Deep Q-Learning-Based Dynamic Management of a Robotic Cluster

Paul Gautier, Johann Laurent, Jean-Philippe Diguet

Deep Q-Learning-based Dynamic Management of a Robotic Cluster

Paul Gautier, Johann Laurent and Jean-Philippe Diguet, *Senior Member, IEEE*

***Abstract*—The ever-increasing demands for autonomy and precision have led to the development of heavily computational multi-robot system (MRS). However, numerous missions exclude the use of robotic cloud. Another solution is to use the robotic cluster to locally distribute the computational load. This complex distribution requires adaptability to come up with a dynamic and uncertain environment. Classical approaches are too limited to solve this problem, but recent advances in reinforcement learning and deep learning offer new opportunities. In this paper we propose a new Deep Q-Network (DQN) based approaches where the MRS learns to distribute tasks directly from experience. Since the problem complexity leads to a curse of dimensionality, we use two specific methods, a new branching architecture, called Branching Dueling Q-Network (BDQ), and our own optimized multi-agent solution and we compare them with classical Market-based approaches as well as with non-distributed and purely local solutions. Our study shows the relevancy of learning-based methods for task mapping and also highlight the BDQ architecture capacity to solve high dimensional state space problems.**

***Note to Practitioners*—A lot of applications in industry like area exploration and monitoring can be efficiently delegated to a group of small-size robots or autonomous vehicles with advantages like reliability and cost in respect of single-robot solutions. But autonomy requires high and increasing compute-intensive tasks such as computer-vision. On the other hand small robots have energy constraints, limited embedded computing capacities and usually restricted and/or unreliable communications that limit the use of cloud resources. An alternative solution to cope with this problem consists in sharing the computing resources of the group of robots. Previous work was a proof of concept limited to the parallelisation of a single specific task. In this paper we formalize a general method that allows the group of robots to learn on the field how to efficiently distribute tasks in order to optimize the execution time of a mission under energy constraint. We demonstrate the relevancy of our solution over market-based and non-distributed approaches by means of intensive simulations. This successful study is a necessary first step towards distribution and parallelisation of computation tasks over a robotic cluster. The next steps, not tested yet, will address hardware in the loop simulation and finally a real-life mission with a group of robots.**

***Index Terms*—MRS, Task Distribution, Robotic Cluster, Multi-Agent Systems, Reinforcement Learning, Deep Q-Learning.**

I. Introduction

THE robust and flexible nature of multi-robot systems (MRS) makes them particularly suitable for critical tasks in dynamic and uncertain environments such as search and rescue (SAR) missions. A MRS requires precise coordination of the robots to efficiently distribute the work, leading to two well-known problems: multi-robot tasks allocation (MRTA) [1] and online planning [2]. These two problems have been the subject of substantial attention and many solutions have been proposed. During a mission (e.g SAR), a MRS must detect and analyze the environment uncertainty at run-time in order to meet security and precision requirements. It relies on algorithms with an increasing complexity that are required to improve detection, analysis and decision-making with a growing sensor infrastructure. In this regard the quantum leap in computer-vision performance is highly significant [3].

However these new and growing computation-intensive tasks represent a major obstacle to the development of MRS. Indeed, despite the improvement in the processing and storage capacities of embedded systems, the computing resources of mobile systems remain inherently limited. The distribution of these computing tasks is the problem we address in this study.

Two approaches have been proposed to overcome this computing limitation. The first one relies on a computing server, that can be based on a ground or cloud-robotic implementation, to offload computation-intensive tasks [4]. Unfortunately, this method is heavily dependent on system capacity to communicate with the server and may result in incompatible processing latency when response time is critical. This solution is particularly impractical in a SAR context where the MRS must evolve autonomously to compensate for the loss or lack of communications structures.

The second approach introduces the concept of Robotic Cluster to speed up complex computational tasks such as SLAM as detailed in [5], [6]. This work clearly shows the possibility of taking advantage of the computational resource multiplicity to parallelize a complex task by mean of load balancing over a cluster of interconnected robots.

By extending this work to several independent tasks, we propose to create a system where the pooling of resources improves the processing ability of each single robot to perform complex tasks. If we consider a set of processing tasks to be executed by the MRS, the new issue to solve is an allocation problem of shared resources where response time and processing cost become the main constraints. This question is a variation the MRTA problem considering processing tasks (MRpTA) [7].

In this paper, we study the distribution problem of the parallelization of complex computing tasks over a MRS evolving in a changing context. As a typical case, we consider the evolution of SAR missions. MRS are designed with state of the art embedded systems, which are based on hetero-

Paul Gautier and Johann Laurent are with Lab-STICC, UMR6285 CNRS, Université Bretagne Sud, Lorient, France. e-mail:{firstname.lastname}@univ-ubs.fr

Jean-Philippe Diguet is with CROSSING, IRL2010 CNRS, Adelaide, Australia, email: Jean-Philippe.Diguet@cnrs.fr

Manuscript received April 19, 2005; revised August 26, 2015.

geneous sensors and multicore architecture equipped with GPUs and dedicated co-processors, and therefore complex to model. In addition, SAR missions take place in dynamic and uncertain environments requiring adaptation and resilience to compensate for the lack of information (unknown obstacles, target location and intermittent communications) as well as for sudden changes (failure, real-time detection).

Based on these observations, we consider reinforcement learning (RL) to solve the distribution problem. RL has been successfully applied to many fields such as power management, server task scheduling and communications optimization and was recently boosted by the success of Deep Q-Learning on games such as Alpha Go [8]. These successes were the result of the combination of RL and neural networks that allow the handling of large-scale continuous environment and thus opening up many new possibilities. Our problem requires dealing with a high-dimensional discrete action space and it is challenging since the number of actions, that must be explicitly represented, increases exponentially with the dimensionality of the actions [9]. Recently several solutions have been proposed such as branching networks, cooperative multi-agent or sequential prediction. In this work, we explore the viability of a solution based on Branching Dueling Q-Network (BDQ) to solve a distribution problem of MRpTA parallelization considering a SAR mission. We address the following questions:

- How can the parallelization of computing tasks over multiple robots improve the system performances?
- In a fully decentralized system, can robots learn how to effectively manage parallelization of computing tasks ?

