VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL," in *ICTAI 2004*, 2004, pp. 294–301.

[10] M. Crosby, "Adp an agent decomposition planner," *Proceedings of the CoDMAP-15*, p. 4, 2015.

[11] V. Vidal, "YAHSP3 and YAHSP3-MT in the 8th International Planning Competition." in *In 8th International Planning Competition (IPC-2014)*, 2014, pp. 64—-65.

[12] M. Crosby, M. Rovatsos, and R. P. Petrick, "Automated agent decomposition for classical planning." in *ICAPS*, 2013.

[13] T. Pereira, M. Veloso, and A. P. Moreira, "Visibility maps for any-shape robots," in *IROS'16, the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[14] J. Faigl, "Approximate solution of the multiple watchman routes problem with restricted visibility range." *IEEE transactions on neural networks*, vol. 21, no. 10, pp. 1668–79, 2010.