Punctual versus continuous auction coordination for multi-robot and multi-task topological navigation

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

Draft manuscript No.
(will be inserted by the editor)

Punctual Versus Continuous Auction Coordination for Multi-Robot and Multi-Task Topological Navigation

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

Received: date / Accepted: date

Abstract This paper addresses the interest of using Punctual versus Continuous coordination for mobile multi-robot systems where robots use auction sales to allocate tasks between them and to compute their policies in a distributed way. In Continuous coordination, one task at a time is assigned and performed per robot. In Punctual coordination, all the tasks are distributed in Rendezvous phases during the mission execution. However, tasks allocation problem grows exponentially with the number of tasks. The proposed approach consists in two aspects: (1) a control architecture based on topological representation of the environment which reduces the planning complexity and (2) a protocol based on Sequential Simultaneous Auctions (*SSA*) to coordinate Robots' policies. The policies are individually computed using Markov Decision Processes oriented by several goal-task positions to reach. Experimental results on both real robots and simulation describe an evaluation of the proposed robot architecture coupled wih the *SSA* protocol. The efficiency of missions' execution is empirically evaluated regarding continuous planning.

Keywords Multi-Robot · Decision Making · Auction Coordination

Guillaume Lozenguez
Dept. Informatique et Automatique, Mines Douai
764 boulevard Lahure, 59500 Douai, France
E-mail: guillaume.lozenguez@mines-douai.fr

Lounis Adouane
Pascal Institute, University Blaise Pascal
24 Avenue des Landais, 63177 Aubiere, France
E-mail: lounis.adouane@univ-bpclermont.fr

Aurélie Beynier
LIP6, University Pierre & Marie Curie,
4 place Jussieu, 75005 Paris, France
E-mail: aurelie.beynier@lip6.fr

Abdel-Illah Mouaddib
GREYC, University of Caen Basse-Normandie
Bd du Marechal Juin, 14000 Caen, France
E-mail: abdel-illah.mouaddib@unicaen.fr

Philippe Martinet
IRCCYN, École Centrale de Nantes
1 rue de la Noë, 44312 Nantes, France
E-mail: philippe.martinet@irccyn.ec-nantes.fr

1 Introduction

Increasing the number of robots in a mission permits to improve the efficiency and to reduce the time needed to complete the mission. The paper focuses on multi-task missions where each task is located in the environment and can be achieved by a single robot. Using a team of robots, tasks can be simultaneously executed. However, multi-robot missions require good coordination between the robots' actions to optimize execution performances.

The required decision making consists in computing the shortest and safest paths allowing the robots to perform all the tasks while minimizing the resources consumed by the robots and the overall mission duration. Optimally mapping individual actions to each possible succession of local perceptions is very hard to compute [1]. In standard distributed approaches, each robot uses its own computing resources to decide its owns movements [2] thus parallelizing the policy computations. Two kinds of coordination approaches can be used to allow robot teams to coordinate in a distributed way: Continuous or Punctual Coordination.

Continuous Coordination consists in interleaving coordinated decision making and action execution at each step of the mission execution. One task is attributed at a time to each robot. When a robot terminates its task, it choses another one in the set of unassigned tasks [3,

4,5,6]. Generally, decision making is based on the distance to the remaining tasks in order to select the closest task. The decision process can also be impacted by the other robot locations to improve task allocation. Continuous Coordination requires efficient communication and work-around computing multi-task paths. However, those solutions handle very well dynamics in tasks and in knowledge as in exploration scenarios.

Punctual Coordination aims to reduce the frequency of coordination requirements [7,8]. During some Rendezvous points, robots share their information and distribute several tasks between them. Robots are then autonomous to perform allocated tasks until the next Rendezvous [8]. However, such approach requires important computational resources to allow the group to coordinate path planning involving several tasks.

This paper question the reduction of the robot's movements and mission duration while using Punctual rather than Continuous coordination. The challenge consists in computing both individual multi-task policies and efficient task allocation with respect to realistic robot control. The complexity of solving punctual coordination is reduced by representing the environment using Topological Map [9,10] and by solving coordination with an appropriate auction protocol [11]. However, the auction protocol has to allocate inter-dependent tasks [12,7].

The Topological Map represents the environment as a graph of paths connecting key positions (or waypoints). The Topological Map is the main element allowing the robot architecture to link reactive control [13] and multi-task planning. This way, by considering only few key positions in the environment, the robots can punctually coordinate their movements to visit a set of task-positions (at least one robot per task). The proposed coordination is based on Markov Decision Processes (*MDPs*) oriented by several goal-tasks to individually plan movements and a specific auction-based protocol to assign tasks between robots.

The remainder of the paper is organized as follows: Research background of the proposed approach is described in Section 2. The robot architecture is detailed in Section 3. Distributed decision making using Continuous coordination is presented in Section 4. The Sequential Simultaneous Auctions coordination protocol is detailed in Section 5. Section 6 describes experimental results before ending with a discussion and a conclusion in Sections 7 and 8 respectively.

2 Background

In open environments with uncertain obstacle shapes and movement achievements, robots have to be autonomous in their movement control and supervision. Hierarchically, the robot autonomy results in its capability of: modeling the robot environment, computing individual policies of actions and negotiating task assignments. This section introduces the background of the proposed approach. Starting with generic robot planning methods discussed regarding multi-task missions, methods dealing with uncertain and multi-robot context are reviewed.

2.1 Map and Path Planning

Two main kinds of maps have been proposed in the literature to localize robots in their environment: the occupancy Grid Maps and the Topological Maps. Occupancy Grid Maps consist in a metric sampling of the space in elementary cells [10]. Each cell could be qualified as *free*, *obstacle* or *unknown*. The occupancy Grid Maps are the most common environment representation used for mobile robots. Robot movements are decomposed by elementary movements (cell-by-cell), then, Potential Fields, Rapidly-exploring Random Trees and Markov Decision Processes represent the most popular path planning approaches used for mobile robots.

In Potential Field approaches, robots are artificially attracted by a positive force oriented to the target while repulsive forces allow the robot to keep a safe distance from obstacles [14]. Applying several sources of attractive forces allows the robot to orient its movements towards one of them (the closest or the most attractive). This approach is myopic since the robot chooses the current most interesting task without anticipating future choices. Furthermore, potential field has to be propagated in each cell of the map.

Rapidly-exploring Random Tree (RRT) algorithm consists in building iteratively a set of reachable positions by applying random elementary movements [15]. The algorithm terminates on a non-optimal solution when a branch of the tree finishes at the target position. Using 2 RRTs, one starting from the robot position and another from the target position allows to speed up the path computation [15]. As far as we know, RRT is not used to handle multi-target missions.

Markov Decision Process (MDP) is a decision making formalism handling uncertainty. MDPs generalize paths planning on graphs with stochastic edge transitions and they have been successfully used in mobile robotics [5,16,17,6,18]. MDP formalism is generic, however, using this formalism for multi-target problems based on Grid Map will produce intractable state space.

For instance, a 200 m^2 apartment mapping at 1 cm precision will result in a minimum of 2×10^6 cells and

$4 \times 2 \times 10^6$ possible elementary movements considering only 4 movement directions while tens of key-positions could describe the apartment. Reducing the map definition will permit to speed-up path planing in order to handle multi-task positions in punctual coordination phases. A Topological Map is a graph representation where nodes match particular locations and edges represent the path connectivity [9]. Nodes match semantic descriptions (intersection, corner etc.) [9,19] or perceptive snapshots [20]. This way, the Topological Map is defined as succession of local configurations linked by local possible displacements. Metric knowledge in a global system (as using Voronoï diagram from a Grid Map [21]) can be interesting but is optional.