Our study results in the three subsequent contributions:

- We show that deep RL can be successfully applied to solve the problem of the parallelization of processing tasks over a robotic cluster (MPpTA) in a distributed way without requiring additional centralization.
- We model a realistic MPpTA problem for a MRS composed of Unmanned Surface Vehicles (USV) conducting a SAR mission, which allows us to run extensive simulations.
- We use auctions for job allocation and compare two candidate approaches capable of dealing with high-dimensional discrete action space problems: a dedicated BDQ and two versions of a multi-agent method. In addition our results also include a classical market-based and a fully local baseline methods. We show the advantage of adopting BDQ and highlight the limits of the multi-agent approach when the number of agents increases.

The rest of this paper is organized as follows, Sec. II discusses relevant work on MRTA, SAR and RL. Sec. III explains our problem modeling. Sec IV presents our approach to solve it. Sec. V describes our experimentation set-up. Experimental results are discussed in Sec. VI. Finally, we conclude and introduce our future work.

II. Related work

A. Multi-robot systems and drones background

1) Multi-robot tasks allocation problem: Robots in our system are independent so we have no direct control over the tasks they perform. Our problem is thus not exactly a MRTA one. But we control the task parallelization so we can rephrase is as "which robots should speed up which tasks?", which is close to a MRTA problem.

The literature provides many examples of MRTA problems applied to different contexts. For the sake of theorizing and globalization, Gerkey and Mataric [1] have proposed a taxonomy for these problems. Based on their definition, our architecture is multitasking, single-robot, instantaneous assignment (MT-SR-IA). However it is closer to MT-MR-IA (multi-robots) since tasks must be distributed over multiple robots to be executed more efficiently. In this decentralized context, we propose to use a market-based approach [1] [10].

In our case, a drone starting a new task can sell jobs to workers in order to maximize task efficiency by means of distribution. Sequential single item auctions are commonly used to solve task allocation problem [11]. Although effective, it is not suitable when offers must be allocated by block or when a global view of the market is needed to put in competition both supply and demand. So we opt for double auctions similar to stock exchange where all available job offers are on one side and all capacity offers on the other side. The bids on each side are in competition allowing simultaneous selection of both the most critical jobs and the most appropriate drones without requiring multiple bidding rounds. These auctions are conducted through an auctioneer which determines the price and therefore the job distribution.

2) Search and rescue mission: Although our work is not directly based on pre-existing SAR work, it is inspired by several works for the definition of tasks such as online planning [12], target search [13] and LSAR [14].

3) Drone energy consumption models: Defining an energy consumption model is difficult for two reasons. First, the energy consumption comes from both the embedded system (computing resources, WiFi and sensors) and the drone engine. Second, the power consumption of the drone propulsion strongly depends on the environment and speed. Additionally, the consumption of the embedded system depends on the processing and communication loads. Therefore, it appears very unlikely to come out with a model that will be both accurate and generic. However our study concerns the distribution principle so realistic models are sufficient to perform fair comparisons between different solutions.

Regarding the embedded system, we use as reference, the very complete study presented in [15]. For the drone, we consider small autonomous vessels. We choose the widely used Heron USV and use the characteristics of its engine [16].

4) Reinforcement learning and robotics: Reinforcement learning has been successfully applied to a wide variety of robotic problems such as grasping [17], [18], obstacle avoidance [19] and path planning [20].

However, its application to multi-agent systems remains limited and complex due to the non-stationary environment

introduced by simultaneous learning agents [21]. Recently, several RL-based approaches have been proposed to solve various decentralized MRS problems such as exploration [22], collaborative decision-making [23] and MRTA [24]. These solutions rely on a partial or total centralization of learning. Our approach relies on the continuous training of a smart auctioneer which is not attached to a particular drone and can be simply secured with backups on one or more other drones. Compared to our work, [22], [23] address different problems than MR(p)TA and [24] assumes single-task robots with delayed allocation and does not consider robotic clusters or task parallelization. Finally our solution is meant to support high dimensional action spaces.

B. Deep Reinforcement Learning

By learning through experiences, reinforcement learning (RL) offers a flexible and adaptive approach. Our RL solution must learn to conduct auctions (by filtering and sorting bids) in order to find the right balance between distribution/acceleration, computational load and autonomy. The objective is to let the AI adapt its policy according to the area while accommodating the uncertainty related to the task generation. Unlike other RL implementations in SAR domain, our approach proposes to (partially) coordinate the MRS without resorting to centralization or increasing communications.

1) Principles and Q-Learning: In RL, the agent observes at each time step, the environment's state s_t and chooses an action a_t. This action modifies the environment, which then proceeds to the next state s_{t+1}. Then, the agent receives a reward r_t according to the quality of its choice. The learning aim of the agent is to maximize the cumulative value of future rewards as illustrated in Fig 1.

One of the most popular reinforcement learning methods is Q-Learning, which chooses its actions by means of Q-values. Q-Learning uses a table to store all Q-values of all possible $\{$state, action$\}$ pairs. This Q-table is updated using the Bellman equation (eq. 1) and the action selection is usually done with an ε-greedy policy.

$$Q(s,a) = Q(s,a) + \alpha[y - Q_(s,a)] \tag{1}$$

where y denotes the temporal difference target:

$$y = r(s,a) + \gamma \max_{a'} Q(s',a') \tag{2}$$

with:

- $r(s,a)$, the reward of action a in the state s
- γ, the discount factor in [0,1]
- α, the learning rate
- $\max_{a'} Q(s',a')$, the optimal possible Q-value for state s'

Although effective, tabular Q-learning cannot be used for high dimensional state space problems. The Deep Q Network (DQN) method, which replaces the Q-table with a deep neural network (DNN), has been introduced [25] to overcome this problem.

2) Deep Q-Network: A Deep Q-Network (DQN) architecture uses a DNN of parameters θ as a function approximator to estimate Q-values as shown in Fig. 1.

