2013 IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT)

Simultaneous Auctions for "Rendez-Vous" Coordination Phases in Multi-Robot Multi-Task Mission

Guillaume Lozenguez and Abdel-Illah Mouaddib
Laboratory GREYC, Caen, France
firstname.lastname@unicaen.fr

Lounis Adouane
Institut Pascal, Clermont-Ferrand, France
lounis.adouane@univ-bpclermont.fr

Aurélie Beynier
Laboratory LIP6, Paris 6^{éme}, France
aurelie.beynier@lip6.fr

Philippe Martinet
Laboratory IRCCYN, Nantes, France
philippe.martinet@irccyn.ec-nantes.fr

***Abstract*—This paper presents a protocol that permits to automatically allocate tasks, in a distributed way, among a fleet of agents when communication is not permanently available. In cooperation settings when communication is available only during short periods, it is difficult to build joint policies of agents to collectively accomplish a mission defined by a set of tasks. The proposed approach aims to punctually coordinate the agents during "Rendez-vous" phases defined by the short periods when communication is available. This approach consists of a series of simultaneous auctions to coordinate individual policies computed in a distributed way from Markov decision processes oriented by several goals. These policies allow the agents to evaluate their own relevance in each task achievement and to communicate bids when possible. This approach is illustrated on multi-mobile-robot missions similar to distributed traveling salesmen problem. Experimental results (through simulation and on real robots) demonstrate that high-quality allocations are quickly computed.**

***Keywords*-Multi-robot systems; Task allocation by auctions; Decision making**

I. INTRODUCTION

Allowing a group of autonomous robots to cooperatively achieve goals requires an automatic coordination process of their actions. One challenge in modeling a robot as an agent in a multi-agent system, consists in computing its individual policy that maps the individual actions to perform to each possible succession of local perceptions. The agents' policies should allow the group to efficiently achieve their goals while reacting safely to external events. In case of dynamic and distributed knowledge of the environment (as in exploration scenarios), the policies have to be actualized several times during the missions.

The approach presented in this paper consists in a protocol allowing the agents to coordinate their policies computed in a distributed way. We are interested in cooperative multi-task mission where each task requires only one robot. Individual model allows each agent to evaluate its costs to perform each task. The proposed protocol aims to quickly allocate tasks between the agents in order to minimize the costs to perform all the tasks.

In auction protocols, an item is assigned to the agent that proposes the greatest value. Individual Markov Decision Processes (MDPs) can be used to evaluate bids on tasks in multi-robot missions [4], [18]. Solutions described in the literature allocate one item at a time to a robot in a sequential process. Though, there are often strong values inter-dependencies between items. For mobile robots, costs match movements and the task relevance value increases if other tasks in a close range have to been performed by the same robot [2]. The task relevance values depend on policy computations which match traveling salesman problems. For coordination, the difficulty consists in converging quickly to a solution by testing a minimum number of candidates allocations, each tested allocation induces new individual policy computations.

In the targeted application, the common initial knowledge of robots is built from the data of an UAV and is potentially flawed (Fig. 1). Therefore, a fleet of mobile robots has to visit a set of positions (points of interest declared by a human operator) which identify important uncertain locations to explore. The knowledge (map) is organized as a graph of paths' connectivity. Errors induce knowledge, policy and task evaluation updates during the execution of the mission.

This paper addresses on-line multi-agent policy computa-

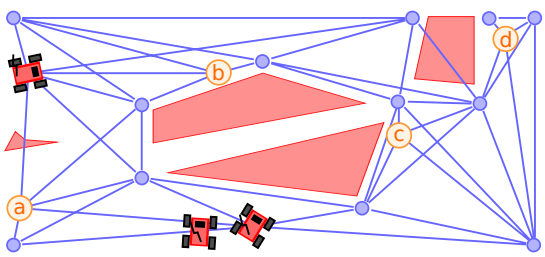

Figure 1. Problem statement: illustration with 3 robots and 4 points of interest $\{a,\ b,\ c,\ d\}$. The initial map (the blue graph) includes errors regarding the real obstacle shapes (in red).