Using Topological Map, the proposed approach aims to map the environment with a minimum number of key-positions while guarantying that key-positions are reachable using reactive control. This way, robots would punctually handle multi-target path planning using MDP framework.

2.2 MDP Framework

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

Solving an MDP consists in searching an optimal policy π^* that maximizes the expected gain. A policy is a function $\pi : S \to A$ mapping each state to an action. The value V^π of the expected gain regarding a policy π can be computed by solving the Bellman equation [22]. This value depends on a parameter $\gamma \in [0, 1]$ which balances the importance between future and immediate rewards:

$$V^\pi(s) = r(s, a) + \gamma \sum_{s' \in S} t(s, a, s') V^\pi(s') \quad \text{where} \quad a = \pi(s) \tag{1}$$

Commonly, the states match the possible configurations of the robot in its environment. For a robot in a static known environment and with a perfect estimation of its position, MDP states match all possible positions in the environment [5,6]. Because of imperfect perception skills, the robot may not be able to observe its state. Partially Observable MDPs (POMDPs) extend MDPs to partially observable settings where a robot makes decisions from its history of actions and partial observations about its state [17]. The history of information can be summarized as a belief state which consist of a probability distribution on possible state configurations. Unfortunately, partial observability increases the computation complexity of computing an optimal policy.

2.3 Distributed Policy Computation

In multi-agent systems under distributed control, a policy is assigned to each agent. The problem consists in computing the optimal decentralized joint policy. A joint policy is a set of individual policies, one for each agent. In cooperative settings, an optimal policy is a policy that maximizes the expected gains of the system [1]. Computing such an optimal policy is NEXP-hard.

Most existing approaches compute the joint decentralized policy in a centralized way and needs important computational resources. Expressive and complete multi-agent models are difficult to solve since they lead to huge state or belief state spaces. It is notably true in case of multi-task problems (as traveling salesmen problems) where states have to represent which tasks are achieved or not [18].

However, policy computation could be distributed as well. In generic distributed approaches, each agent computes its own policy while considering that the policies of other agents are fixed [23,24]. In such cases, any policy actualization of an agent induces modification in the individual transition and reward functions of the other agents. Thus, each agent iteratively updates its policy until an equilibrium is reached.

Ad-hoc distributed approaches are widely used to solve specific problems. The paper focuses on coordination problems considering that dependencies between robot policies only result from tasks allocation.

2.4 Coordination by Tasks Allocation

In consensus approaches [2], auction-based coordination consists in attributing tasks or resources as items among agents. These approaches were successfully used in robotics [11]. In Contract Net [3] or MURDOCH [4], an item (object, resource or task) is put up for sale by a robot who becomes manager. The other robots are potential clients for the item. Clients send bids for the item to the manager and the item is allocated to the client with the highest bid.

Bids and attribution rules can be defined differently to optimize the sum of individual interests or to lead to balanced allocations [25]. Individual MDPs and Bellman equation can be used to compute individual gains regarding a set of tasks or regarding the addition or

the subtraction of one task (or several) to the set of already assigned tasks. This mechanism has been used to evaluate bids in robot auctions for coordination [5]. The value of each task depends on the tasks already assigned to the agent. This generally requires to put back in sale, several times, tasks with strong dependencies.

Combinatorial auctions [12], where agents can bid on a set of items, are able to express synergies between items' values. This kind of auctions reaches optimal tasks allocation in a unique simultaneous sale [7] by biding on combinations of items. However, agents have to detect and evaluate synergy between tasks. This induces several policy computations, one per possible task allocation. Thus, the number of policy computations is exponential with the number of task combinations.

In punctual coordination during the mission execution, robots have to efficiently and quickly coordinate their actions. The main issue is both to perform fast policy computations and to allocate several interdependent items while minimizing policy actualization. In the proposed approach, a control architecture based on reactive controllers permits to reduce policy computation complexity without defining belief states. Robots build and solve MDPs based on a low density Topological Map and oriented by several tasks. Finally, a Sequential Simultaneous Auctions protocol is defined and used to distribute tasks between robots with a few expected number of iterations.

3 Robot Architecture

Previous section shows that controlling a group of robots for multi-task missions is a difficult problem. The proposed robot control architecture (Fig. 1) is based on hierarchical separation between decision and control in two levels: Functional and Deliberative [26,27]. The originality of the proposed architecture mainly lies in the definition of the Topological Map as the key element connecting perception, decision making and control. The architecture aims to reduce the complexity of distributed policy computation by considering low density graph model under realistic hypothesis.

The Functional level provides an interface to perform local tasks such as catching an object, following a trajectory, describing the environment, etc. The Deliberative level plans the sequence of local tasks in order to reach the goals. The proposed Functional level is split into Perception and Control parts and the Deliberative level is split into Representation and Supervision parts. Each part of this control architecture is detailed in the remaining of the section.

3.1 Perception Part

The perception part refines the sensors input stream to produce usable information for the localization and robot control. It is composed of several refiner modules working in parallel: *Local Interpreter*, *Odometer* and *Landmark Detector*.

The *Local Interpreter* allows to separate navigable space free of obstacles in the perceived area and to add a semantic perceptive state to the local configuration. The obstacles shapes around the robot are defined as a list of polygons (Fig. 2b). In fact, obstacle detection is only efficient in a limited range around the robot because of obstacle obstruction and image or laser refinement (number of pixels or number of beams). In a limited range around the robot, shapes could be defined with enough accuracy.

Therefore, from obstacle shapes, the *Local Interpreter* associates a semantic to the local configuration. The local configuration is defined by the number of obstacles and more importantly the numbers of exits to move out of the local area. An exit is detected if the distance between obstacles allows the robot to move through or if there are no other obstacle, in the local perception, which prevent the robot to move on (e.g. Fig. 2b has 3 exits). Exits can be delimited by one or two obstacles (open or regular exits). Considering local configuration of open or regular exits, we dissociate eight semantic perceptive states from the free space (no obstacle) to the intersection (4 exits or more) (Fig. 2 c).

In parallel to the *Local Interpreter*, we consider that the robot is equipped with an *Odometer* giving an estimation of the robot displacement and a *Landmark Detector* allowing the robot to recognize a place crossed several times. We consider that these two refiners per-

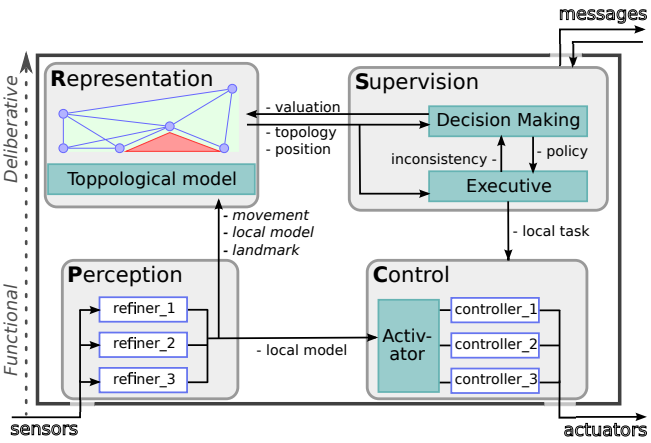


Fig. 1 The topology-based hierarchical architecture.

mit the robot to maintain an accurate enough localization in the environment and more specifically to mark funnel and intersection area with unique identifier whatever the used entrance in the area. However, this paper does not focus on localization problems. Semantic place recognition coupled with metric information achieve good performances in indoor environments [10]. This hypothesis is valid also in urban environments using visual sensors [20,19].