The network is trained by minimizing a loss function sequence which evolves at each iteration i :

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[(y_i - Q(s',a';\theta_i))^2\right] \tag{3}$$

with :

$$y_i = r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) \tag{4}$$

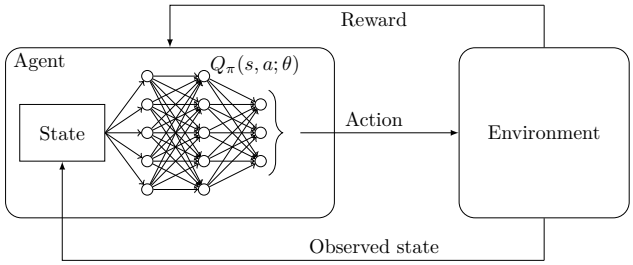

Fig. 1. Reinforcement learning diagram with DQN: The agent observes the state of the environment, takes an action and then receives a reward.

However, using a DNN leads to instability of the algorithm [25] requiring the use of another network of parameters θ^- to calculate the target (eq. 5). In addition, in order to break the correlation between consecutive samples, it is necessary to resort to a experience replay mechanism [26] that can be prioritized [27].

$$y_i = r + \gamma \max_{a'} Q(s',a';\theta^-) \tag{5}$$

But, instability is not the only challenge facing the DQN.

3) Double Deep Q-Network and Dueling Architecture: Both Q-learning and DQN suffer from a problem of overestimation of action values [28]. Double DQN (DDQN) solves this issue by using the target network when evaluating [29] leading to :

$$y_i = r + \gamma Q(s', \arg\max_{a'} Q(s',a';\theta_i);\theta^-) \tag{6}$$

In 2016, Wang et al. [30] have proposed a dueling architecture, called Dueling Double Deep Q-Network (3DQN), which explicitly separates the representation of state values and (state-dependent) action advantages as shown in Fig.2. The combination of streams is made by a special aggregating layer which produces the estimate of state-action value function via the Eq. 7. Compared to DDQN, this architecture learns the state-value function more efficiently and is less vulnerable to sudden policy switch.

$$\begin{aligned} Q(s,a;\theta,\alpha,\beta) \; &= V(s;\theta,\beta) + \\ &\left(A(s,a;\theta,\alpha) - \frac{1}{\mathcal{A}}\sum_{a'} A(s,a';\theta,\alpha)\right) \end{aligned} \tag{7}$$

Although capable of dealing with high dimensional state space problems, these methods suffer from the same limitation as Q-Learning for high dimensional action spaces.

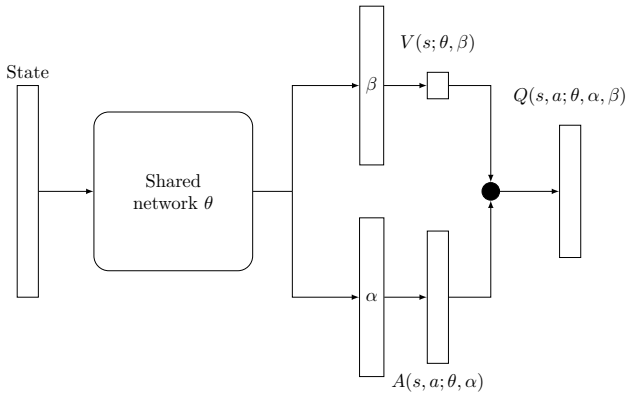

Fig. 2. Dueling Double Deep Q-Network has a common network from which two streams arise in order to separately estimate the state value $V(s)$ and the advantages for each action $A(s,a)$. Their aggregation is carried out by a special layer which produces Q(s, a) values for each action.

4) High dimensional discrete action spaces: The number of actions, that must be explicitly represented, grows exponentially with action dimensionality. Indeed, for an environment with an N-dimensional action space and n_d discrete sub-actions for each dimension d, $\prod_{d=1}^{N} n_d$ actions must be represented [9]. Several solutions are proposed to solve this problem from autoregressive network [31] to cooperative multi-agents [21]. In this work, we use the recently developed BDQ architecture introduced in [9], which is a variant of the dueling architecture. This solution divides the advantage part into several sub-branches (one per dimension) while sharing the state value part across the sub-branches. This approach achieves a linear increase of the number of network outputs, does not require prior information nor raise convergence issues. An example of this architecture can be found in Fig.6.

III. Method definition

A. Mission presentation

A MRS composed of homogeneous USVs and organized as a robotic cluster must conduct a SAR mission in an area. It must complete every mission task as fast as possible before any USV runs out power. To reach this goal, the MRS must share its resources by distributing the tasks over the cluster.
In a first approach, we focus on tasks that can by distributed, namely we only consider complex computational tasks that can benefit from robotic clustering. But USVs also run local tasks, such as navigation control and obstacle avoidance, that impact CPU/GPU load and so available computing resources. Therefore, the computing resources are split into local and distributed parts and the task mapping method only considers the second one.

Since our problem crosses multiple complex research fields such as distributed High Performance Computing (HPC), SAR and power optimization, we make some assumption to focus on our specific question.

1) Area Definition: An area represents a zone of the mission environment and determines the conditions the MRS faces. Each area is defined by several features with values between 0 and 1 representing the environment state as well as the mission progress as described in Table I. Some parameters are fixed and characterize the area while the others evolve according to the tasks completed by the USVs.

TABLE I
AREA CHARACTERISTICS

Types	Not.	Descriptions	Fixed
Hazard	H	Represents the environmental conditions and directly influences tasks efficiency	Yes
ROI density	D	Illustrates the region of interest (ROI) and influences tasks efficiency	Yes
Exploration	E	Reflects the current exploration level	No
Search	S	The current search level which cannot exceed the exploration level	No
Rescue	R	Shows the current rescue progress in this area which cannot exceed the search level. The mission end when this value reach 1.	No
Path planning	P	Represents the necessity to re-evaluate path planning after several USVs movements. Its speed evolution depend on a factor λ_P	No
Merging	M	Depicts the data to be merged due to tasks completion. The merging rate depend on a factor λ_M	No

The need for path planning (Np) increases each time a robot starts a new job requiring motion. The value is updated at the end of the cycle according to Eq. 8.

$$Np_t = Np_{t-1} + \lambda_P \sum_{i \in \omega_t} j_i \tag{8}$$

with: ω_t the new jobs starting at cycle t requiring movement.