978-1-4799-2902-3/13 $31.00 © 2013 IEEE
DOI 10.1109/WI-IAT.2013.92

tion based on individual MDPs during "rendez-vous" coordination phases. These coordination phases may take place, punctually, several times during missions which consists in distributed tasks achievement. A framework based on Successive Simultaneous Auctions for Coordination (SSAC) is presented to allow agents to evaluate and to exchange several tasks at a time while each agent computes its own policy. We prove the convergence on locally optimal allocation, then, the proposed protocol is experimented on multi-robot missions to statistically guarantee the capability to control coordination phase durations while leading to interesting allocation. Finally, the capability to distribute policy computation is compared with a protocol based on simple sequential auctions.

II. BACKGROUND

This paper describes a distributed planning approach where each agent computes its own policy. The coordination is performed by allocating individual sets of tasks to achieve in a way that minimizes the robot action costs (movements). The proposed approach is divided into 2 parts: individual policy computations and negotiation protocol for task assignments. The coordination between agents consists in solving those 2 parts simultaneously. In fact, comparing several policy values regarding different sets of tasks allows an agent to evaluate its own relevance for each set. The protocol, presented in the next section, aims to minimize the number of individual policy computations in order to speed up coordination phases.

A. Distributed policy computation

A Markov Decision Process (MDP) allows to model a stochastic system in order to compute the control policy which optimizes the expected gain [15]. In distributed models [4], [13], an MDP can be built for each agent where the controlled systems are the agents themselves. 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 is defined as $r : S \times A \to \mathbb{R}$, $r(s, a)$ gives the reward obtained by executing a from 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 [1]. This value depends on a parameter $\gamma \in [0, 1]$ which balances the importance between future and immediate rewards:

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

Commonly, the states match the possible configurations of the agent. For example, for a robot moving in a static environment, with a perfect perception of its movements, the individual MDP states match the positions of the robot in the environment [4], [13]. Because of imperfect perception skills, the agent's state may be defined as its belief state regarding its possible configurations. Partially Observable MDPs (POMDPs) allow each agent to decide of its actions from its observations. POMDPs can be used in robotics when the evaluation of robot movements leads to imperfect localization [18], [5].

In multi-agent systems under distributed control, each agent decides its own actions given its own perceptions. 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 [6], [14]. In such case, any policy actualization of an agent induces modification in the individual transition and reward functions of the other agents. Thus, the agents iteratively actualize their policies until stable coordinated policies are reached.

A policy actualization needs computational resources. Unfortunately, expressive and complete models are difficult to solve since they lead to huge state (MDP) or belief state (POMDP) spaces. It is notably true in case of multi-goal problems (as traveling salesmen problems) where each state has to memorize which goals are achieved or not [11].

B. Coordination by tasks allocation

In consensus approaches [16], cooperation based on auction sales consists in attributing tasks or resources as items among agents. These approaches were successively used in robotics [8]. In "Contract net" [7] or "MURDOCH" [9], 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. The item is allocated to the potential client with the highest offer. Each robot can be manager or potential client and several sales can be made simultaneously.

Bids and attribution rules can be defined differently to optimize the sum of individual interests or to perform fair allocations [19]. 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 robots' auctions for coordination [4]. This way, the value of each task depends on which other tasks the agent has to perform and tasks with interdependencies could be put back in sale several times.

Combinatorial auctions [17], where agents can bid on a set of items, allow to express synergy between items' values. This kind of auctions reaches optimal tasks allocation in a unique simultaneous sale [2] 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 in the number of task combinations.

The difficulty is to allocate several interdependent items with a minimum of individual policy computations in order to allow agents to punctually update their coordination during the mission. In fact, in the problem we tackle, we aim to control coordination phases duration with a weak impact on the solution optimality. This duration directly depends on the number of successive policy actualizations.

III. PROBLEM STATEMENT AND FRAMEWORK

An allocation of goals $\mathbb{G} = \langle G_1, G_2, \ldots, G_n \rangle$ defines the set of goals $G_i \in G$ allocated to the agent $i \in [1,\ n]$ (G the set of all goals). In the addressed problem, a goal matches a task to perform by an agent. Each task (or goal) is assumed to be attributed to one and only one agent. Then, the set of possible allocations $D_{\mathbb{G}}$ is composed by a total of $n^{|G|}$ elements. The value of an allocation $\mathbb{G}$ is defined as the sum of all agents expected gains (gn_i) considering their individual current state, their individual tasks to accomplish and the others tasks allocated to the others agents. An optimal allocation $\mathbb{G}^*$ is an allocation in the set of all candidates $D_{\mathbb{G}}$ which maximizes this sum:

$$value(\mathbb{G}) = \sum_{i=1}^{n} gn_i(G_i),\ \mathbb{G}^* = \underset{\mathbb{G} \in D_{\mathbb{G}}}{\operatorname{argmax}}(value(\mathbb{G})) \quad (2)$$