3.2 Control Part

The control part produces an immediate response to a local situation given by the refined perception (Local model and movement estimation) in order to permit reactive robot to navigation in presence of obstacles [13]. It is composed of several controller modules which can be parametrized and activated on demand. The robot control is defined by one metric *Reaching Controller* and one *Topological Controllers* parametrized by a local task.

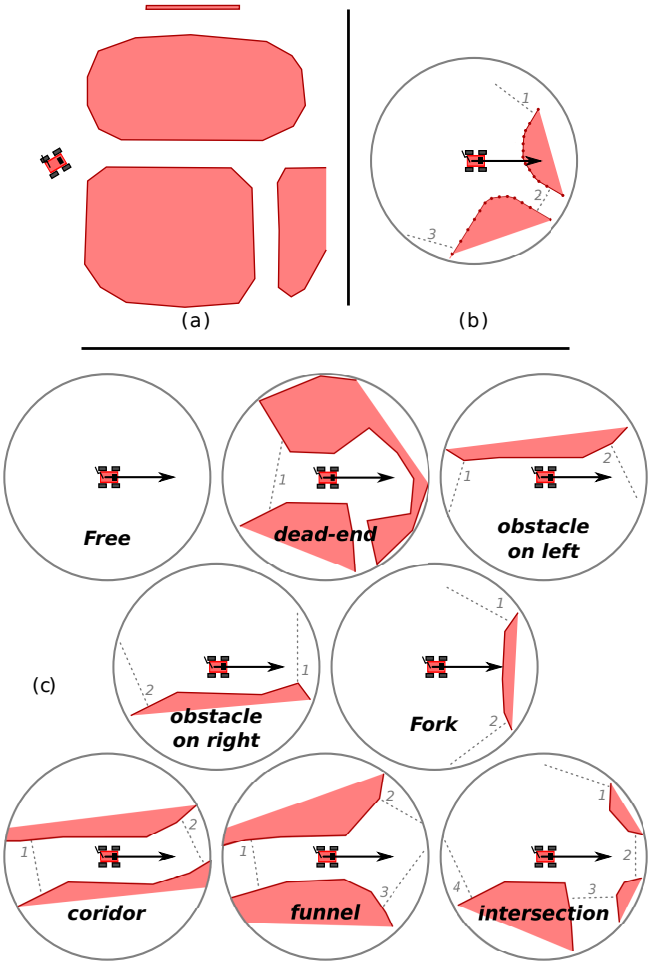


Fig. 2 From local perception to semantic characterization: (a) the robot in the environment ; (b) the corresponding local model, this configuration has 3 exits (in gray dotted lines) ; (c) the possible 8 local semantic states.

Reaching Controller is defined as the default controller of the robot to reach a distant target. If an obstacle of the local model defined by *Local Interpreter* refiner prevents the robot to reach the target, two candidate local targets are built, one on the left and another on the right of the obstacle. The target inducing the minimal deviation is chosen. This way, the robot smoothly moves to the target while avoiding convex obstacles. The local task is defined as a target distant position with respect to the current robot position. The target position is actualized according to movement estimation that prevents using the *Reaching Controller* for long movements.

The *Topological Controller* defines control rules directly from the local model rather than from metric targets. The *Topological Controller* works in a similar way to the *Reaching Controller* but by targeting the middle of an exit in the local navigable space (Fig. 2b, c). The *Topological Controller* is independent to the movement estimation while the exit to target is topologically defined. Therefore, local task associates an exit to target to the current semantic perception state by considering that local exits are indexed from the robot's left to its right (Fig. 2b, c). The *Topological Controller* permits the robot to navigate in corridor, to circle an obstacle and to choose a direction in an intersection.

The controllers could be activated or not depending on the semantic of the local configuration and to the current local task. The current local task is given by the Deliberative level according to the next distant task position to reach.

3.3 Representation Part

The Representation part manages the knowledge about the environment to allow localization, decision making and supervision of task achievement. This part is mainly composed of a Topological Map directly built from the Functional level capacities: refiners and controllers (Fig. 1).

The Topological Map $\langle W,\ P\rangle$ (Fig. 3) is a graph where: the nodes represent particular waypoints; the edges represent the paths connectivity between the nodes. A waypoint $w \in W$ in the map identifies a key position in the environment where the local semantic perception state sp_w switch between punctual and stable semantic perception states. Punctual semantic perception state is a state that is available in a limited

range and (most of the time) that requires the robot to take a decision in the direction to move on: *dead-end*, *funnel* and *intersection*. In contrary, stable semantic perception state identifies corridor and one obstacle configurations. A waypoint $w \in W$ is also characterized by its position and orientation $(x_w,\ y_w,\ \theta_w)$ and the landmark identifier id_w.

A path $p \in P$ is characterized by the initial w_p and target w'_p nodes and the control supervision rule $sr_p \in SR$ used to move from w_p to w'_p (*Reaching* or *Topological Controllers* with local task parameters). Using this configuration, the Topological Map induces topological control in stable semantic perception states and reaching control in punctual states.

Several attributes are associated to the knowledge about the path p in order to permit efficient control and decision making. An attribute c_p gives the related movement cost and matches the average path duration. In fact, the movement cost between two positions varies according to the type of ground, the slope or the obstruction of the path. According to the possible error during a movement, the function d_p (deviation) gives the probability to reach other waypoints by moving from w_p to w'_p. We assume that $d_p(w'_p)$ returns the highest probability.

The Topological Map $\langle W, P\rangle$ is defined as:

$$
\begin{aligned}
&W = \{(x_w, y_w, \theta_w, sp_w, id_w) \in \mathbb{R}^3 \times SP \times \mathbb{N}\},\\
&P = \{(w_p, w'_p, sr_p, c_p, d_p)\\
&\quad w_p,\ w'_p \in W,\quad sr_p \in SR,\\
&\quad c_p \in \mathbb{R},\quad d_p : W \to [0,1]\ \}
\end{aligned}
$$

Structuring knowledge in a stochastic collision-free connectivity permits to plan a safe and efficient path between two positions [28]. Given a global topology, the deliberation module has to compute a policy of actions to reach several goals. However, the topological mapping has to maintain a consistent knowledge about the node localization and the normalized deviation functions. Evaluating or learning the deviation function is out of the scope of this paper.

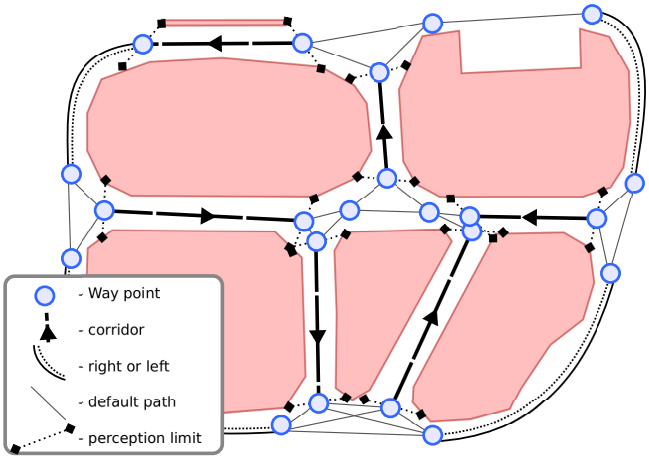


Fig. 3 Example of a Topological Map.