The need for merging (Nm) increases each time a mission task is completed according to Eq. 9.

$$Nm_t = Nm_{t-1} + \lambda_M \sum_{i \in \Gamma_t} T_i \tag{9}$$

with Γ_t the set of completed mission tasks at cycle t.

Areas are independent of each other and the MRS is deployed in one area at a time.

2) MRS: The multi-robot system involves n homogeneous USVs deployed simultaneously in the area. They can communicate with each other, but they evolve autonomously without exchanging information about their states. To limit the bandwidth use, communications only take place during auctions or when distributing tasks. Each USV is defined by the following characteristics:

TABLE II
SUMMARY OF USV CHARACTERISTICS

Characteristics	Description	Type
ID	The USV ID	Integer
Processing unit	The current use rate of the USV processing (CPU + GPU)	Float
Memory	The current use rate of the USV memory	Float
Battery	The available USV battery	Float
Specific path	Assert if the USV is currently following a specific path	Boolean
Speed	The current USV speed	Float
Server jobs	The list of server jobs currently performed by the USV	List
Worker jobs	The list of worker jobs currently performed by the USV	List

One of the USVs performs the function of auctioneer which seems to lead to the creation of a single point of failure (SPoF). In our first approach not using reinforcement learning, the problem does not arise because the auctioneer does not carry a specific capacity so any USV can handle this function and it is quite possible to change auctioneers between auctions. For the others approaches, this is not as straightforward since the auctioneer keeps a periodically updated neural network. A simple and inexpensive solution to alleviate the SPoF risk is to appoint a backup USV that can take over in case of the auctioneer failure. The backup USV keeps a copy of the network which is periodically transmitted by the first one.

3) Time definition: Our use cases being simulated, the time is discretized as a succession of iterations t_i with a fixed Δ of continuous time. It means that updates of tasks, robots, area states are performed at the end of each iteration in this order. Moreover it is common that a set of tasks can start in each iteration.

Minimizing the number of iterations required to complete the mission is one of the two objectives of the MRS that must manage to efficiently perform a serie of tasks.

B. Tasks

1) Task definition: The mission progresses with the completions of tasks and the nature of this evolution depends on the type of task. Our approach considers five types split into two categories: mission tasks and regular tasks. The mission tasks include exploration, search and rescue types which are directly necessary for mission completion. There is a chain of dependencies between these tasks. It is necessary to explore before searching and to research before rescuing. The regular tasks consist of path planning and merger types that must be performed regularly to limit penalties. The difficulty lies in maximizing the effectiveness of certain task types at the right time. Each task is divided into jobs to allow its parallelization. The task type affects the job characteristics which are defined in Table IV. Regardless of their type, tasks can (should) be distributed to improve the system efficiency.

2) Task distribution and generation: Task parallelization (distribution), obtained by allocating worker jobs, is the primary mechanism to increase tasks efficiency. All distributions are carried out according to a Master-Slave paradigm described by Camargo-Forero et al. [35] and require at least one server job and worker jobs. Processed Data can either be stored at worker or server level as illustrated in Fig. 3.

The target distributed system is such that task generation and task distribution are independent, the first one is driven by mission requirements and the second one by the optimization of computing tasks. Indeed, we expect USVs of a decentralized system to be as autonomous as possible to limit communication and speed up the decision process. Periodically, each USV attempts to start a new task based on its local understanding of mission progress. The start of a task depends on a probability that varies with the relevance (possible efficiency) of the task type at the present time according to the current area values. These probabilities are calculated at each time step and are given in Table III with E, S, R, P, M being the current values

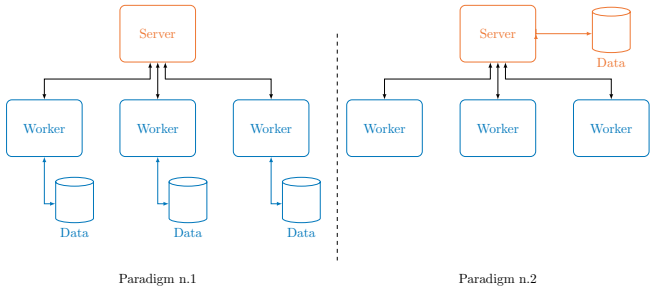

Fig. 3. Worker/Server paradigm: Communications only between a worker and the server. Processed Data can be stored at worker (1) or server level (2).

of exploration, search, rescue, path planning need and merging need respectively. There is also a probability ($p_{\text{None}} = 0.1$) of not starting a task illustrating the need to wait for a more convenient time.

3) Task effect: Each task completion affects the corresponding area characteristic. In the case of a task directly necessary for the mission (exploration, search, rescue), its completion increases the value of the corresponding attribute. Dependencies between types must be respected otherwise the task will have a limited impact or even be counter productive. On the other hand, in the case of a path planning or merger type task, the task completion reduces the value of the associated need and therefore the penalty incurred. The magnitude effect of completing a task depends on the number of workers assigned to it, the area states at the beginning and the end of the task. We formalize the completion effects of tasks with the realistic models presented in Table III. Other models can be chosen without loss of generality for the task management method. In this table t_n is the new value, t_c the current value, t_s the value when the task starts, w the number of worker jobs participating in the task, μ_1 the gain for necessary tasks, μ_2 the gain for the other tasks and $\mathcal{P}$ the penalty defined as follows:

$$\mathcal{P} = (1 - Np^2) + (1 - Nm^2) \tag{10}$$

In addition to the computing power, a task may require a movement of the USV that increases its energy consumption.