The expected gain of an agent i corresponds to its individual interest decreased by a social cost. The interest can be computed using the Bellman value (Eq. 1) of an optimal policy from the current state s_{ic}. The social cost aims to individually evaluate the impact of the agent policy on the rest of the group. In task allocation, the proposed social cost aims to decrease the group needed times by balancing the allocation and it matches the difference between the assignment size $|G_i|$ and an ideal size gs^* ($gs^* = |G|/n$ if each task is as important as the others). The more important is the difference, the greater is the social cost:

$$gn_i(G_i) = V_i^{\pi^*_{G_i}}(s_{ic}) - oc \sum_{j=0}^{|gs^* - |G_i||} j \quad (3)$$

The opportunity cost oc defines the threshold value that allows an agent to unbalance its allocation comparatively to the other agents' gains. By this way, the first task which unbalances 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 with interesting results [3].

In sequential auctions for tasks allocation, at each step, a task is assigned to the agent that proposes the greatest utility value. The utility $u_i(G_i, g)$ of a task $g \in G$ is computed for each agent i by adding or subtracting g from its individual current allocation G_i. The utility matches the difference between the expected gain regarding the referent allocation G_i and the new one G'_i built by addition/subtraction of g ($G'_i = G_i + g$ or $G'_i = G_i - g$).

$$u_i(G_i, g) = \begin{cases} gn_i(G_i) - gn_i(G_i + g) \text{ if } g \notin G_i \\ gn_i(G_i - g) - gn_i(G_i) \text{ if } g \in G_i \end{cases} \quad (4)$$

To ensure that all tasks will be performed, the utility to assign a task to an agent has to always be positive. Utility functions are bounded by $[0, maxu]$. The opportunity cost oc can be defined proportionally to the maximal utility $maxu$.

$$oc = noc.maxu, \quad maxu = \max_{i,\ G_i,\ g} (\ u_i(G_i, g)\ ) \quad (5)$$

By this way, an oc value of 100% of $maxu$ prevents to unbalance the allocation. Using this framework, we present an approach which allows us to find a locally optimal allocation in a distributed way and that minimizes the number of policies to evaluate.

IV. MULTI-TASK ALLOCATION

The particularity of auctions for tasks allocation among agents is that bids depend on policies that change while tasks are allocated. Sequential auctions consist in switching a single task from an owner to another at each iteration. This mechanism allows only 1 or 2 agents at a time to re-compute their policies. We propose the Sequential Simultaneous Auctions for Coordination (SSAC) protocol in order to allow all the agents to compute new policies at a time. The SSAC protocol starts with an initial allocation (possibly empty) and converges to a locally optimal solution deduced from exchanged utility values (Eq. 4). At each iteration, SSAC searches all the possible modifications which improve the built allocation with respect to bids computation.

A. Simultaneous auctions for coordination

In order to evaluate each task utility for the current allocation, each agent builds and solves individual MDPs. The SSAC protocol permits to combine utilities in order to improve the allocation. The process is iterated until no more improvment in the allocation is found. The SSAC is splited into 5 steps:

1) Opening: A SSAC is opened with an agent demand which becomes the manager. A demand results from modifications in the individual knowledge of the agent which induce updates in the set of tasks and utilities. This step consists in taking inventory of participants. The participants are all the agents with an efficient communication connection (direct or not) with the manager. Once the registration is done, agents can not enter or leave the SSAC before the end of the protocol. This step is also useful to identify the communication network in order to allow the most efficient message transmission.

2) Task identification: This step allows the agents to identify the set G of tasks to assign. All uncompleted tasks can be reallocated and thus belong to G. This step allows the agents to compute their shared parameters such

as the opportunity cost. The role of the manager is not required when all agents communicate in broadcast. With no manager, deterministic attribution rules have to be defined in order to guarantee that all the agents will build the same allocation. Finally, a heuristic initializes the initial allocation $\mathbb{G}$. For example, the initial allocation could be empty, random or based on the existing allocation before the SSAC opening.