3.4 Supervision Part

The Supervision part connects the topological model, the goals and the control. It aims to choose the control tasks (local task) to perform regarding a global policy. Local tasks are defined in the topology according to punctual or stable semantic perception states and match the *Reaching Controller* with a target position or the *Topological Controller* with a local exit to target. Supervision part is split into a decision making module and an executive module (Fig. 1). The executive module supervises the local task to activate regarding the policy computed by the decision-making module.

The proposed approach uses MDP formalism coupled with reactive control to ensure long-term planning with smooth robot movements under uncertainty. In multi-robot missions, the policy has to be computed cooperatively by integrating communication protocols. The next two sections describe the main contribution of our approach by focusing on Punctual Task allocation and Sequential Simultaneous Auctions to compute cooperative policies.

4 From Continuous to Punctual Task Allocation

Cooperative multiagent decision-making problems are composed of individual decision making problems and collective decision making problems. Given a robot i, the individual decision making problem consists in determining all paths to take in order to visit all its attributed tasks G_i. The collective decision making problem consists in distributing tasks between robots in a way that minimizes consuming resources (movements and mission duration) to achieve all the tasks G. Both levels require the agents to be coordinated.

An allocation of tasks $\mathbb{G} = \langle G_1, G_2, \ldots, G_n\rangle$ defines a collection of subsets of tasks, one for each robot $i \in [1, \ldots, n]$. By assuming that a task is assigned to one and only one robot, auction protocols allocate each task g to the robot i maximizing the utility value $u_i(G_i,\ g)$ taking into account other allocated tasks G_i.

4.1 Continuous Coordination

To solve the coordination problem, solutions based on a continuous approach assign one task at a time to each robot and need an active coordination process all over the mission execution. Using utility values computed in a distributed way, Continuous coordination mechanism based on Simultaneous Auctions with single attribution was implemented (Algo. 4.1). This solution attributes a task to each robot at the beginning of the mission and at each time a robot finishes to execute its task.

Algorithm 1 Simultaneous Auctions

Require: List of robots and list of tasks
1: Wait for all (robot × task) utility values.
2: **while** robots and tasks in lists **do**
3: Search the highest utility value in lists.
4: Associate the selected task and robot.
5: Remove selected robot and task from lists.
6: **end while**
7: **return** Pairs of (robot, task). (Max. 1 per robot)

Simultaneous Auctions are started with all robots and the set of available tasks (all the uncompleted and unattributed tasks). Robots compute and communicate their new utility values resulting from their current position. The utility values is computed from a goal constant reward r_g decreased by the movement cost to reach it. The tasks assigned to the free robots (all the robots at the beginning) are pull out the set of available tasks. This mechanism prevents idle time while other robots terminate their tasks and guarantees to select new task with respect to "one per robot" future task attributions.

Simultaneous Auctions allow us to formalize a Continuous coordination process similar to the ones proposed in previous works in the literature (cf. Section 2.4). Furthermore, the proposed solution reaching a complete allocation in Punctual coordination is an augmented version of the Simultaneous Auctions algorithms (cf. Section 5). This way, the differences between Continuous and Punctual coordination will be easily highlighted. From Continuous approach, to reach coordination where several tasks are attributed per robot, the first difficulty is to individually compute utility values regarding different set of attributed tasks.

4.2 Individual Decision Making

The utility values mainly depend on the robot i's policy to reach a set of tasks G_i. In the proposed approach, individual Goal-Oriented Markov Decision Processeses (GO-MDPs) are used to compute policies and associated utility values. GO-MDPs allow each robot to make optimal decisions in order to perform sequentially several tasks. A GO-MDP $\langle S_i, A_i, t_i, r_i \rangle$ is defined like a standard MDP that includes a memory of achieved goals in the state definition.

An optimal GO-MDP policy π^*_{i,G_i} is computed for each robot i from its Topological Map and a set of goal-tasks G_i to perform ($G_i \subset W_i$). A state $s \in S_i$ includes the last recognized waypoints $w_s \in W_i$ and the set of achieved goals $G_s \subseteq G_i$. The actions match the set of paths P_i. An action p_s is added for every waypoint $w_s \in G_i$ as a symbolic action validating that the goal located in w_s is performed:

$$\begin{aligned} S_i &= \{\ s = (w_s, G_s) \mid w_s \in W_i, \quad G_s \subseteq G_i\ \} \\ A_i &= \{\ p_a \in P_i\ \} \cup \{\ p_s = (w_s,\ w_s) \mid w_s \in G_i\ \} \end{aligned} \tag{2}$$

When executing an action p_a from the waypoint w_s, the transition function t returns the probabilities to reach neighbor positions according to the deviation function d_{p_a}. A transition for each validation action reaches the corresponding state where the set of achieved goals is augmented if the waypoint matches a goal location to reach by the robot i (i.e. $w_s \in G_i$):

$$t_i(s, p_a, s') = \begin{cases} d_{p_a}(w_{s'}) & \text{if} \quad G_{s'} = G_s \\ 0 & \text{else} \end{cases} \tag{3}$$

$$t_i(s, p_s, s') = \begin{cases} 1 & \text{if} \quad G_{s'} = G_s \cup (G_i \cap w_s) \\ & \quad \text{and } w_{s'} = w_s \\ 0 & \text{else} \end{cases} \tag{4}$$

The reward function returns a negative value regarding the movement cost related to the path, and a positive constant gain r_g common to all robots if a new task location is reached. Therefore, the GO-MDP structure guarantees that the positive rewards r_g can only be perceived once per goal.

$$\begin{aligned} r_i((w_s, G_s), p_a) &= c_{p_a} \\ r_i((w_s, G_s), p_s) &= \begin{cases} r_g & \text{if} \quad w_s \in G_i - G_s \\ 0 & \text{else} \end{cases} \end{aligned} \tag{5}$$

4.3 Robot Utility Values

Using Continuous approach, a utility value is computed based on the individual expected gain to perform a task. At each time-step, the sets of attributed tasks are composed by zero or one task ($\forall G_i, \quad G_i = \emptyset$ or $\{g\}$). The individual expected gain of a robot i is defined as the Bellman value (Eq. 1) attached to optimal policy to perform $\{g\}$ from the current position $w_{ic} \in W$ of the robot in the environment:

$$u_i(\emptyset, g) = V^{\pi^*_{i,\{g\}}}(w_{ic}) \tag{6}$$

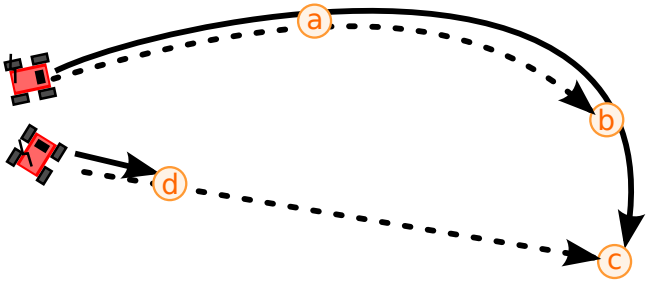


Fig. 4 Minimal movement allocation (dark arrows) versus minimal duration allocation (doted arrows) with 2 robots and 4 goals.