C. Energy consumption model

We consider the two sources of energy consumption. The first one is related to the embedded system including CPU/GPU computing resources, sensors and WiFi. It increases with the computing load and task distribution according to energy-proportional computing techniques such as dynamic voltage and frequency scaling (DVFS). The second one is the engine and increases exponentially with the USV speed.

1) Embedded system: We distinguish the processing and the communication parts. The computing resource model is extremely complex for new heterogeneous system-on-chip (SoC) since it must include partially correlated parts (multi-core CPU, GPU, RAM, etc.). In addition, the increased load of these components has several implications such as increasing the operating frequency and the activation of more cores. Our study required a realistic high-level model compliant with simulation time. So we built our model according to study [15]

TABLE III
TASK MODEL: EFFECT AND PROBABILITY OF EACH TASK TYPE ON THE MISSION

Type	Effect	Probability	Category	Example
Exploration	$E_{t_n} = \text{Min}\left(\frac{E_{t_c} + \mu_1 \times w \times \mathcal{P}}{H},\ 1\right)$	$p(E) = \begin{cases} 1 - E \\ 0 \text{ if specific path} \end{cases}$	Mission	Maps the area using a LIDAR to identify critical section [6].
Search	$S_{t_n} = \text{Min}\left(\frac{S_{t_c} + \mu_1 \times w \times \mathcal{P}}{D},\ E_{t_s}\right)$	$p(S) = E - S$	Mission	Examines critical section using computer vision to finds targets [13].
Rescue	$R_{t_n} = \text{Min}\left(\frac{R_{t_c} + \mu_1 \times w \times \mathcal{P}}{H + D},\ S_{t_s}\right)$	$p(R) = \begin{cases} S - R \\ 0 \text{ if specific path} \end{cases}$	Mission	Circle around the target, diagnose and calculate the required actions [32]
Path planning	$\text{Min}\left(\text{Max}(Np - \mu_2 \times w,\ 0),\ 1\right)$	$p(Np)\text{Max}(0, Np)$	Regular	Reassess USVs movements to avoid collisions and enhance efficiency [33].
Merger	$\text{Min}\left(\text{Max}(Nm - \mu_2 \times w,\ 0),\ 1\right)$	$p(Nm) = \text{Max}(0, Nm)$	Regular	Fusion new data to update the mission and prevent research overlapping [34].

TABLE IV
JOBS MAIN CHARACTERISTICS

Characteristics	Descriptions	Types
Processing	The job processing requirement	Server Worker
Memory	The job memory requirement	Server Worker
Execution time	The time during which the job must be executed	Server Worker
Specific path	Indicates if the job requires that the USV follows a specific path	Server Worker
Number	Defines the maximum workers number for the task	Worker

that provides regression models based on real measures with different mobile SoC. We made the following assumptions on power management policy to get tractable models:

1) Clock frequency and number of cores: an additional core is activated each time the clock frequency reaches its maximum and the frequency then returns to its minimum.
2) CPU and GPU use: we consider a global model where both are correlated, so ρ is the CPU and GPU use rate.
3) We introduce a multiplier scaling factor to models of [15] to adjust them to observations made on a Jetson TX2.

The final model is given by the following equation (mW):

$$C_C(\rho) = 0.68\rho^2 + 42.62\rho + 1485.93 \tag{11}$$

2) Engine model: Our engine consumption model corresponds to the consumption of the USV 'Heron' developed by the Clearpath company and presented in Fig. 4. The information provided by the manufacturer [16] allowed us to model the USV consumption according to its speed. We assume the speed is constant in an operating mode that depends on each task.

Fig. 4. Heron USV from the company Clearpath [16]

After including the Wifi power consumption, that we assume linear with the number of jobs ω according to [15], we obtain the following global power consumption model where ρ, ω and ν are the computational load, the number of jobs and the USV speed respectively (mW).

$$C_T(\rho, \omega, \nu) = 0.68\rho^2 + 42.62\rho + 1000\omega + 3485.93 + 22.918\text{e}^{1.127\nu} \tag{12}$$

IV. SOLUTION DEFINITION

We propose two kinds of approaches to solve this problem, both rely on auctions to perform the job allocation.

A. Auctions

1) Auction mechanism: Whenever a USV wants to start a new task, it starts a server-type job and sends job worker requests to the auctioneer as shown in Fig.5. Each request takes the form of a bid where the price means the computational need of the task. An offer is published for each possible worker job. At the same time, each USV transmits its ability to accept a worker-type job in the form of a bid whose valuation corresponds to its available computational resources. Since a USV can execute several jobs simultaneously, it sends several bids with decreasing values in order to ensure that it is still able to execute the job. In addition to a valuation linked to computing resources, each bid has a second valuation representing the state of its owner's battery and the job types concerned. It is then up to the auctioneer to match supply and demand. Conventionally, double auctions take place in a non-cooperative environment where the auctioneer role is to set a sale price that satisfies both buyers and sellers. In our case, the environment is totally cooperative and the price only represents computing capacities/requirement. A simplistic approach therefore consists of sorting supply and demand to match a maximum of bids. Of course, this solution is not without consequence since it monopolizes computing and energy resources which could be spent on more efficient/relevant tasks. On the other hand, if the system is not using its full capacity, it is wasting time. The auctioneer must therefore strike balance between task efficiency and maximization of the computing resources use.

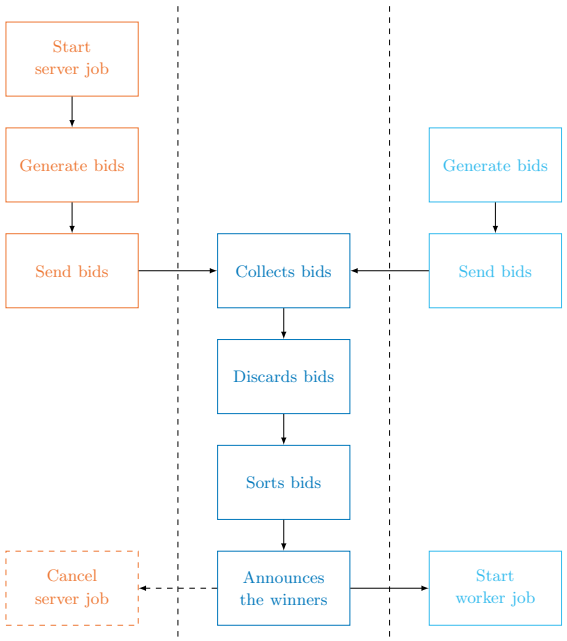

Fig. 5. Auction principle: each USV starting a new task (orange) generates all needed worker-type offers and sends them to the auctioneer (blue). Simultaneously, all USVs (cyan) communicate their computing-capacity offers to the auctioneer that conducts the auctions and communicates the results to the MRS. Finally, tasks without enough workers are canceled (dashed).

