Map Partitioning to Approximate an Exploration Strategy in Mobile Robotics

Guillaume Lozenguez, Lounis Adouane, Aurélie Beynier, Abdel-Illah Mouaddib, Philippe Martinet

Map partitioning to approximate an exploration strategy in mobile robotics

Guillaume Lozenguez[a,b,*], Lounis Adouane[b], Aurélie Beynier[c], Abdel-Illah Mouaddib[a] and Philippe Martinet[d]
[a]*GREYC, Campus Côte de Nacre, Caen, France*
[b]*LASMEA, Campus des Cézeaux, Aubiere, France*
[c]*LIP6, Université Pierre and Marie Curie, Paris, France*
[d]*IRCCYN, Nantes, France*

Abstract. In this paper, an approach is presented to automatically allocate a set of exploration tasks between a fleet of mobile robots. The approach combines a *Road-Map* technique and Markovian Decision Processes (MDPs). The addressed problem consists of exploring an area where a set of points of interest characterizes the main positions to be visited by the robots. This problem induces a long term horizon motion planning with a combinatorial explosion. The *Road-Map* allows the robots to represent their spatial knowledge as a graph of way-points connected by paths. It can be modified during the exploration mission requiring the robots to use on-line computations. By decomposing the *Road-Map* into regions, an MDP allows the current group leader to evaluate the interest of each robot in every single region. Using those values, the leader can assign the exploration tasks to the robots.

Keywords: Multi-robot cooperation, exploration task, graph partitioning and markovian decision process

1. Introduction

The problem of exploring an environment with a fleet of robots is a persistent topic in mobile robotics [10]. It is a challenging problematic to obtain an on-line and accurate decision process embedded on each robot of the fleet. In fact, it is difficult to consider a problematic in a long term horizon to produce a complete strategy of navigation involving all the exploration tasks. Traditional approaches do not separate decision making from the robot control loop (perception, process and control) and has to produce several responses per second. They propose heuristic strategy as, for example, reaching the closest position in the frontier of exploration for mapping missions [10] or in the set of unvisited positions for search missions [24]. In multi-robot exploration, the proposed approaches [10,19,23] are increased to taking into account the other robots position and/or wished. Due to the frequency of map actualization and the limited time of the control loop, fast selection processes decides only in short term horizons.

Hierarchical control architecture [1,26] could permit to save resources for decision process by splitting the robot control into a functional and a deliberative levels with different execution times. This way, the notion of *Road-Map* (a topological map increased by metric informations [18]) appears as an interesting

*Corresponding author: Guillaume Lozenguez, GREYC, Campus Côte de Nacre, Bd Maréchal Juin, BP 5186, 14032 Caen, France. E-mail: guillaume.lozenguez@unicaen.fr.

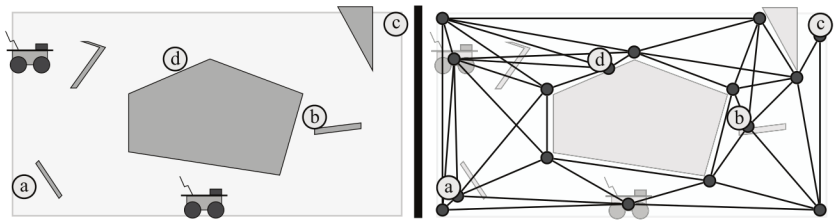

Fig. 1. An exploration problem with 4 points of interest $\{a,\ b,\ c,\ d\}$ (left) and its attached *Road-Map*(right). This example involves 2 robots which are close enough to communicate.

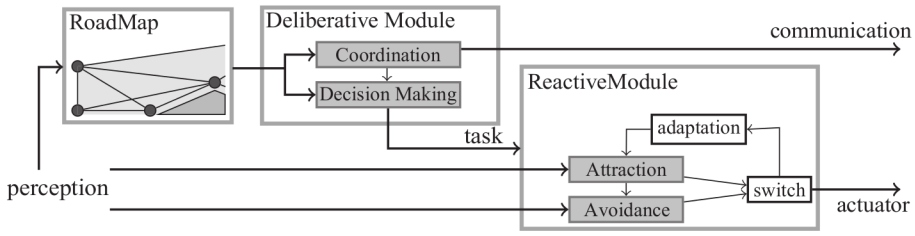

Fig. 2. The robot architecture schema.

tool to connect between reactive control and decision making [3]. Supervising a reactive control while lacking knowledge impose the concern about uncertainty in the action achievement. The approach given in this paper proposes to use a *Road-Map* with on-line stochastic decision making while considering a long-term horizon.

Our research is employed in a project named R-Discover[1] that aims at exploring an external area using a fleet of mobile robots. An Unmanned Aerial Vehicle (UAV) takes several pictures of an area and then a fleet of ground robots are set to refine knowledge about this area. Indeed, the pictures allow us to build a first map regarding the environment. Next, a human operator defines key positions which must be visited to increase the map definition as a set of points of interest in the map (Fig. 1). The UAV's autonomy and communication constraints in addition to the need of pictures refinement do not permit constant collaboration between the UAV and the ground robots.

The main issue addressed in this paper is linked to the robots' capacities to cooperate in order to calculate and adapt exploration strategies along the mission. Robots are equipped with communication devices efficient only in a given radius. A hierarchy is defined with between the robots; robots with higher capacity to lead the fleet are given a higher identifier (ID). The robot with the higher ID in sub-groups of robots is defined as the leader. This way, several times during the mission, two or more robots can communicate, and the current leader robot between them, needs to dispatch mission parts in few seconds. Here the mission is defined by the local set of points of interest to visit $I = \{a, b, c, \ldots\}$ shared by the communicating robots (Fig. 1). Otherwise, the robots evolute alone and actualize on-line their decision making regarding their allocated points of interest while the *Road-Map* is modified.