Punctual Coordination aims to find the optimal allocation $\mathbb{G}^*$ in the set of all candidates $D_{\mathbb{G}}$ which maximizes the sum of expected gains.

$$value(\mathbb{G}) = \sum_{i=1}^{n} gn_i(G_i) \tag{7}$$

$$\mathbb{G}^* = \underset{\mathbb{G}\in D_{\mathbb{G}}}{\operatorname{argmax}}(value(\mathbb{G})) \tag{8}$$

The utility value of a task g can be computed by comparing the robot expected gain gn_i when the task g is included or not to its allocation G_i:

$$\begin{aligned} u_i(G_i, g) = gn_i(G_i + g) - gn_i(G_i) \text{ if } g \notin G_i \\ gn_i(G_i) - gn_i(G_i - g) \text{ if } g \in G_i \end{aligned} \tag{9}$$

The utility $u_i(G_i, g)$ of a task $g \in G$ for a robot i matches the difference between the expected gain regarding the referent allocation G_i and the new one G'_i built by addition/subtraction of the task g ($G'_i = G_i + g$ or $G'_i = G_i - g$).

4.4 Altruistic Expected Gain

In Punctual auction coordination, using individual expected gain allows the fleet of robots to reach allocations which minimizes the movement cost but not necessarily the mission duration (Fig. 4). In fact, the minimal movement cost solution does not consider the parallel execution of the mission which might result to allocate all the tasks to only one robot.

We propose to compute the expected gain $gn_i(G_i)$ with an altruistic heuristic consisting in subtracting a social cost from the individual expected gain. The social cost individually evaluates the impact of the robot attribution G_i on the rest of the group. The proposed social cost aims to decrease the time needed to complete the mission by balancing the allocation. It matches the difference between the assignment size $|G_i|$ and an ideal size gs^* ($gs^* = |G|/n$ for example). The more important is the difference, the greater is the social cost.

$$gn_i(G_i) = V_i^{\pi^*_{i,G_i}}(s_{ic}) - oc \sum_{j=0}^{|gs^* - |G_i||} j \tag{10}$$

The social cost is based on the multiplication of an opportunity cost oc constant. The opportunity cost defines the threshold value that allows a robot to unbalance its allocation. The first task unbalancing the allocation costs oc, the second costs $2oc$ and so on. The notion of decreasing individual rewards with opportunity costs has been already used in multi-robot planning of constrained missions [29].

Utility functions are bounded by zero and the maximal possible utility value considering only the individual expected gains. The opportunity cost oc can be defined proportionally to the maximal utility:

$$oc = noc \times \max_{i,\ G_i,\ g}(u_i(G_i, g)) \quad \text{with } noc \in [0, 1] \tag{11}$$

This way, a normalized opportunity cost noc at 1.0 prevents unbalanced allocation. Using this framework, It is presented in the following section the Sequential Simultaneous Auctions protocol which allows a team of robots to quickly and efficiently allocate a set of tasks.

5 Sequential Simultaneous Auctions

GO-MDPs coupled with altruistic expected gain allows the robots to compute utility values according to synergies between tasks. Furthermore, this solution permits to parametrize more or less balanced allocation. However, the evaluation of all allocations to retain the one maximizing the sum of utilities, leads to an exponential enumeration of possibilities and involves an impracticable number of multi-task policy computations.

The proposed Sequential Simultaneous Auctions protocol (*SSA*) aims to allow all the robots to compute new policies involving all unassigned tasks at Punctual Coordination Phases. The *SSA* protocol starts with an initial allocation (possibly empty) and converges to a locally optimal solution deduced from exchanged utility values. At each iteration, *SSA* repeats a Simultaneous Auctions process to search for modifications which improve the built allocation with respect to the utility value function definition (Eq. 9).

5.1 Protocol Definition

In order to evaluate each task utility value for the current allocation, each robot builds and solves several GO-MDPs. *SSA* protocol permits to combine utility

values in order to improve the allocation. The process is iterated until no more improvement in the allocation is found. *SSA* is split into 5 steps:

1) Opening: SSA is opened with a robot demand which becomes the manager. A demand results from modifications in the individual knowledge which induce updates in the set of tasks and utility values. This step permit to list the participants. The participants are all the robots with an efficient communication connection with the manager. Once registration is done, robots can not enter or leave the *SSA* before the end of the protocol. This prevents robots to move out of communication ranges.

2) Task identification: This step consists in taking inventory of all uncompleted tasks by the communicating robots. Step 2 also allows robots to merge knowledge and compute shared parameters such as the opportunity cost constant. Finally, a heuristic initializes the allocation $\mathbb{G}$. For example, the initial allocation could be empty, random or based on the existing allocation before the *SSA* opening.

3) Value computation: Each robot i computes its own optimal policies based on GO-MDPs. A GO-MDP is defined for each task g not yet attributed to the robot i ($g \in G - G_i$) at the current iteration in order to evaluate the individual expected gain of adding each of those tasks to the robot i. This step involves $|G| - |G_i|$ policy computations on $2^{|G_i|+1}$ states. However, individual GO-MDPs are strongly similar, that permits to speed up the computing process. The general idea is to reuse policies of previously solved GO-MDPs. These policies allow robots to compute and exchange their current utilities regarding all tasks in G.

4) Allocation update: Once the last utility message ($n \times |G|$ messages) is received by the manager, the allocation $\mathbb{G}$ can be updated. This consist in a modified version of the Simultaneous Auctions algorithm (Algo. 4.1) that takes into account already attributed tasks. The allocation is updated by switching a task from a robot (sender), if exists, to another (receiver) with a greater utility value. Task modifications are chosen in sequential order by selecting tasks with the greatest difference between sender and receiver in a manner that induces unique task modification per robot. If at least one update has been done, the protocol returns to Step 3.

5) Closing: When no more updates is possible (robots' utilities do not allow to increase the task allocation) a consensus is found with a locally optimal task allocation. At this moment, robots end the *SSA* Punctual coordination phase and start completing their individual set of tasks.

SSA protocol locks robots during its process. The use of broadcast communication permits the allocation update process to be done separately by each robot in a distributed way without manager. The group has to always wait for all robots' step 3 *value computation* to end before switching to Step 4 *Allocation update*.

SSA protocol is parametrized by the heuristics used to initialize the allocation (Step 2) and the assignment rules that update the allocation from the utilities (step 3). *SSA* protocol permits task allocation between heterogeneous robots. The utility function definition is not necessarily shared by all the robots (while they share similar coherence in reward and cost definitions). Moreover, each robot can have its own individual topological model.

5.2 Convergence

The convergence of *SSA* protocol is guaranteed by the fact that the updated allocation (Step 4) which has a greater value than the previous one. The main reason comes from the following assumptions: the allocation is modified in a way that involves a unique modification per robot at each iteration and the individual gain functions are constant during the *SSA* process.

The demonstration is done by considering that it is always positive for a robot to perform a task (the constant task reward r_g is much greater than movement costs). Thus, adding a task to the allocation of a robot always leads to an improvement in the individual gains (Eq. 10) and the utilities are positive (Eq. 9). This facilitates the proof but it is not a restriction.

Each update based on the utility function (Eq. 9) for a task g, induces an improvement in the utility between old and new individual allocations of the sender and receiver robots. Thus, the loss in gain of the sender is lower than the gain of the receiver. Each update induces that the sum of gains of the group is increased by the sum of the differences in the utilities of the sender and the receiver.