3) Value computation: Each agent computes its own optimal policies based on MDPs it builds in consideration of its individual current allocation $G_i \in \mathbb{G}$. Each MDP is built for a set G'_i based on the G_i set of tasks assigned to the agent i and more or less one task. This step, involves $|G| - |G_i|$ policy computations on $2^{|G_i|+1}$ states. However, the MDPs of an agent are strongly similar, that permits to speed up the computing process. The general idea is to reuse policies of previously solved MDPs. These policies allow agents to compute and exchange their current utilities regarding all tasks in G.

4) Allocation update: Once the last utility message is received by the manager or by all the agents ($n.|G|$ messages) the allocation $\mathbb{G}$ can be updated . The allocation is updated by switching a task from an agent (sender), if exists, to another (receiver) with a greater utility. Task modifications are chosen in a sequential process by selecting tasks with the greatest difference between sender and receiver utilities in a manner that induces a unique task modification per agent. If at least one update has been done, the protocol returns to step 3.

5) Closing: When no update is no more possible (The agent utilities do not allow switch on task allocations) a consensus is found (with a locally optimal task allocation). At this moment, agents end the SSAC and start completing their individual set of tasks.

The SSAC protocol locks agents during its process, it is designed for missions with punctual efficient communication phases. Under this hypothesis, the difference between peer to peer communication and broadcast communication costs is insignificant. The broadcast communication permits the allocation update process to be done separately by each agent while they wait for all agents' step 3 to end.

The SSAC 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). The SSAC protocol permits task allocation between heterogeneous agents. 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 MDP model.

B. Convergence

The convergence of SSAC protocol is guaranteed by the updated allocation (step 4) which has a greater value than the old one. The main reasons come from the following assumptions: the allocation is modified in a way that involves a unique modification per agent at each iteration and the individual gain functions are constant during SSAC process. The demonstration is done by considering that it is always positive for an agent to perform a task. Thus, adding a task to the allocation of an agent always leads to an improvement in the individual gains (Eq. 3) and the utilities are positive (Eq. 4). That facilitates the proof but it is not a restriction.

Each update based on the utility function (Eq. 4) for a task g, induces an improvement in the utility between old and new individual allocations of the 2 agents (sender and receiver). 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 difference in the utilities of the sender and the receiver.

The convergence is conditioned to a unique modification in each agent's allocation at the same iteration of the step 4. The utility function is defined for a single task. The difference in gain function of an agent, 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 agents. By this way, increasing the number of agents, theoretically, weakly impacts the number of SSAC iterations. It is also expected that increasing the number of agents will speed up the SSAC coordination if gain functions aim at balancing the number of tasks between agents ($oc \simeq$ maximal difference between task individual relevances).

Considering that for all SSAC iterations the individual gain values of the allocations are constant 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. 2). Thus, the finite number of solutions and the convergence of the allocation value ensure that the succession of simultaneous auctions 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).

C. Desynchronization

The proposed SSAC protocol for task allocation has two main limitations. It is not fully desynchronized (the agents have to continuously wait for all the other agents) and the solution is only range-1 optimal. In SSAC, the process distribution is limited by step 4 (allocation update), where the agents have to be synchronized before actualizing a common new allocation. Asynchronous individual processes induces the possibility of inconsistencies in task assignments that will conclude on unallocated or multi-allocated tasks.

The SSAC can be upgraded to a desynchronized SSAC protocol (D-SSAC) based on locking tasks to limit errors in allocations. The idea consists in adding a mechanism that prevents several agents to take the same task. At each time

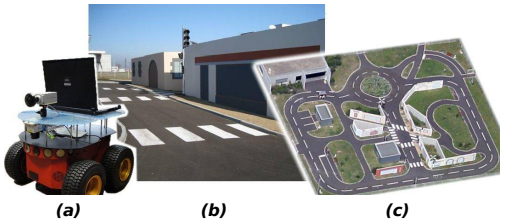

Figure 2. (a) one pioneer robot, (b) the experimental area and (c) the Google aerial view.

Figure 3. Experiments in free area.

Figure 4. Experiments in urban area.

step, the agent i, with the higher utility, locks the task g by communicating an unreachable utility for g (higher than the maximal one). A predefined hierarchy between the agents allows them to disambiguate situations where several agents lock the same task at the same time.

If another agent j communicates a utility greater than the hidden utility of the agent i, the task g will be unlocked. The agent i subtracts the task from its assignment and communicates again, its real value. By this way, the other agent j can take and lock the task at its turn. This mechanism guarantees the coherence of the allocations built in a protocol with asynchronous agents such as robots.