Allocating the set of points of interest is computed in a way that maximizes the sum of individual expected gains. The complexity of evaluating the interest of a robot in visiting a sub-set of points of interest does not permit an optimal computation of a solution for problems where $|I| > 20$ (Section 1.2). The idea of environment map partitioning permits to obtain fast path planning [4]. To increase the number of

[1]Supported by the National Research Agency of France (ANR).

points of interest considered in on-line computing, the *Road-Map* is quickly partitioned. That allows the robots to plan and reason over regions instead of all the set of points of interest.

The following of this article is organized as following: Section 1 presents the used architecture for the robot decision making. Section 2 details a greedy map partitioning and the hierarchical computation of multi-robot strategy in order to perform an exploration mission. Before ending with a discussion regarding the related work and a conclusion, the work is evaluated in Section 3.

2. The context

During the mission, when a group of robots are close enough to efficiently communicate between them a communication phase starts to lead the re-allocation of the points of interest. In the following, the term robots will involve the communicating robots which are a sub-group of the fleet of robots. The leader robot (which has the higher level identifier in a predefined hierarchy) allocates the points of interest to the robots, then, in execution phases, each robot executes alone its allocated part of the mission. The mission is initialized by a communication phase involving all the robots and all the points of interest given by the operator with the map. The proposed architecture (Fig. 2) allows us to separate the robot locomotion problem from the deliberative aspect. This way, more focus is assigned on the problem of allocating the points of interest to optimize the robots movements.

2.1. The robot's control architecture

Each robot control architecture is composed of two modules in the manner of the hierarchical architectures [1,26]. The first module is reactive and permits the robot to move between two positions while avoiding obstacles. The second module is deliberative and aims at organizing the tasks before submitting the current task to perform to the reactive module (Fig. 2). The reactive module controls movements according to the events of low importance which does not necessitate map modification. A hybrid multi-controller architecture [2] is used for the navigation of the mobile robot in cluttered environments. This architecture is based on a flexible switching between different atomic controllers as attraction to a target or obstacle avoidance.

Developing the reactive module is out of the paper scope. In fact, we are interested in describing the deliberative module. The interaction between the deliberative and the reactive modules (Fig. 2) is define in an single way through elementary tasks to achieve. A task consists in reaching a target position while avoiding obstacles.

A Probabilistic *Road-Map* [17] is defined as a graph $< W, P >$ where W is the set of way-points (nodes) and P is the set of paths (edges). The way-points set W is composed of the points of interest I in addition to the environment way-points (points around the known obstacles) (Fig. 1). The *Road-Map* is assumed to be fully connected.

$$\begin{aligned} &\text{Road-Map} = \{W, P\}, \quad \text{where:} \quad W = \{w_1, ..., w_k\} \\ &P = \{(w_1,\ w_2,\ \vec{v},\ c,\ u) \mid w_1, w_2 \in W, \quad \vec{v} \in \mathbb{R}^2, \quad c \in \mathbb{R}, \quad u \in [0,1]\} \end{aligned} \tag{1}$$

Each path p is defined by the current w_1 and the targeted w_2 way-point. A vector $\vec{v}$ gives the relative position of w_2 from w_1. The attribute c is the associated cost; it depends on the distance and the quality of the path (obstruction, slope, kind of the ground). The attribute u is the probability to reach a target position without map actualization resulting from detecting an unknown passage or obstruction (obstacles up to a given dimension).

Structuring knowledge of collision-free connectivity in a *Road-Map* allows the deliberative module to use graph algorithms like $A*$ to plan the movements of an agent or a fleet of agents [5]. Graph theory does not directly fit to the stochastic aspect to decide a robot action in uncertain environments. However, it is possible to combine the *Road-Map* with Markovian Decision Processes [21]. It was used to improve path finding by minimizing the movement cost and considering collision safe paths [3].

2.2. The markovian decision processes

An MDP is defined as a tuple $< S, A, t, r >$ with S and A respectively, the state and the action sets that define the system and its control possibilities. t is the transition function defined as $t : S \times A \times S \rightarrow [0, 1]$ that gives the probability $t(s, a, s')$ to reach the state s' from s by doing action $a \in A$. The reward function r is defined as $r : S \times A \rightarrow \mathbb{R}$, $r(s, a)$ gives the reward obtained by executing a from s.

A policy function $\pi : S \rightarrow A$ assigns an action to each system state. Optimally solving an MDP consists in searching an optimal policy π^* that maximizes the expected gain. π^* maximizes the value function of Bellman equation [6] defined on each state:

$$V^{\pi}(s) = r(s, a) + \gamma \sum_{s' \in S} t(s, a, s') V^{\pi}(s'), \quad a = \pi(s) \tag{2}$$

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} (r(s, a) + \gamma \sum_{s' \in S} t(s, a, s') V^{\pi^*}(s')) \tag{3}$$

The parameter $\gamma \in [0, 1]$ balances the importance between future and immediate rewards. In case of finite problems, as visiting a set of points, γ is set to 1.

Several studies [15,24] use MDPs in mobile robotics. From the proposed architecture, a policy is a precomputed task response to each robot state. To visit a set of points, each robot state needs to includes the current robot position in the *Road-Map* and the exploration step (the set of already visited points of interest I'). The number of states increases exponentially regarding the total number of points of interest ($|S| = |W|.2^{|I|}$) that prevent on-line solving for problems involving too mush points of interest. With regard to experiment (Section 3), we consider a threshold of 20 points of interest.