The convergence is conditioned by a unique modification in each robot's allocation at the same iteration in Step 4. The utility function is defined for a single task and the difference in gain function of a robot, in case of several modifications at a time, is not equal to the sum of its utility values. Each modification may lead to utility actualizations.

However, the update of the allocation (step 4) can include several modifications concerning different robots. This way, increasing the number of robots, theoretically, weakly impacts the number of *SSA* iterations. It is also expected that increasing the number of robots

will speed up the *SSA* coordination if gain functions aim at balancing the number of tasks between robots ($oc \simeq$ maximal difference between task individual relevances).

Considering that the expected gain values only depend on the allocations and there is a finite number of possible allocations, there is therefore at least one optimal allocation which maximizes the sum of expected gains (Eq. 7). Thus, the finite number of solutions and the convergence of the allocation value ensure that *SSA* will terminate. Furthermore, the resulting allocation $\mathbb{G}$ is locally optimal regarding the range-1 allocations (allocations built with a single difference in task assignments).

5.3 Desynchronization

The proposed *SSA* protocol is not fully desynchronized and the robots have to continuously wait for all the other robots. In *SSA*, the distribution of the process is limited by Step 4 (allocation update), where the robots have to be synchronized before actualizing a common new allocation. Asynchronous individual processes may lead to of inconsistencies in task assignments (unallocated or multi-allocated tasks).

The *SSA* can be upgraded to a Desynchronized *SSA* protocol (*D-SSA*) based on locking tasks to prevent inconstancies in allocations. The idea consists in adding a mechanism that avoids several robots to take the same task. Considering a task g, the robot i with the higher utility locks the task g and communicates an unreachable utility for g (higher than the maximal one). Later, if another robot j communicates a utility value greater than the hidden one of the robot i, the task g will be unlocked. The robot i removes the task from its assignment and communicates again, its real value. This way, the robot j can take and lock the task at its turn. This mechanism guarantees the coherence of the allocations built by asynchronous robots without forcing synchronization steps.

In *D-SSA* protocol, each robot is focused only on its individual task assignment and not on the global allocation. Even if several robots are interested in the same tasks, the lock mechanism ensures that each task will be assigned to one and only one robot at the end of the process. As *SSA* protocol, it is possible to start *D-SSA* with an empty allocation where no task is assigned. *D-SSA* ends if all robots' processes are in step (4) "allocation update" with no modification on the allocation. This corresponds to a situation where all tasks are allocated and locked.

It is expected that *SSA* and *D-SSA* will converge on similar solutions. However, synchronization on Step 4 ensure *SSA* to be deterministic. *D-SSA* execution and the resulting solution depends on individual duration of utilities' computation and latencies in communication.

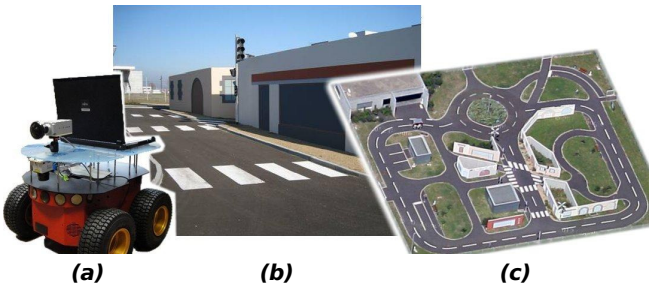


Fig. 5 From left to right: Pioneer robot, the experimental area and its aerial view.

6 Experiments

SSA and *D-SSA* protocols are designed to speed up the mission execution while minimizing robot consumed resources. However, protocols need a finite but undetermined number of iterations to converge to the solution and each iteration requires policy computations. *D-SSA* protocol based on topological decision making is used by fleet of cooperative robots that aim to visit a set of positions (tasks). The task positions are described by an initial efficient topological map. The map and the environment are statics during our scenario and fully operational. In this optimal configuration, the experiment aims at quantifying the number of tasks the fleet of robots can deal with and to estimate how Punctual coordination performs compared to continuous coordination. In Section 7, we will discuss the ability of our approach to deal with dynamic maps (exploration scenarios).

The first series of experiments are both performed in real and simulated conditions to validate the approach coupling the hybrid architecture and the *D-SSA* protocols. These experiments allow us to evaluate the required coordination time in function of the number of robots, the size of the map and the number of tasks. The second series of experiment are simulated and permit to statistically evaluate the efficiency of using GOMDP coupled with the *D-SSA* protocol compared to continuous coordination where one task per robot is allocated and performed at a time. The evaluation concerns both mission duration and individual consumption of resources.

6.1 Evaluating Coordination Time

Several experiments were performed in the context of the R-Discover project[1] in order to validate the approach in real settings. Two scenarios were tested in an open-space environment and in an urban experimental environment (Fig. 5). In both scenarios, a fleet of robots has to visit a set of task locations and the robots have to return to their initial positions.

The scenario in the open-space environment permits to test the Desynchronized Sequential Simultaneous Auctions (*D-SSA*) protocol on real mobile robots (Fig. 6a). Experiments use 3 Pioneer robots equipped with simple odometers, standard laptop computers and Wifi devices. *D-SSA* protocol was configured with an opportunity cost to reach balanced allocation (0.1 time the maximal distance between 2 waypoints). The mission execution is divided into three phases: the initial coordination phase where the robots compute their policies ; the task execution phase consisting in following the computed policies and the return home phase. The Topological Map is built by spreading randomly goal-task waypoints in the free area and adding paths to connect them using reaching target controller.

These scenarios validate the architecture and allow us to estimate the time spent by coordination phases. A robot equipped with an ordinary laptop requires around half a second to compute a policy involving 8 tasks and around 1 second for 10 tasks. During an *SSA* iteration, the evaluation procedure requires to compute several multi-task policies, one for each non-attributed task to the robot. In the experiment, with 3 robots and up to 20 tasks (more than 6 expected tasks per robot) the unique initial coordination phase (considering initially that no task is allocated) reaches 1 minute (22 seconds for 13 tasks in the experiment presented Fig. 6a). The time required by an iteration of the *SSA* protocol grows exponentially with the number of tasks per robot.

The scenario in the urban environment enhances the first scenario by adding static obstacles (sidewalks) (Fig. 5 and 6b) and using the corresponding map (Fig. 3). Robots are equipped with a plan laser at 45 degrees to detect sidewalks that are around one meter in front of them. Experimental results show that the control architecture based on the proposed Topological Map allows robots to maintain their localization in a 32×26 meters environment despite weak and inaccurate local perceptions.

In computer simulations, we increased the number of robots (n) to 10 and kept 4 expected tasks per robot ($|G| = 4n$). We counted the number of modifications in attribution of each robot. A modification matches an iteration of *D-SSA* requiring the actualization of the policies and the utilities. A set of 200 simulations with random task positions has been performed for each considered fleet size ($n \in [2 \dots 10]$). Figure 7 presents the average number of modifications regarding all the robots and regarding only the robot with the highest number of modifications.

[1] French National Research Agency (ANR) R-Discover project videos: www.greyc.fr/node/1629

Fig. 6 Task allocation in free area (left), Task allocation and topological navigation in urban area (right).