In D-SSAC protocol, each agent is focused only on its individual task assignments and not on the global allocation. Even if several agents are interested in the same tasks, the lock mechanism ensures that each task will be assigned to one and only one agent at the end of the process. By this way, it is also possible to start the D-SSAC with an empty allocation where no task is assigned. D-SSAC ends if all agents' processes are in step (4) "allocation update" with no modification on the allocation and all tasks are allocated.

D. Range-m optimality

The second limitation also exists in sequential auction protocols and concerns the range-1 optimality of the solution. Most of the time, range-1 optimality builds coherent and acceptable allocations but, this limitation can lead to poor results if a balanced allocation is expected ($oc \simeq$ maximal difference between task individual relevances).

In fact, with an already balanced allocation, it is impossible for an agent to remove a task from its allocation, in order to take another one in the next steps. It is possible to increase the optimality range guarantee to m by considering utilities with at most m task modifications. This will induce a number of utilities $|U_i|$ that is exponential in m (U_i the set of considered utilities):

$$U_i = \{u_i(G_i, g), \ldots, u_i(G_i, g, \ldots, g_m)\} \qquad (6)$$

In a range-m succession of simultaneous auctions for coordination protocol, the allocation update (steps 4), consists in searching up to m allocation modifications per agent with a maximal sum of combinatorial bibs. Nonetheless, range-m combinatorial auctions make the allocation update (step 4) more difficult. The updates are computed from $n.|U_i|$ communicated values and lead to an exponential number of combinations of task exchanges between agents. The range dimension m has to be defined as small as possible and such that it guarantees efficient allocation processes.

V. EXPERIMENTS

The proposed approach is illustrated by the need for a fleet of cooperative robots to visit a set of points of interest given an initial approximate topological map. The mission execution is divided into 2 kinds of phases: the individual execution task phases and the coordination phases. The proposed approach D-SSAC is used in coordination phases. A first coordination phase is initialized with all the robots of the fleet at the beginning of the mission. Several other coordination phases take place with a sub-group of robots when robots meet each other during task execution.

D-SSAC was tested on 3 real mobile ground robots during the initial coordination phase (cf. videos[1]) in an experimental indoor free area (Fig. 3) and outdoor urban area (Fig. 2 and 4). In the rest of the paper, we present experiments showing that SSAC and D-SSAC allows robots to coordinate their policies with interesting characteristics.

A. Topological map and decision making

A topological map $\langle W,\ P\rangle$ (Fig 1) is a graph where: nodes W represent particular way-points; edges P represent the paths connectivity between nodes [10]. Way-points match particular positions in the environment where the robot is able to localize itself. The path achievement is uncertain and can end in unexpected way-points. In fact, we consider that the robots are equipped with a module allowing them to reactively move along a path p with deviation probabilities ($d_p(w) \in [0,\ 1]$) [12]. The deviation models the difference between an expected movement and its real outcome regarding the way-point set W. The robots set of goals G match a subset of the nodes to visit (named the points of interest).

Using this architecture, the decision making problem consists in building a policy mapping each node in the map to a path to take. In case of multi-task achievement, this path also depends on the current step in task achievement. An

[1] R-Discover project: www.greyc.fr/node/1629

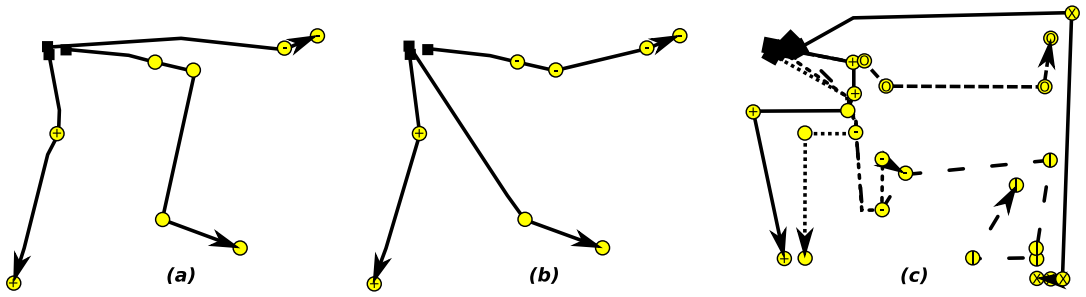

Figure 5. The 2 tested maps and examples of coordinated policies: (a)(b) with few obstacles and (c) the labyrinth