2.3. MDPs decomposition

The decomposition of an MDP permits to decrease the complexity of the policy computation by building a hierarchy between local problems and a global solution. It is particularly efficient in spatial problems as it is based on the topological aspect of transitions. The idea of Decomposed MDPs is to aggregate strongly connected states together in sub-MDPs to compute the policy in a distributed way [9,12]. Several policies are computed for each sub-MDP depending on the current policy of the neighboring sub-MDPs.

The presented approach aims to partition an MDP built for all the *Road-Map* and all the points of interest in order to solve the problem region by region. The problem of computing an optimal graph partition is known to be NP-Complete [8,16]. In case of Decomposed MDPs, the optimal partition on S is the partition that allows us to compute the policy as quickly as possible. Decomposing an MDP into sub-MDPs allows us to reduce the computation complexity of the overall MDP policy. This is true if the sub-MDPs are enough independent between them. In fact, solving the overall policy necessitates to compute at least one policy per sub-MDP. In case of dependencies, several computations per sub-MDP are needed in order to adjust the local policies together.

Generally, decompose an MDP consist in build partition of the state set as balanced as possible by minimizing connections between sub-MDPs [20,22]; that means minimizing the cuts on the transitions set. A greedy decomposition (Section 2.2) permits the robots to instantaneously build regions in the *Road-Map* in order to abstract a global and smaller MDP based on the regions set. This MDP allows the robots to evaluate their interest of exploring or not each of the regions regarding their current exploration step.

3. The deliberative module

The Deliberative Module (Fig. 2) consists of a coordination sub-module and a decision making sub-module. From a partitioned *Road-Map*, the coordination sub-module is available only in communication phases where the leader computes a new allocation of the regions sharing non-visited points of interest. The allocation aims to minimize the sum of the expected movement costs of all the robots waiting for their mission part. In execution phases, the decision making sub-module computes an exploration policy and supervises its execution.

3.1. The Road-Map partition

The aim of partitioning the *Road-Map* is to compute local policies in regions rather than an overall policy. The idea is to reduce the number of possible states from $|W|.2^{|I|}$ to $|W|.2^{|I|/k}$ with k regions and speed up the decision making. A partition $\Phi_k = \{R_1 \dots R_k\}$, defined on k regions, covers the way-points set with no intersection between two regions Eq. (4).

Limiting the sum of local policies computations times induces to control the number of points of interest $|I_{R_i}|$ inside each region R_i (minimizing $\sum \mid n_i^* - |I_{R_i}| \mid$, n_i^* the optimal number of points of interest per region. Similarly to the MDP decomposition problem [22], the partition must also maximize the ratio between the number of paths contained in each region ($|Input(\Phi_k)|$) over the number of paths which connect two regions ($|Output(\Phi_k)|$). Finally, the optimal partition Φ_k^* maximizes the criterion $C(\Phi_k)$:

$$\bigcup_{R_i=\Phi_k} R_i = W, \quad \forall R_i, R_j \in \Phi_k, \quad (R_i \neq R_j) \Rightarrow (R_i \cap R_j = \emptyset) \tag{4}$$

$$C(\Phi_k) = \alpha \frac{|Input(\Phi_k)|}{|Output(\Phi_k)|} + (1-\alpha) \left(1 + \sum_{R_i \in \Phi_k} \mid n_i^* - |I_{R_i}| \mid \right)^{-1} \tag{5}$$

$$I_{R_i} = I \cap R_i,\ Input(\Phi_k) = \{p(w_1, w_2, \vec{v}, c, u) \mid \exists R_i \in \Phi_k, \quad (w_1, w_2) \in R_i \times R_i\}$$
$$Output(\Phi_k) = \{p(w_1, w_2, \vec{v}, c, u) \mid \exists R_i, R_j \in \Phi_k,\ R_i \neq R_j,\ (w_1, w_2) \in R_i \times R_j\}$$

The $\alpha \in [0, 1]$ parameter balances the importance between the 2 parts of the criterion, minimizing the transition cuts and balancing the number of points of interest inside each region. This way, the final number of regions k could depend of the shape of obstacles more than an expected initial number of regions $k^* = |I|/n_i^*$. In indoor environments, it could be expected to have a region by room whatever the points of interest arrangement.

Many algorithms are proposed for graph partitioning [8]. A majority of the approaches are based on an iterative decomposition or a multilevel partitioning. The second method consists in aggregating strong

connected nodes from a level to another before partitioning and refine the partition by level. The need of a solution computed in a finite time during the mission induces to look at greedy heuristics. The proposed heuristic build regions incrementally by adding the way-points which maximize a local criterion for a current constructed region.

3.2. The sequential greedy heuristic

Similarly to $Input(\Phi_k)$ and $Output(\Phi_k)$, during the construction of the partition Φ_k $Input(R_i, w)$ and $Output(\Phi_k, R_i, w)$ define the sets of paths that connect a region R_i to a way-point w and that connect w to way-points not contained yet in the current Φ_k. Starting with a given way-point w_0, Algorithm 1 builds the region R_i by selecting the way-points which maximize a local criterion defined from $\frac{|Input(R_i, w)|}{|Output(\Phi_k, R_i, w)|}$. In case of no way-point w has a criterion value up to a bound b (fixed to 1 in this study), a new region begins with the closest free way-point to w_0.