2) Auction objectives: The use of auctions and the market serves four purposes. First, auctions allow an efficient distribution of worker jobs among USVs without requiring explicit coordination from the server or workers. Second, by making possible to classify offers by price, least loaded USV become favored, which contributes to computational load balancing within the system. Third, the USVs communicate their current battery state, so their energy levels can be taken into consideration to ensure whole system energy balance. Last but not least, the double auction system allows the auctioneer to have a global vision of the new tasks allowing disqualifying, temporally preventing or promoting certain task types. In summary, the decentralized system manages to exercise some control over the autonomous task selection mechanism. Of course, this control comes with limitations because the system cannot force the start of a task type and disqualification induces a waste of time.

However, determining the correct bid pairing requires the system to be able to understand the relationship between the task type, the task efficiency and the dynamic mission state.

B. Deep reinforcement learning auctions

We expect to learn how to define at each auction how many tasks of each type must be kept and which minimum energy level a USV must have to participate in the auction. However, the problem complexity leads to the existence of a high dimension action space. Indeed, there are five types of tasks and two auction listings which add up to seven dimensions. Since each dimension has up to nine levels, a total of $\prod_{d=1}^{N} n_d = 1\ 166\ 886$ possible actions need to be considered which leads us to use a BDQ architecture.

1) Branching approach: Our BDQ architecture described in Fig.6 uses a common network from which a branch emerges for each action dimension as well as one to estimate the state value function. For each action dimension the corresponding action value is obtained by combining state value and advantage values through a special aggregation layer. Similary to [9], we test several aggregation formulas and obtain the best performances by locally subtracting the average advantage of each branch from its sub-action advantages, before summing them with the state value as follows:

$$Q_d(s, a_d) = V(s) + \left(A(s, a_d) - \frac{1}{n} \sum_{a'_d \in \mathcal{A}_d} A_d(s, a'_d) \right) \quad (13)$$

with:

- d and action dimension and $d \in \{1, ..., N\}$
- $|\mathcal{A}_d| = n$ discrete sub-actions
- The state s
- A sub-action $a_d \in \mathcal{A}_d$

The TD target for the BDQ updates can be calculated in different ways. We tested the three propositions of [9] and like the authors, we get better performance by averaging rather than maximizing with the following equation :

$$y = r + \frac{1}{N} \sum_{d} \gamma Q_d^-(s', \arg\max_{a'_d \in \mathcal{A}_d} Q_d(s', a'_d)) \quad (14)$$

Although showing promising results, the BDQ approach remain under explored. In this study we will assess its performances against another method that can handle high dimensional discrete action spaces, the multi-agent solution.

2) Multi-agent approach: It is considered to overcome the high dimensional action space problem by dividing the learning agent into independent sub-agents dealing with sub-dimensions of the problem. The combination of the resulting sub-actions constitutes the agent's action. Unlike the BDQ approach branches, sub-agents do not share a common state value and therefore must be coordinated using a team reward.

a) Standard architecture: The agent is divided into as many sub-agents as the action space of our problem has sub-dimensions. It results in seven independent 3DQNs (one per sub-agent) as illustrated in the Fig. 7. However, the multi-agent approach has only been successfully tested for two sub-agents and convergence problems are expected with the increase in the number of sub-agents [36].

b) Tuned architecture: In order to overcome the possible convergence issue, we propose a second approach with a reduced number of sub-agents. To achieve this, the sub-dimensions are grouped by type. Thus, a sub-agent deals with the actions relating to the mission tasks, a second with the regular tasks and the last agent handles the management of the energy. This results in an architecture with three sub-agents as illustrated in Fig. 8. This clustering provides a better understanding to the sub-agent by making them directly responsible for part of the problem. In addition, we were able to disqualify some irrelevant actions sub-combinations. For example simultaneously trying to maximize the exploration, search and rescue tasks does not make sense since the MRS does not have sufficient resources.

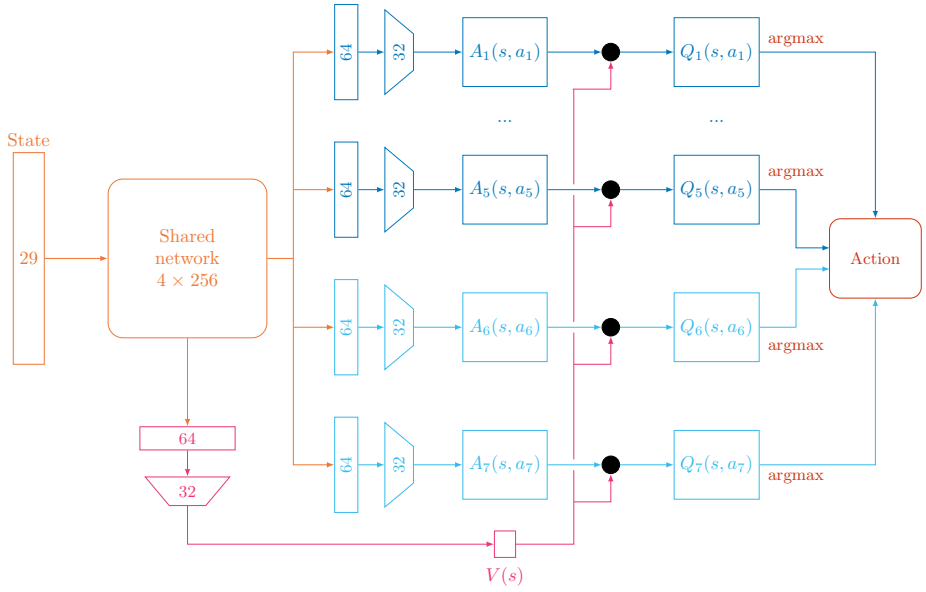

Fig. 6. BDQ architecture: A common network leads to eight branches where seven correspond to the action dimensions and the last one calculates the state value to be aggregated to each other branches to obtain the action values for each dimension. The combination of all sub-actions defines the global action.