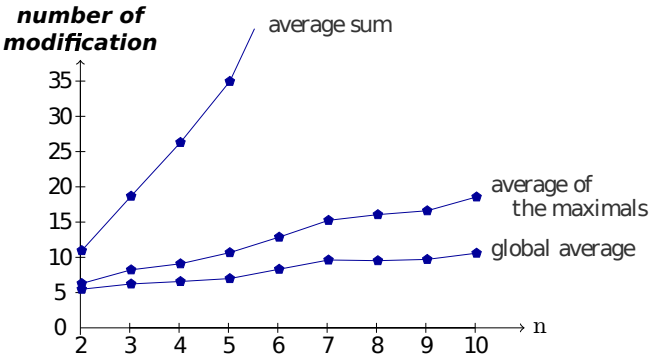


Fig. 7 Average numbers of iterations per robot (with *D-SSA*) by increasing the number of robots and tasks (n robots and $4n$ tasks).

Figure 7 shows that few modifications, in regard to the total number of tasks, are necessary to converge to a range-1 locally optimal allocation. Statistically, this means that each robot's *D-SSA* process leads a number of modifications which is proportional to the number of expected tasks per robot ($|G|/n$).

6.2 Punctual vs. Continuous Coordination

The second series of experiments addresses the interest of Punctual versus Continuous coordination of the fleet of mobile robots. The empirical evaluation focuses on gains in mission duration and sum of movements using *D-SSA* protocol in Punctual coordination. Experiments were performed in ideal settings: static environments and deterministic Topological Maps.

We consider a simulated urban environment (Fig. 8a) similar to the real one (Fig. 3) and a more Labyrinthine

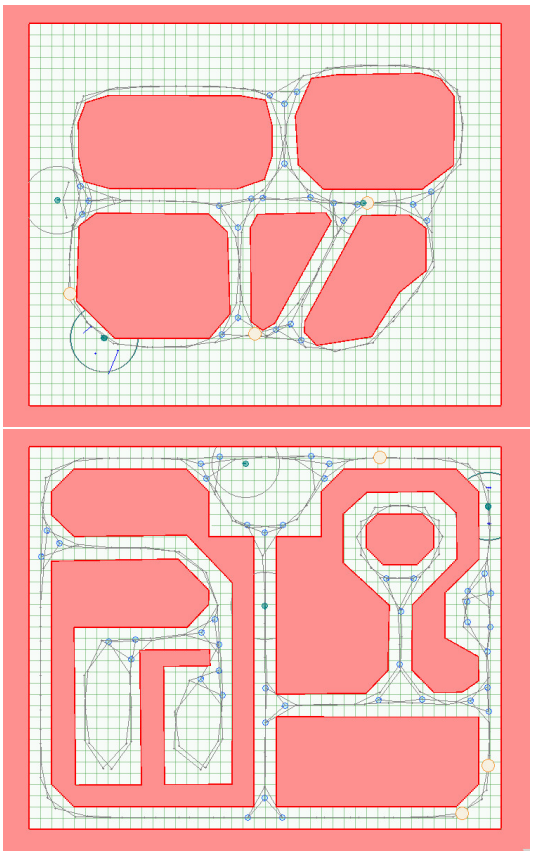
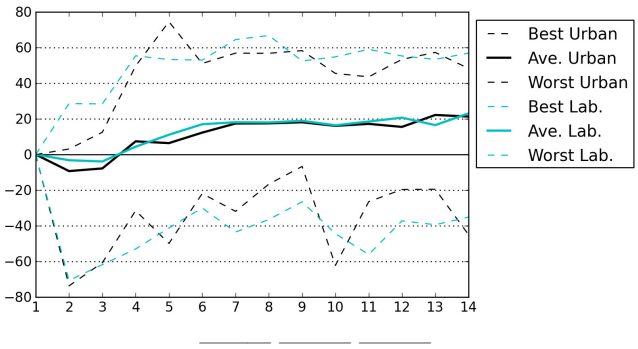


Fig. 8 *Urban* (top) and *Labyrinth* (botom) environments.

Difference regarding the overall mission duration:



Difference regarding the sum of robot movements:

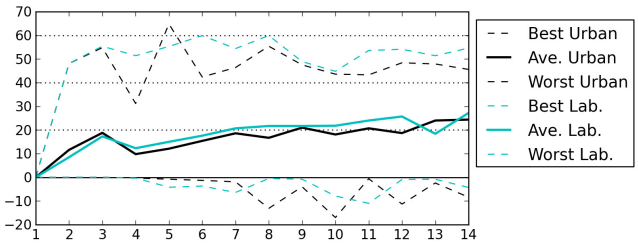


Fig. 9 Best, worst and average Difference in percent by using *D-SSA* protocol in Punctual Coordination despite using Continuous coordination in *Labyrinth* and *Urban* environments.

environment (Fig. 8b). The experiments were done for 3 robots and between 1 to 14 randomly located goals (50 random generations). For each configuration, (we ran $50 \times 13 \times 2$ configurations), we retained the difference in duration time and in the sum of movements' costs using initial *D-SSA* strategy and Continuous auction strategy. The sum of movements' costs is computed considering all the paths taken during the mission execution by all the robots. *D-SSA* is initialized with a normalized opportunity cost fixed to 0.1 and an empty initial allocation.

Figure 9 presents the average difference and the extrema in mission duration times. The difference is given for the *D-SSA* protocol in percentage regarding the duration resulting from Continuous auction strategy. The duration did not take coordination phase into account. In fact, simulations did not use a computer per robot and the computation was not parallelized as in real situations. Figure 9 gives the gains considering the sum of the robots' movements. Those gains are computed in a similar way as the gains in term of durations.

With less than 4 goals, we observe less movements but longer durations. In several configurations, the *D-SSA* protocol unbalances the task allocation while Continuous Coordination forces one task per robot.

For 4 goals and more, the altruistic mechanism (Section 4.3) used in *D-SSA* to coordinate the policies leads, in average, to less movements and shorter missions. The average gains increase slowly to around 20% between 7 to 14 tasks. The results do not highlight significant differences between urban and labyrinthine environments.

Negative effects on the sum of movements using *D-SSA* concern only few configurations. However, negative impacts on duration are more significant and can reach -60% with more than 7 goals. The benchmark of configurations regarding positive, null and negative effects on duration (Fig. 10) shows that, from 9 goals (more than 3 per robot), the duration of *D-SSA* increases in less than one configuration over six.

7 Discussion

The evaluation of the coordination times validates the interest to build the robot decision and control using the Topological Map. The proposed Topological Map permits to model the environment with few waypoints regarding coherent robot perception and control capabilities (Fig. 1). However, the paper does not address the question of simultaneous topological localization and mapping considering the Topological Map pro-

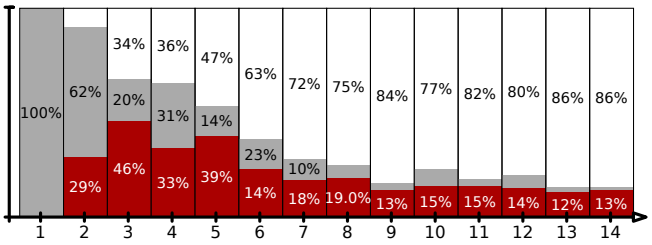


Fig. 10 Percentage of configurations with positive (white), null (gray) and negative (dark red) effects on duration using *D-SSA*.

posed in the architecture. In the proposed experiment the map is provided and the robots decision making handle several task-positions. The computed individual policies are optimal. The Topological Map coupled with Markov Decision Processes offers the possibility to address punctual distributed task allocation.

Compared to the Continuous approach, using Punctual long-term coordination allows groups of robots to decrease both mission duration and the sum of movements. Protocols like *D-SSA* allow the robot to coordinate in a distributed way and can be done by robots equipped with ordinary laptop.