Algorithm 1 Greedy Partitioning

Require: Road-Map $M = \{W, P\}$, $w_0 \in W$, $b \in \mathbb{R}^+$, $\Phi_k \leftarrow \emptyset$, $k = 0$
  while $\bigcup_{R_i \in \Phi_k} R_i \neq W$ do
    $R' \leftarrow \emptyset$
    choose $w' \in W - \bigcup_{R_i \in \Phi_k} R_i$, w' the closest way-point to w_0
    repeat
      $R'.add(w')$
      choose $w' \in W - \bigcup_{R_i \in \Phi_k} R_i$, w' maximizes $M.criterion(\Phi_k, R', w')$
    until $M.criterion(\Phi_k, R', w') > b$
    $\Phi_k.add(R'), (k \leftarrow k + 1)$
  end while
  return Φ_k

The expected size of regions is controlled by reducing or increasing the importance of $Output(\Phi_k, R_i, w)$ in the criterion. It is done with a parameter $e(R_i)$ which corresponds to the supposed number of future paths inside R_i. The criterion is defined as:

$$\begin{aligned} criterion(\Phi_k, R_i, w) &= \frac{|Input(R_i, w)|}{|Output(\Phi_k, R_i, w)| - e(R_i) + \epsilon} \\ e(R_i) &= \min\left(|Output(\Phi_k, R_i, w)|,\ \frac{n_i^* - |I_{R_i}|}{n_i^*}.\frac{|P|}{|W|}\right), \quad n_i^* \in \mathbb{N}^+ \end{aligned} \tag{6}$$

At each step, $e(R_i)$ is defined inversely proportional to the number of points of interest added to the region and it is bounded by the number of $Output(\Phi_k, R_i, w)$. This way, if the region R_i includes less points of interest than the optimal number n_i^*, the $|Output(\Phi_k, R_i, w)|$ number of paths is reduced and the region could be expanded. While point of interest is added, the influence of $e(R_i)$ is reduced and the criterion is more selective. Negative values of $e(R_i)$ force the selection to close the region by artificially increasing the number of cut paths. Furthermore, k^* denotes the expected number of regions ($k^* = |I|/n_i^*$). In fact, the greedy partitioning does not guarantee that $k = k^*$.

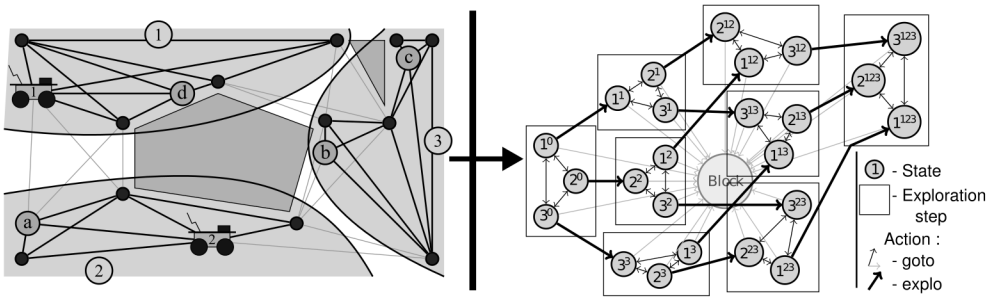

Fig. 3. A partitioned *Road-Map* in 3 regions and the attached global MDP. For example, a state 2^{13} means that the robot is in the region 2 and the regions 1 and 3 are explored.

3.3. The global MDP

After partitioning, the leader robot has to evaluate the interest of each robot in each region to visit and each robot need to orient its local policies regarding the neighbor regions. This way, the MDPs formalism is used to model a global exploration problem from the greedy region set. The global MDP $< S_g, A_g, t_g, r_g >$ (Fig. 3) returns a policy based on the abstract actions: "move from a region to another" or "explore the current region".

From a partition $\Phi_k = \{R_1 \dots R_k\}$, a *Road-Map* and a set of points of interest to allocate I, a set of regions of interest $J \subset \Phi_k$ is defined as the set of all regions with at least one point of interest. A state $s = (R_s,\ J_s)$ of the global MDP includes the region R_s where the robot is positioned and $J_s \subset J$ the set of already explored regions. A specific state called *block* is added to represent the situation where an unknown obstacle prevents the robot from reaching a way-point. Falling in the *block* state means that the robot needs to recalculate the global computed policy. This situation comes with important updates of the *Road-Map* that modifies the region configuration. In this model, when a robot is in a state (R_s, J_s), its possible actions are moving to an adjacent region R'_s or exploring the current region R_s.

$$
\begin{aligned}
S_g &= \{(R_s, J_s) \mid R_s \in \Phi_k, \quad J_s \subseteq J\} \ \cup \ \{block\} \\
A_g &= \{goto_{R_s} \mid R_s \in \Phi_k\} \ \cup \ \{explo_{R_s} \mid R_s \in J\}
\end{aligned}
\tag{7}
$$

For robot 1 in region 1 with nothing visited yet, its state is $(1, \emptyset)$ (1^0 in Fig. 3). Its possible actions are to explore region 1 (visit the point of interest (d)) or move to region 2 or 3. The exploration step J_s is increased while a new region is explored. Robot 1 could end in states $(2, \emptyset)$ or $(3, \emptyset)$ when choosing a moving action, in $(1, \{1\})$ when choosing the exploration action or in *block* if obstacles prevent the robot to reach its action.