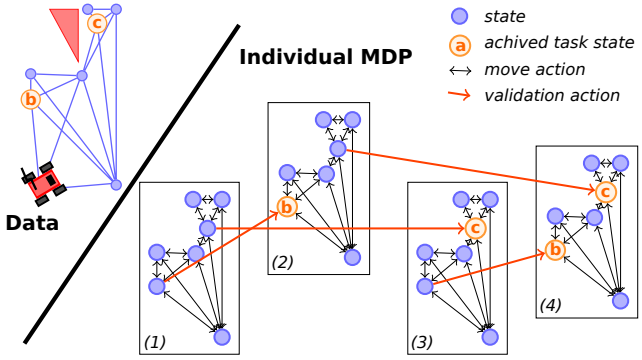

Figure 6. Example of states and actions enumeration for an individual MDP oriented by 2 goals b and c. (1) $\forall s \in S_i,\ G_s = \emptyset$, (2) $\forall s \in S_i,\ G_s = \{b\}$, (3) $\forall s \in S_i,\ G_s = \{c\}$, (4) $\forall s \in S_i,\ G_s = \{b, c\}$.

individual Goal Oriented MDP (GO-MDP) $\langle S_i, A_i, t_i, r_i \rangle$ is defined for each robot from the current knowledge shared in the topological map and a set of goals G_i as points of interest to visit $G_i \subset W$. A state $s \in S_i$ includes the last recognized way-points $w_s \in W$ 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 all way-points $w_s \in G_i$ as a symbolic action validating that the way-point w_s is visited (Fig. 6).

$$\begin{aligned} S_i &= \{\ s = (w_s, G_s) \mid w_s \in W_i, \quad G_s \subseteq G_i\ \} \\ A_i &= \{\ p_a \in P_i\ \} \cup \{\ p_s = (w_s,\ w_s) \mid w_s \in G_i\ \} \end{aligned} \tag{7}$$

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

$$\begin{aligned} t_i((w_s,\ G_s),\ p_a,\ (w_{s'},\ G_s)) &= d_{p_a}(w_{s'}) \\ t_i((w_s,\ G_s),\ p_s,\ (w_s,\ G_s \cup (G_i \cap w_s)) &= 1 \end{aligned} \tag{8}$$

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 point of interest is reached. By this way, the MDP structure guarantees that the positive rewards r_g can only be perceived once per goal.

$$\begin{aligned} r_i((w_s,\ G_s),\ p_a) &= c_{p_a} \\ r_i((w_s,\ G_s),\ p_s) &= r_g \ \textit{if}\ \ w_s \in G_i - G_s \mid \textit{else}\ 0 \end{aligned} \tag{9}$$

B. Experiments on optimality

A first series of experiments aims to validate the efficiency of the range-1 D-SSAC protocol. From 9600 experiments, the value of the D-SSAC allocation is compared to the maximal and minimal allocation values computed in an exhaustive centralized way (Eq. 2). Optimality is defined regarding the maximal value of the possible task allocations (cf. Eq. 2). Each experiment involves 3 robots placed in one of 2 different maps (Fig. 5) and between 2 and 13 points of interest (tasks). Therefore, for 13 tasks, there are 3^{13} possible allocations and between 50×2^{13} and 70×2^{13} (regarding $|W|$) states in the GO-MDPs involving all tasks.

The cost of each path p is defined as the distance between the start and targeted way-points of p. The maximal cost for reaching a target is defined as the environment diagonal (51.2). The task achievement reward (r_g in Eq. 9) is set to 1000 for all tasks in order to guarantee that each task will be reached. Finally, the normalized opportunity cost (noc in Eq. 5) is set to 0 or 0.1.

The experiments are classified regarding the type of map used, the number of tasks considered, and the value assigned to the opportunity cost. The series of experiments is based on 200 random generations of task positions for each of the 48 class of experiments. Considering the value of an allocation as the sum of the associated individual expected gains (Eq. 2), a score (in percent) is computed for each built allocation by comparing its value with the worst (0%) and the best (100%) allocation values. For example, one of the experiments in the class (labyrinth, $|I| = 4$, $oc = 0$) gave the values: 3891.11 (best allocation) ; 3836.41 (worst allocation) ; 3890.84 (D-SSAC allocation) so, a score of 99.5% for D-SSAC. Table I presents averages of obtained scores.