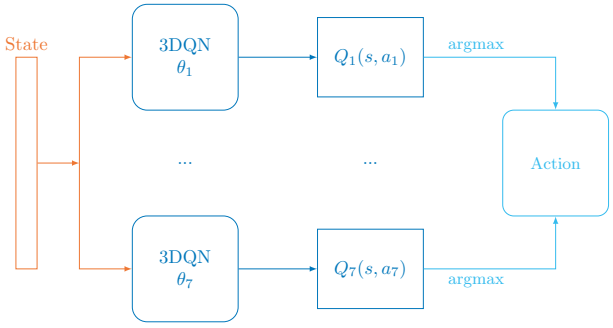

Fig. 7. Standard architecture : The learning agent is made up of seven independent 3DQNs each dealing with a problem sub-dimension. The combination of their choices forms the action taken by the agent.

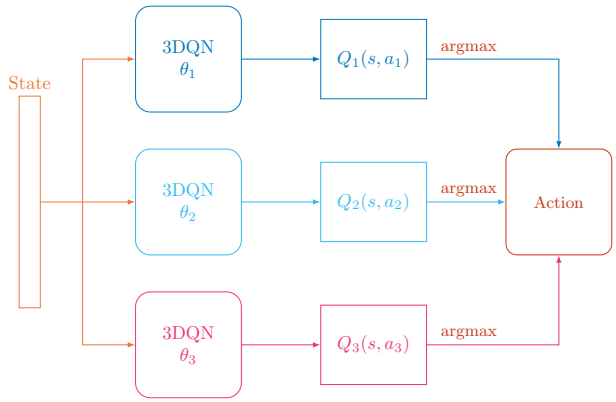

Fig. 8. Optimized architecture: The learning agent is composed of three independent 3DQNs each dealing with a clustering of problem sub-dimension. Each sub-agent produces a sub-combination of actions, the combination of which forms the action taken by the agent.

V. Experimental Setup

We have developed a simulator to compare the different approaches, efficiently explore most of the parameters and evaluate the relevancy of our solution before a future real deployment.

A. Experimental Set Up

Our simulator is coded in python 3.8.5 and the learning and deployment parts of our RL methods rely on the widely used Tensorflow (with Keras), open source ML framework. Our simulator can handle a wide variety of missions depending mainly on the parameters used to define the area, the MRS and the tasks. In this study we had to reduce the number of parameters explored, and therefore, fix some of them in order to highlight the relevant key points.

To assess our method viability in real conditions, we tested the inference and learning times on an embedded architecture adapted to mobile robots. The Jetson TX2 is a power-efficient ($< 15W$) computing device suitable for Embedded AI. Since it is very popular on mobile robot, it will serve as a reference. As shown in Table V, the results of our BDQ approach (other solutions provide comparable times) are fully compliant with the computing power available on the target small USV.

TABLE V
INFERENCE AND LEARNING TIME BY ARCHITECTURE :

Configuration	Emulator	Nvidia Jetson TX2
Components	Intel Xeon Silver 4114 (x2) Nvidia GTX 1080 TI	ARM Cortex-A57 Nvidia Denver 2 GPU Pascal
Hardware	20 cores at 2.2 GHz 64 GB of DDR4 3584 cores CUDA 11 GB of GDDR5X	4 cores at 2 GHz+ 2 cores at 2Ghz 8 GB of LPDDR4 256 cores CUDA
Inference (BDQ/MA-T)	3.9/5.3 ms	10.4/14.5 ms
Experience replay (batch of 32)	517/728 ms	2918/3947 ms

B. Settings

The main fixed parameters are listed in Table VI. In this first approach, the system is homogeneous and therefore all USVs have the same capacities.

TABLE VI
ENVIRONMENT SETTINGS : LIST OF FIXED PARAMETERS.

Parameters	Values
MRS size	20
Battery power capacity	325 W.h
Iteration duration	3 min
Maximum workers number	5
Number of iterations required to complete a task	4
Number of iterations between auctions	2
μ_1	0.01
μ_2	0.025
λ_P	1
λ_M	1

Table VII shows the main learning parameters used by our RL solutions. They were determined by trial and error. For all the results presented, the RL approaches were previously trained on 40 missions and no learning is carried out during the evaluation.

TABLE VII
LEARNING SETTINGS

Types	Values	
	BDQ	Multi-agent
Learning rate : α	10^{-3}	10^{-3}
Discount factor: γ	0.95	0.90
Target network update frequency : c	40 iterations	40 iterations
Memory size	256	256
Batch size	32	32

C. Task set

Table VIII shows the different job computational prerequisites depending on the task type and the set used. The use of multiple sets allow us to observe the solution efficiency when facing imbalance. Indeed, if on average each set requires as much CPU as memory, sets 2 and 3 introduce phases where a resource becomes more valuable (e.g. at the beginning, when exploration is necessary, memory become the most requested resource). In addition, regardless of the set used, a server job requires 10 % of CPU and memory.

TABLE VIII
TASK SET : PREREQUISITES FOR EACH JOB

Types	Set n.1		Set n.2		Set n.3	
	CPU	RAM	CPU	RAM	CPU	RAM
Exploration	0.3	0.3	0.2	0.4	0.1	0.5
Search	0.3	0.3	0.4	0.2	0.5	0.1
Rescue	0.3	0.3	0.3	0.3	0.3	0.3
Path planning	0.3	0.3	0.35	0.25	0.4	0.2
Data synchronization	0.3	0.3	0.25	0.35	0.2	0.4

VI. RESULTS

The results are the average obtained over 1000 missions. The terms ‘BDQ’, ‘MA-T’, ‘MA-S’ and ‘Market’ denote the following approaches: the BDQ architecture, the tuned multi-agent (3 sub-agents), the standard multi-agent (7 sub-agents) and the static market. We adopt a question-based method to analyze the results. Before comparing these two solutions, we want to ensure the relevance of the use of a robotic cluster compared to a local solution.