The transition function $t(s, a, s')$ gives the probability to reach the targeted region with the expected exploration step or to get $blocked$. Transitions and rewards for moving from a state to another in the global MDP depend on the robot position in the start region and the position to reach in the targeted region. Optimally evaluating transitions and rewards induces to average all possible policies for all regions. Due to time computation constraint, the transition and the reward functions are approximated from averages on the set of paths attributes. The approximation is done proportionally to the values of uncertainty u and the cost c defined on the set of paths $P_{R_i R_j}$ from the region R_i to R_j and the number

of included points of interest $|I_{R_i}| = |R_i \cap I|$ of the explored region R_i.

$$
\begin{aligned}
& t_g(s_g,\ a_g,\ s'_g) : t_g((R_s, J_s), goto_{R_{s'}}, block) = \frac{1}{|P_{R_s R_{s'}}|} \sum_{p \in P_{R_s R_{s'}}} u_p \\
& t_g((R_s, J_s), goto_{R_{s'}}, (R'_s, J_s)) = 1 - t_g((R_s, J_s), goto_{R_{s'}}, block) \\
& t_g((R_s, J_s), explo_{R_s}, block) = t_g((R_s, J_s), goto_{R_s}, block).|I_{R_s}| \\
& t_g((R_s, J_s), explo_{R_s}, (R_{s'}, J_s \cup R_s)) = 1 - t_g((R_s, J_s), explo_{R_{s'}}, block)
\end{aligned}
\tag{8}
$$

$$
\begin{aligned}
& r_g(s_g,\ a_g) : \quad r_g((R_s, J_s), goto_{R_{s'}}) = \frac{1}{|P_{R_s R_{s'}}|} \sum_{p \in P_{R_s R_{s'}}} c_p \\
& r_g((w_s,\ I_s), explo_{R_s}) = |I_{R_s} I|.r_g((R_s, J_s), goto_{R_s}) \\
& \qquad\qquad + (|I_{R_s}|.g \text{ if } R_s \in J - J_s \mid 0 \text{ else})
\end{aligned}
\tag{9}
$$

This approximation is meaningful with homogeneous cost and uncertainty values linked to the paths set and an important density of points of interest. Using more accurate approximations induces a cost in term of computation. In the approach, the computed global policy is not directly used in the exploration. In next sections, the global policy is used to approximate an allocation and to orient the local explorations.

3.4. Regions allocation by the leader

Several times during the mission two or more robots efficiently communicate. In demand, the current leader robot re-allocate the points of interest shared by the communicating robots. A demand is set out if one of the robots have modified its global MDP by report to the last allocation. To speed up the coordination process, the leader allocate the points of interest region by region. The aim is to find the best allocation $\mathbb{J}^*_{n_r} = \{J_0, \ldots, J_{n_r}\}$ where each $J_i \in \mathbb{J}_{n_r}$ matches the set of regions allocated to the robot Ag_i. The optimal allocation maximizes the sum of the n_r robots expected gains in the $n_r^{|J|}$ possible allocation.

Using the *Value Iteration* algorithm [21] on the global MDP allows a robot to compute the optimal abstract policy as moving between regions and exploring them. From this policy, the value function of bellman $V^{\pi^*}(s)$ Eq. (2) returns the expected accumulation of future gains from the robot state s. Knowing the set R_{Ag_i} of the current regions of the robot Ag_i, the computed global policy $\pi^*_{Ag_i}$ and the value function of bellman $V^{\pi}(s)$, The current leader robot search:

$$
\underset{\mathbb{J}^*_{n_r} \in \{\mathbb{J}^0_{n_r},\ \ldots,\ \mathbb{J}^{n_r^{|J|}}_{n_r}\}}{\operatorname{argmax}} \left(\sum_{i=0}^{n_r} V^{\pi^*_{Ag_i}}(R_{Ag_i},\ J - J_i) \right) \tag{10}
$$

By considering up to 3 robots and 12 regions ($n_r \leqslant 3$ and $k^* \leqslant 12$) it is possible to test all the set of n_r-allocations. The leader, for each possible allocation $\mathbb{J}^j_{n_r}$, computes the sum of expected gain for each robot Ag_i ($V^{\pi^*_{Ag_i}}(R_{Ag_i},\ J^j_i)$) and holds $\mathbb{J}^*_{n_r}$ with the maximum cumulated expected gains. The using of a *Road-Map* with multi-level MDPs permits the leader to consider more than 3 robots for on-line computations. In fact, the approach needs a single Global MDP by class of homogeneous robots (robots sharing a same *Road-Map*) and the policies computation necessitates n_r *Value Iteration* in the worst case. The calculation of $\mathbb{J}^*$ by an exhaustive way limits the considered number of robots to only few robots.

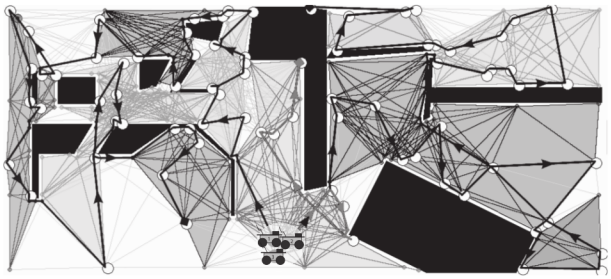

Fig. 4. The 3 likely trajectories following the computed policy ($n_r = 3$, $|I| = 80$ and $n_i^* = 10$).

3.5. Exploration strategy in region

After a communication phase or after all map modifications each robot has to compute a local policy concerning its movements in the region where the robot is located. The global policy is used by robots to orient the local policy computation in region. It is expected that a region exploration ends close to the next region to explore (Fig. 4).