The obtained scores allow us to validate that allocations built using D-SSAC protocol are close to optimal ones in restricted sizes of problems. We denote only 2 experiments over 9600 with scores less than 70%. On the other hand, the probability to find an optimal allocation decreases when the

Table I
OBTAINED SCORES ON RESTRICTED SIZES OF PROBLEMS

$\|I\|$	few obstacles, $noc = 0$			few obstacles, $noc = 0.1$		
	Score		Opt.	Score		Opt.
	D-SSAC	Wo.		D-SSAC	Wo.	
2-3	99.5	85.8	89.5	99.6	80.0	92.5
4-5	99.0	77.5	80.3	99.0	47.4	69.3
6-7	98.9	83.3	71.8	99.2	82.6	47,8
8-9	98.4	86.1	63.8	99.2	86.5	45.0
10-11	98.1	72.7	57.0	99.3	87.6	44.5
12-13	98.4	83.9	55.7	99.2	91.6	37.0
	labyrinth, $noc = 0$			labyrinth, $noc = 0.1$		
2-3	99.7	89.0	97.8	99.5	78.7	86.8
4-5	99.7	77.8	85.5	99.7	88.8	63.0
6-7	99.5	81.4	73.6	99.2	75.8	42.0
8-9	99.3	85.5	67.2	99.0	75.0	34.0
10-11	99.0	85.1	53.8	98.8	82.8	32.3
12-13	98.8	83.2	44.5	98.8	86.8	23.0

(D-SSAC): average scores of the D-SSAC allocations, (Wo.) Worst obtained score by D-SSAC. The column (Opt.) gives the percentage of experiments where D-SSAC built an optimal allocation (score $= 100\%$).

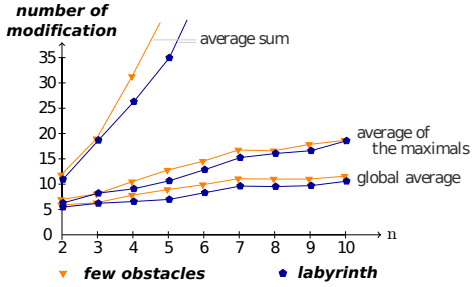

Figure 7. Average numbers of useful iterations per robot (with D-SSAC) by increasing the number of robots and tasks (n robots and $4n$ tasks).

probability to fall into a local optima grows. It is particularly true in the labyrinth environment.

C. Experiments on scalability

A second series of experiments is proposed to validate the capacity of the Desynchronized Simultaneous Auctions for Coordination (D-SSAC) protocol to deal with large fleets of robots. The experiments involve up to $n = 10$ virtual robots, an average of 4 tasks per robot ($|G| = 4n$) and a normalized opportunity cost fixed to 0.1. An optimal allocation is in a set of 10^{40} possible solutions. Figure 5(c) presents experimental results for 6 robots and 24 tasks in the labyrinth (to show the allocations, the robot paths drawn represent the movement done in case of no deviation.)

Using D-SSAC, we count the number of individual modifications for each robot in the initial coordination phase and no task allocated yet. Each modification is associated to policy computations for all the sets built with the new task assignments and one more or less task. The computations are followed by the broadcast of the new utilities. A set of 200 experiments with random task positions has been performed for each considered fleet sizes. Figure 7 presents average numbers of modifications regarding all the robots and only the robot with the highest number of modifications.

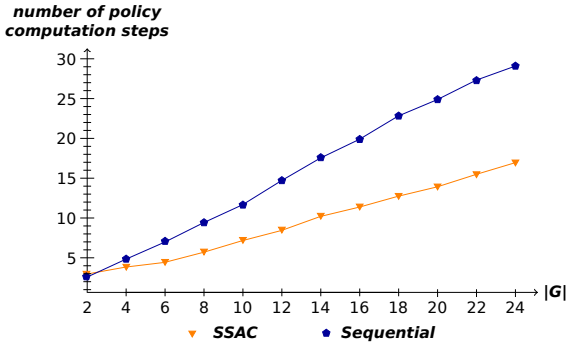

Figure 8. Sequential auctions versus SSAC: number of paralleled policy computation steps.

It is possible to conclude that few modifications in regard to the total number of tasks are necessary to converge to a range-1 locally optimal allocation. This statistically means that each robot's D-SSAC process uses a number of modifications which is proportional to the number of expected tasks per agent ($|G|/n$).