The initial *D-SSA* requires a lot of time to converge to the solution. This time increases exponentially according to the number of tasks per robot and linearly according to the topology size. Despite the few numbers of *D-SSA* iterations, *D-SSA* duration is directly impacted by optimally solving GO-MDPs at each *D-SSA* iteration. The proposed approach is interesting while coordination durations are coherent with the expected gains in the mission execution. This is actually the case in our urban-like domains where a multi-task robots' mission takes several tens of minutes to be performed and requires a unique initial coordination phase of about one minute.

In conclusion, Continuous coordination is more suitable in scenarios characterized by frequent computation as in exploration scenarios with weak initial knowledge. The continuous approach can be enriched to handle lost communication because of moving robots and communication range constraints [30]. On the other hand, while considering mission scenarios with important initial knowledge inducing no or few actualizations (as in simple navigation, in search and rescue or in exploration with a limited unknown areas), approaches based on Punctual long-term coordination can significantly increase the efficiency.

In case of loss in communication, Punctual coordination phases can be processed during the mission at some Rendezvous points defined in a more or less intentional way [8]. Coordination phases that occur during the mission execution can be speed-up considering the current allocation as the initial allocation of the *D-SSA* protocol. Furthermore, in intensional Punctual coordination, it is possible to mix Punctual and Continuous approaches by computing robot policies with only the subset of goals to reach between two Rendezvous points.

Finally, experimental results have shown interesting improvements in mission durations while the social cost heuristic only balances the allocation between robots in term of the number of tasks. In multi-robot missions where only the overall mission duration has to be optimized, coordination will require specific mechanism where the group has to detect the robot with the longest mission and try to minimize its part.

8 Conclusion

This paper presented a control architecture coupled with a distributed policy computation framework used for mobile multi-robot coordination. The robot architecture is based on reactive functional modules allowing the robots to plan their movements regarding the proposed Topological Map. The Topological Map permits the robot to plan the sequence of semantic perception states according to used reactive control. Based on the architecture enriched with Goal-Oriented Markov Decision Process (GO-MDP), each robot individual plan can involve several tasks to perform.

Coordinate distributed policies was achieved by using Auction Protocols. We proposed Sequential Simultaneous Auctions (*SSA*) protocols to allow robots to coordinate their policies regarding all the goal-tasks in Punctual coordination phases. *SSA* protocol is proposed as an heuristic to handle combinatorial auctions. *SSA* protocol converges to a range-1 locally optimal allocation. *SSA* protocol was extended with a Desynchronized protocol (*D-SSA*) more suitable in mobile robot applications.

Experiments validated the proposed approach in real settings. The architecture allows robots to uncouple smooth reactive control and long-term mission planning, thus permitting more flexibility for complex tasks achievement. *D-SSA* protocol coordinates effectively the robots policies involving several tasks per robot with a few number of iterations. Each iteration induces distributed GO-MDPs solving. Finally, experimental results in virtual simulations allow us to conclude about the average reduction on both duration times and movements by using Punctual rather than Continuous coordination.

However, Punctual coordination requires exponential computation resources regarding the number of goals.

In the proposed approach, coordination durations was mainly impacted by optimally solving GO-MDPs in each *SSA* iteration. Future works will deal with developing more efficient approach to solve GO-MDPs while still ensuring that *SSA* converges.

References

1. D. S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of markov decision processes," in *16th Conference on Uncertainty in Artificial Intelligence*, (2000)
2. W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control, Theory and Application*. Springer-Verlag London, (2008)
3. R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed problem solving," *Artificial Intelligence*, vol. 20, pp. 63–109, (1983)
4. B. P. Gerkey and M. J. Mataric, "Sold!: auction methods for multirobot coordination," *Transactions on Robotics and Automation*, vol. 18, pp. 758–768, (2002)
5. W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, pp. 376–386, (2005)
6. L. Matignon, J. Laurent, and A. Mouaddib, "Distributed value functions for multi-robot exploration," in *International Conference on Robotics and Automation*, (2012)
7. M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1957–1962, (2003)
8. H. Hourani, E. Hauck, and S. Jeschke, "Serendipity rendezvous as a mitigation of exploration's interruptibility for team of robots," in *International, Conference on Robotics and Automation*, vol. 1093, pp. 2969–2976, (2103)
9. 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*, vol. 8, pp. 47–63, (1991)
10. S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, (1998)
11. M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, pp. 1257–1270, (2006)
12. M. H. Rothkopf, A. Pekec, and R. M. Harstad, "Computationally manageable combinational auctions," *Management Science*, vol. 44, pp. 1131–1147 (1998)
13. A. Benzerrouk, L. Adouane and P. Martinet, "Lyapunov Global Stability for a Reactive Mobile Robot Navigation in Presence of Obstacles," in *International Workshop on Robotics and Intelligent Transportation System*, (2010)
14. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, pp. 90–98, (1986)
15. J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *International Conference on Robotics and Automation*, vol. 2, pp. 995–1001, (2000)
16. F. Teichteil-Königsbuch and P. Fabiani, "Autonomous search and rescue rotorcraft mission stochastic planning with generic dbns," in *IFIP AI*, pp. 483–492, (2006)
17. A. F. Foka and P. E. Trahanias, "Real-time hierarchical pomdps for autonomous robot navigation," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 561–571, (2007)
18. G. Lozenguez, L. Adouane, A. Beynier, P. Martinet, and A.-I. Mouaddib, "Map partitioning to approximate an exploration strategy in mobile robotics," in *Advances on Practical Applications of Agents and Multiagent Systems*, vol. 88, pp. 63–72, (2011)
19. G. Floros, B. van der Zander, and B. Leibe, "Openstreetslam: Global vehicle localization using openstreetmaps," in *International Conference on Robotics and Automation*, pp. 1054–1059, (2013)
20. H. Korrapati, Y. Mezouar, P. Martinet, and J. Courbon, "Image sequence partitioning for outdoor mapping," in *International Conference on Robotics and Automation*, (2012)
21. H. Choset and J. Burdick, "Sensor based planning, part ii: Incremental construction of the generalized voronoi graph," in *International Conference on Robotics and Automation*, pp. 1649–1655, (1995)
22. R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, pp. 679–684, (1957)
23. I. Chades, B. Scherrer, and F. Charpillet, "A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem," in *ACM symposium on applied computing*, pp. 57–62, (2002)
24. R. Nair, P. Varakantham, M. Tambe, and M. Yokoo, "Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps," in *National Conference on Artificial Intelligence*, p. 7, (2005)
25. C. Tovey, M. Lagoudakis, S. Jain, and S. Koenig, "The generation of bidding rules for auction-based robot coordination," *Multi-Robot Systems. From Swarms to Intelligent Automata*, vol. 3, pp. 3–14, (2005)
26. R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, (1998)
27. R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The claraty architecture for robotic autonomy," in *Aerospace Conference*, vol. 1, pp. 121–132, (2001)
28. R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty," in *Robotics: Science and Systems*, (2007)
29. A. Beynier and A. Mouaddib, "An iterative algorithm for solving constrained decentralized markov decision processes," in *The 31st National Conference on Artificial Intelligence*, pp. 1089–1094, (2006)
30. M. N. Rooker and A. Birk, "Multi-robot exploration under the constraints of wireless networking," *Control Engineering Practice*, vol. 15, no. 4, pp. 435–445, (2007)