Inside the region, the robot Ag_i builds a local MDP $< S_R, A_R, t_R, r_R >$ in a similar way to the global MDP by considering only the points of interest inside the region $I_R = R \cap I_{Ag_i}$. The actions are to move between way-points and the transition and reward functions are given by the paths set included in the considered region R.

$$S_R = \{(w_s, I_s) \mid w_s \in W,\ I_s \subseteq I_{Ag_i}\} \cup \{block\},\ A_R = \{goto_{w_s} \mid w_s \in W\} \tag{11}$$

$$\begin{aligned} &t_R((w_s, I_s), goto_{w_{s'}}, block) = u_p \text{ if } \exists p \in P,\ w_p = w_s \in R, w'_p = w_{s'} \mid 1 \text{ else} \\ &t_R((w_s, I_s), goto_{w_{s'}}, (w_{s'}, I_s \cup (w'_s \cap I_s))) = 1 - t_R((w_s, I_s), goto_{w_{s'}}, block) \end{aligned} \tag{12}$$

$$r_R((w_s, I_s), goto_{w_{s'}}) = c_p + ((1 - u_p).g \text{ if } w_{s'} \in I_R - I_s \mid 0 \text{ else}) \tag{13}$$

Specific rewards are added when the region is totally explored ($I_R \subset I_s$). All the external way-points connected to the current region R define the possible outdoors. The value of taking an outdoor w is given by the expected gain of the neighbor region R' in the global MDP ($V_G^{\pi^*}(R', J_s)$, J_s the current exploration step on the region set including R). The region R' is the region of the Partition Φ_k where $w_{s'}$ is located:

$$r_R((w_s, I_s), goto_{w_{s'}}) = c_p + (1 - u_p) V_G^{\pi^*}(R', J_s) \text{ if } I_R \in I_s, w_{s'} \in R' \neq R \tag{14}$$

4. Experimental results

A visual representation of the built partitions (Fig. 5) shows that when the environment is more structured with a coherent expected region size Fig. 5(c), the algorithm builds a partition closer to the expected one. Otherwise, intersected regions are observed in free space environments Fig. 5(a). It is due to the non-consideration of the path cost in the used criteria Eq. (6). This phenomenon is reduced in cluttered environment Fig. 5(b).

By separating between decision and control in the robot architecture, the goal is to decrease the decision process frequency in order to optimize the policies calculations. The proposed solution decomposes

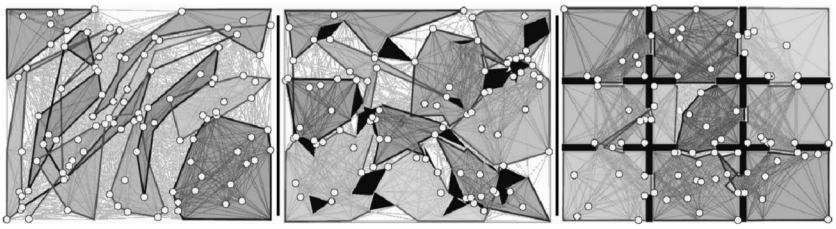

Fig. 5. Examples of partitions built from differently structured environments (Road-Maps).

the problem into regions to approximate a global policy. This process does not guarantee the optimality of the solution. In a first series of experiments, a statistic score is measured regarding the optimal solution for small problems (12 points of interest and 3 homogeneous robots).

If the control architecture allows the robot to save computing resources for decision processes, these resources still have to be controlled. A second series of experiments evaluates the capability to bound computing times while considering maps with tens of points of interest. The approach is experimented with a standard computer (Intel Core2 Quad CPU Q9650 at 3.00 GHz).

4.1. Qualitative evaluation

The first series of experiments compares the value of the allocation built after a greedy partitioning of the *Road-Map* with the values of the best and the worst allocations computed without partitioning. The value of an allocation is calculated by summing the Bellman value of the individual optimal policy considering each attribution I_{Ag_i} regarding the current robot position w_{Ag_i}:

$$value(I_{Ag_1},\ I_{Ag_2} \ldots I_{Ag_{n_r}}) = \sum_{i=1}^{n_r} V^{\pi^*}(w_{Ag_i},\ I - I_{Ag_i}) \tag{15}$$

The Bellman value $V^{\pi^*}(s)$ depends on the path cost and the gain for visiting a point of interest. The path cost is computed directly from the distance between the 2 way-points with a maximum cost of 51.2 for the diagonal limit of the environment. The reward gain for visiting a point of interest is set to 1000 and the greedy partitioning are defined to include 3 points of interest by region ($c_p \in [0, -51.2]$, $g = 1000$ and $n_i^* = 3$).

The experiments are done for the map represented in Fig. 4 and they are classified regarding the considered number of points of interest $|I|$. This number is bounded to 12 because of the complexity of computing the best and the worst allocation using an exhaustive way. For each experiment, a normalized score is calculated for the computed allocation and a random allocation regarding the possible best and worst allocations. For example, an experiment in the class $|I| = 11$ gave values of 10913.8 (best); 10822.9 (worst); 10872.9 (random); 10884.5 (after partitioning) with a linked normalized score of 0.677. The first series of experiments is composed of 200 random generations of points of interest positions for each class of experiments in order to statistically evaluate the allocations (Fig. 6).

The obtained scores (Fig. 6) show the difficulty of finding the optimal allocation. The average score is around 77% between 6 and 12 points of interest that induce between 2 and 4 regions. It is an interesting average considering that the computation is instantaneous. Furthermore, the averages are affected badly by a few numbers of bad experiments induced by unsuitable partitions; The score and the number of these experiments are inversely proportional with the number of points of interest which induces partitions with more regions.