D. Comparison with sequential auctions

The third series of experiments aims to compare the efficiency and the convergence speed of SSAC with sequential auction protocols. The experiments were performed in the labyrinth environment (Fig. 5(c)) by considering a fleet of 6 robots. The normalized opportunity cost is defined to 0.1 (10% of the theoretical upper bound) in order to encourage, without forcing, balanced allocations. The number of goals is increased successively from 2 to 24.

200 simulations have been done for each considered number of goals by randomly positioning the goal points of interest in order to compare the allocations built by the 2 protocols. The comparison mainly concerns the number of iterations and the number of commutations in the allocation. In fact, in sequential protocol, each iteration does not involve commutation and policy actualizations.

The comparison between the 2 protocols of the number of needed policy computation steps requires to count the number of steps with parallelized policy computations (Fig. 8). In the sequential protocol, the number of computation steps matches the number of commutations and involves only 2 robots. In SSAC, this number matches the number of iterations and each iteration involves several commutations at a time which induces more parallelized policy computations.

Thus (Fig. 8), in these experiments (labyrinth, 6 robots and between 2 and 24 goals), the number of sequences of policy computations in SSAC is decreased by 42.9% comparatively to sequential auctions. It is an important result since the critical point in distributed coordination process consists in actualizing the individual policies regarding different sets of parameters.

VI. CONCLUSION

This paper details the Sequential Simultaneous Auctions for Coordination (SSAC) protocol based on goal-oriented MDP bids valuation allowing multi-agent coordination in a distributed way. In order to model uncertain outcomes of actions, we proposed to use GO-MDPs that allow agents to compute their expected values regarding possible allocations. SSAC protocol converges and yields a range-1 optimal coordination while task and/or resources can be assigned punctually and separately to agents during "rendez-vous" phases. The use of this solution is illustrated by the problem where n robots have to visit a set of points of interest. The SSAC protocol is extended with a desynchronized protocol (D-SSAC) more suitable to mobile robot applications.

Experimental results in virtual conditions allow us to conclude about the quality of the solutions and the efficiency of the distributed process. In fact, the computed task allocations are close to optimal ones in restricted sized problems. The distributed process seems to be linear, in the computing resources, when increasing the number of robots and, proportionally, the number of goals. Furthermore, comparative experiments exhibit the relevance of using simultaneous auctions rather than sequential auctions in case of costly distributed policy computations. Experimental results prove the capability of SSAC to better parallelize the coordination process by allowing several pairs of agents to improve their cooperative policies at a time.

Future works aim at studying different frequencies of coordination phases in exploration missions and aim at extending the SSAC protocols to cooperation problems where coordination variable domains are not the agents themselves. For example, tasks and resources allocation between robots could depend on common behavior rules definitions (priorities rules, etc.).

REFERENCES

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

[2] 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*, volume 2, pages 1957–1962, 2003.

[3] A. Beynier and A. Mouaddib. An iterative algorithm for solving constrained decentralized markov decision processes. In *The 31st National Conference on Artificial Intelligence*, pages 1089–1094, 2006.

[4] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21, 2005.

[5] J. Capitan, M. T. J. Spaan, L. Merino, and A. Ollero. Decentralized multi-robot cooperation with auctioned pomdps. In *International Conference on Robotics and Automation*, pages 3323–3328, 2012.

[6] 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*, pages 57–62, 2002.

[7] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983.

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

[9] B. P. Gerkey and M. J. Mataric. Sold!: auction methods for multirobot coordination. *Transactions on Robotics and Automation*, 18, 2002.

[10] 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*, 8:47–63, 1991.

[11] 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*, volume 88, pages 63–72, 2011.

[12] G. Lozenguez, L. Adouane, A. Beynier, A.-I. Mouaddib, and P. Martinet. Map partitioning to approximate an exploration strategy in mobile robotics. *Multiagent and Grid Systems*, 8:275–288, 2012.

[13] L. Matignon, J. Laurent, and A. Mouaddib. Distributed value functions for multi-robot exploration. In *International Conference on Robotics and Automation*, 2012.

[14] 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*, page 7, 2005.

[15] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[16] W. Ren and R. W. Beard. *Distributed Consensus in Multi-vehicle Cooperative Control, Theory and Application*. springer, 2008.

[17] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44, 1998.

[18] M. T. J. Spaan, N. Goncalves, and J. Sequeira. Multirobot coordination by auctioning pomdps. In *International Conference on Robotics and Automation*, pages 1446–1451, 2010.

[19] 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*, 3:3–14, 2005.