Table 1
A presentation of average measures to characterize the built solutions: the average number of regions, the average bounds of regions sizes, the needed time to compute them and the most common allocation. This table represents a relevant part of experiments regarding a selection of class of problems ($|I|$)

$\|I\|$	average number of regions (k)	average bound size of regions	time (s)	commonest allocation
5	1.13	4.38–4.97	0.059	1–0–0
20	3.41	3.43–8.13	0.047	2–1–0
40	5.6	4.12–10.14	0.053	3–2–1
60	7.71	4.29–11.44	0.116	4–2–1
80	9.43	4.73–12.74	0.339	5–3–1
100	10.63	4.36–14.19	0.897	6–3–1
120	12.42	3.74–14.81	4.95	6–4–2

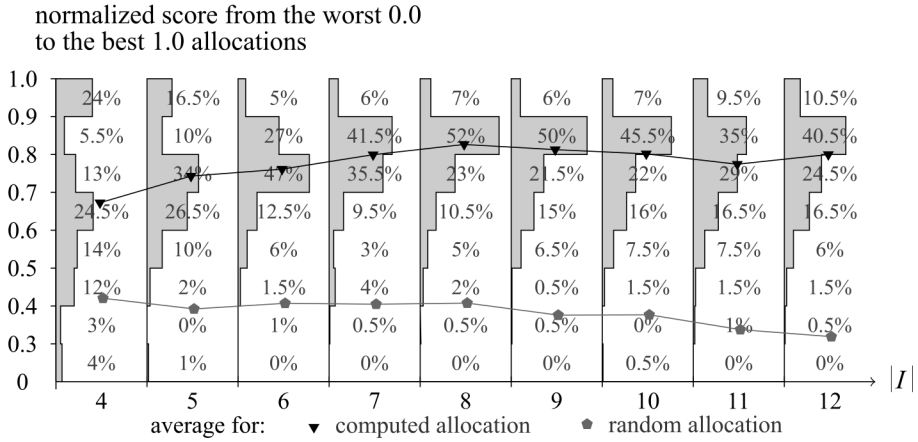

Fig. 6. Population of obtained normalized scores by the allocation after partitioning. The population is presented for each considered number of points of interest $|I|$.

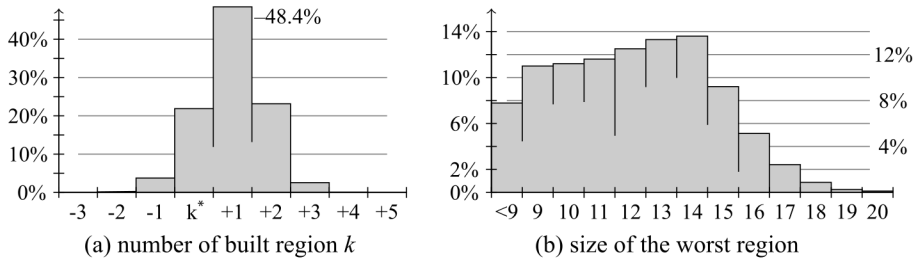

Fig. 7. Classification of the experiments regarding the difference between the built k and the expected k^* number of regions and the size of the worst region in term of number of contained points of interest.

4.2. Computation duration control

The approach is designed to consider on-line a large number of points of interest. The second series of experiments evaluates the capability to control the needed time to compute the global policy and to allocate the regions of interest. The considered problem involved up to 120 points of interest, 3 robots and a desired number of 10 points of interest by region ($|I| \leqslant 120$, $n_r = 3$ and $n_i^* = 10$). Our experiments generate points of interest randomly (Fig. 4) (500 random generations for different numbers of points of interest $|I|$ growing by 5).

Table 1 and Fig. 7 present some experiment results that validate the control of the expected number of regions, the numbers of points of interest in a region and the associated computation time. MDP sizes and computation time grow exponentially with the number of regions k or, with the number of points of interest in a region (in local MDPs). We notice that the allocations are unbalanced, it is normal in regard to the placement and shapes of obstacles. Indeed, the intention was to minimize the sum of robots movement cost, the criteria Eq. (6) does not does not especially balancing the allocation.

Experiments with few points of interest in regard to the number and the structure of obstacles lead to a partition with more regions than expected ($k > k^*$) and worst regions which have a lower size than desired size d (Fig. 7). This results denotes the capacity of using the shapes of existing obstacles to built different regions.

5. Related work

The approach proposed in this paper focuses on task allocation for multi-robot exploration problem based on a topological representation of the environment. In the literature, the majority of deliberative robot's architectures are based on a grid map representations [10,11,15,19,23] which permit a discretization of the environment according to the space and time. Using grid map permits efficient simultaneous localization and mapping protocols (SLAM) [14,25] but induces a high frequency of map modifications. Time rate limits the computation resources and heuristics are used to decide the exploration strategy. Generally robot reaches the closer frontier position of exploration [14,25].

In the same way, deliberative methods for multi-robot exploration are focused on high frequency algorithms. Market-based multi-robot coordination [13] assigns task to the robots with the highest bids. Generally the bids are calculated regarding the current robots location. In [23], the bid for a client position is computed regarding its distance from the robot and the frontier size that will be push away. In [10], the bids are actualized between two allocations in order to take into account all the zone in a given radius (sensor range) that will be explored by robots with an assigned task. In [19], task assignment is made in a distributed way via a set of individual MDPs. Each robot incorporates knowledge about the other robots positions in its distributed value function.

The proposed solutions provide a unique assignment of tasks at a time depending on the current robot position, they are dependent to communication or to the possibility of observing other robots. In case of a break in communication, the robots can store latest assignments and/or other robots positions [10] that are available only for a short duration. These solutions are efficient when robots evolute together, we aims to give more autonomy in mission execution part in order to automatically dispatch robots in the exploration area.

Using a topological representation of the environment during the topological SLAM [18] allows us to reduce the rate of the map modifications and to reduce the input data size of the deliberative process. More resources can be allocated to compute robots joint policies for a long term horizon. Multi-agent planning in stochastic domains could be defined by the Decentralized MDP formalism [7] (Dec-MDP). The Dec-MDP applied to mobile robotics could not be optimally solved for real size problems. Splitting the Dec-MDP into a set of MDP by the agents permits to compute sub-optimal solutions. This approach was used on the Pursuit Problem [11] by iterating on the built policies until a consensus is found between all agents. This solution is used before the mission execution, the joint policies of the collaborative agents can not be modified.

6. Conclusion and future work

This paper presents a decision making architecture for a fleet of mobile robots sharing an exploration mission. The aim is to improve the exploration strategy by considering all the tasks to achieve in the communication and the execution phases. It is difficult for a leader to evaluate the allocations of many tasks between robots. The study is based on knowledge organized in a *Road-Map* and a solver based on MDPs to model individual exploration of the environment.

Decomposing the input data is an appropriate idea to deal with the size problem and the constraint of on-line computation. From a greedy decomposition of the map, the robots are able to allocate the regions to visit and compute coherent local policies with regard to their global exploration. The approach is based on several heuristics as the partitioning, the abstract MDP generation or the allocation criterion. They may be correctly parametrized to produce expected results. With coherent parameters, the experiments show that the built allocations is statistically close to the optimal one for a given criterion. It was also demonstrated that the approach permit the robots to deal with tens of tasks during the mission, by controlling the number of tasks in each region and the number of regions.

In future work, the partitioning quality will be improved by using algorithms based on finding minimal cuts. The idea is to converge to a local optimum from the first greedy decomposition and assume other characteristics as disjunction between regions. Furthermore, adding cooperation in the local MDPs solving will permit several robots to explore a region together and increase the allocations quality.

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab and F. Ingrand, An architecture for autonomy, *Journal of Robotics Research* **17** (1998), 315–337.

[2] L. Adouane, Hybrid and safe control architecture for mobile robot navigation, *9th Conference on Autonomous Robot Systems and Competitions*, 2009.

[3] R. Alterovitz, T. Siméon and K. Goldberg, The stochastic motion Road-Map: A sampling framework for planning with Markov motion uncertainty, *Robotics: Science and Systems*, 2007.

[4] B. Bakker, Z. Zivkovic and B. Kröse, Hierarchical dynamic programming for robot path planning, *International Conference on Intelligent Robots and Systems* (2005), 2756–2761.

[5] O.B. Bayazit, J.M. Lien and N.M. Amato, Swarming behavior using probabilistic Road-Map techniques, *Swarm robotics: SAB international workshop*, 2004.

[6] R. Bellman, A markovian decision process, *Journal of Mathematics and Mechanics* **6** (1957), 679–684.

[7] D.S. Bernstein, S. Zilberstein and N. Immerman, The complexity of decentralized control of markov decision processes, *16th Conference on Uncertainty in Artificial Intelligence*, 2000.

[8] C.E. Bichot and P. Siarry, Graph partitioning, *Wiley-ISTE*, 2011.

[9] C. Boutilier, T. Dean and S. Hanks, Decision-theoretic planning: Structural assumptions and computational leverage, *Journal of Artificial Intelligence Research* **11** (1999), 1–94.

[10] W. Burgard, M. Moors, C. Stachniss and F. Schneider, Coordinated Multi-Robot exploration, *IEEE Transactions on Robotics* **21** (2005), 376–386.

[11] I. Chades, B. Scherrer and F. Charpillet, A heuristic approach for solving Decentralized-POMDP: Assessment on the Pursuit Problem, *ACM Symposium on Applied computing* (2002), 57–62.

[12] T. Dean and S.H. Lin, Decomposition techniques for planning in stochastic domains, *14th International Joint Conference on Artificial Intelligence*, 1995.

[13] M.B. Dias, R. Zlot, N. Kalra and A. Stentz, Market-Based multirobot coordination: A survey and analysis, *Proceedings of the IEEE* **94** (2006), 1257–1270.

[14] A. Elfe, Sonar-based real-world mapping and navigation, *Robotics and Automation* **3** (1987), 249–265.

[15] A.F. Foka and P.E. Trahanias, Real-time hierarchical POMDPs for autonomous robot navigation, *Robotics and Autonomous Systems* **55** (2007), 561–571.

[16] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified NP-complete problems, *6h Symposium on Theory of Computing* (1974), 47–63.

[17] L. Kavraki, P. Svestka, J.C. Latombe and M. Overmars, Probabilistic Road-Maps for path planning in high-dimensional configuration spaces, *IEEE International Conference on Robotics and Automation* (1996), 566–580.
[18] B. Kuipers and Y.-T. Byun, A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, *Journal of Robotics and Autonomous Systems* **8** (1991), 47–63.
[19] L. Matignon, L. Jeanpierre and A.I. Mouaddib, Distributed value functions for Multi-Robot exploration, *6th Workshop on Multiagent Sequential Decision-Making in Uncertain Domains*, 2011.
[20] R. Parr, Flexible decomposition algorithms for weakly coupled markov decision problems, *14th Conference on Uncertainty in Artificial Intelligence* (1998), 422–430.
[21] M.L. Puterman, Markov decision processes: Discrete stochastic dynamic programming, *John Wiley & Sons Inc*, 1994.
[22] R. Sabbadin, Graph partitioning techniques for markov decision processes decomposition, *15th Eureopean Conference on Artificial Intelligence* (2002), 670–674.
[23] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun and H. Younes: Coordination for Multi-Robot exploration and mapping, *National Conference on Artificial Intelligence*, 2000.
[24] F. Teichteil-Königsbuch and P. Fabiani, Autonomous search and rescue rotorcraft mission stochastic planning with generic DBNs, *IFIP International Conference on Artificial Intelligence* (2006), 483–492.
[25] S. Thrun, Robotic mapping: A survey, Exploring Artificial Intelligence in the New Millenium, Chapter I, *Morgan Kaufmann*, 2002.
[26] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras and H. Das, The CLARAty architecture for robotic autonomy *Aerospace Conference* **1** (2001), 121–132.