☐ Does the distribution of tasks within the robotic cluster increase the mission success rate?

A. Local vs distributed

For comparison purpose we first introduce a local solution that obviously cannot parallelize its tasks within the cluster. However, this solution can parallelize tasks locally without requiring any server-type job and without generating any additional energy consumption from communications. We observe that the local solution is unable to successfully complete a single mission. Distribution is therefore necessary.

☐ Does task distribution allow a faster mission completion?

We increase the power capacity of the local solution (3.25 kW.h) by a factor of 10 to be able to compare the completion speeds. Figure 9 shows the completion speeds of the different solutions according to the task set used. The local solution presents a completion speed close to the distributed approaches for set 1, but it turns out to be totally ineffective for the other two sets highlighting the relevance of distribution. In addition, we observe a greater disparity (45 % more iterations needed for the slowest mission) suggesting a poor reliability. The lack of results for ‘MA-S’ with the sets 2 and 3 is addressed later.

The distribution advantage is clear and the local approach will not be considered henceforth. We then observe that the market solution completion times outperform BDQ for all sets.

☐ Does a solution ability to quickly complete a mission induces a high success rate ?

B. Success rate

Figure 10 displays the failure rates of the four solutions depending on the task set used. The results show that ‘BDQ’ and ‘MA-T’ offer much better performance with a failure rate up to more than 30 times lower for set 1. Although offering better results than the static approach, the solution ‘MA-S’ still fails more than 50 % of these missions. When an unbalanced set is used, the mission appears more difficult to complete and the performance gap between the solutions is reduced. The solution ‘MA-S’ is unable to complete a single mission suggesting, as expected, some convergence issue. Nevertheless, the other RL approaches remain much more effective (31.8 % and 19. % failures against 77.8 %). It is worth mentioning that the ’MA-T’ approach outperforms the ‘BDQ’ one for the set n.3.

Figure 11 shows the failure rates of the ‘BDQ’ and ‘Market’ solutions based on the initial power capacity with task set 1. The results confirm the dynamic advantage conferred by the learning capacity. Indeed, this approach makes the system

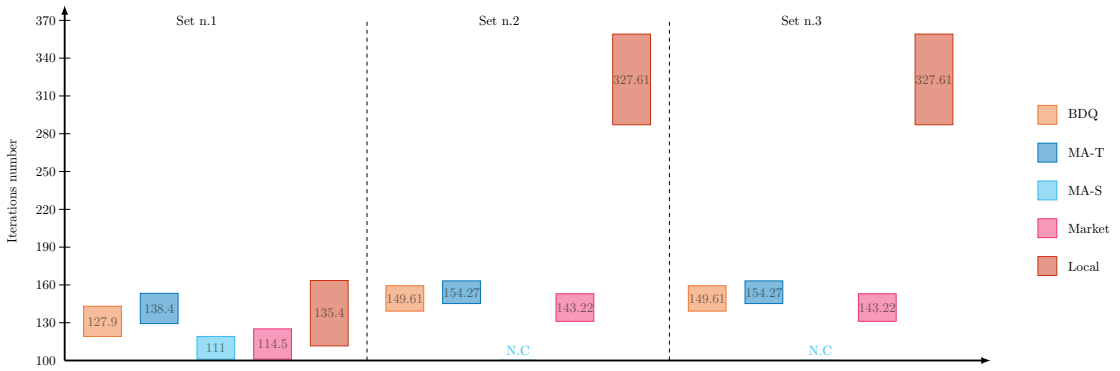

Fig. 9. Average completion speeds vs task set: rectangles indicate maximum and minimum values while numbers are the average values.

TABLE IX
NUMBER OF TASKS EXECUTED PER TYPE (OPTIMAL SOLUTION: 40 FOR MISSION TASKS)

Type	Set n.1				Set n.2				Set n.3			
	BDQ	**MA-T**	**MA-S**	**Market**	**BDQ**	**MA-T**	**MA-S**	**Market**	**BDQ**	**MA-T**	**MA-S**	**Market**
Exploration	50	45.9	50.4	49.1	49	47.6	N.C	51.8	50	51.8	N.C	50.2
Search	53.7	52.4	55.1	53.8	49.6	55	N.C	55.6	59.9	60	N.C	54.2
Rescue	49.6	48.6	48.8	48.8	50	50.7	N.C	50.9	50.3	46.2	N.C	49
Path planning	66.1	54	62.3	72	55.3	56	N.C	71.7	56.4	55.5	N.C	70.4
Merging	79.6	102	71.3	108.5	69	70	N.C	89.6	71.2	72.1	N.C	91

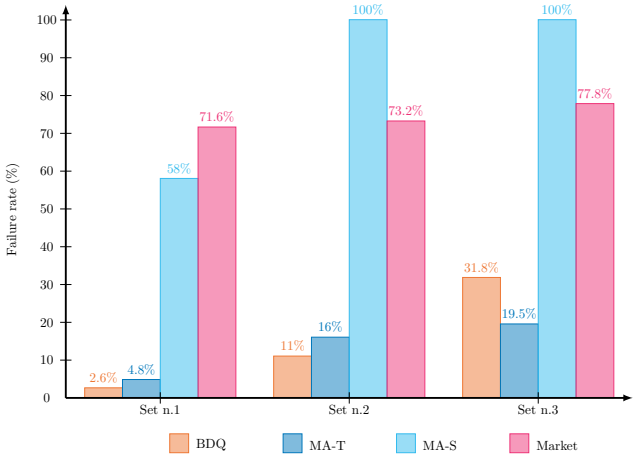

Fig. 10. Failure rate vs task set

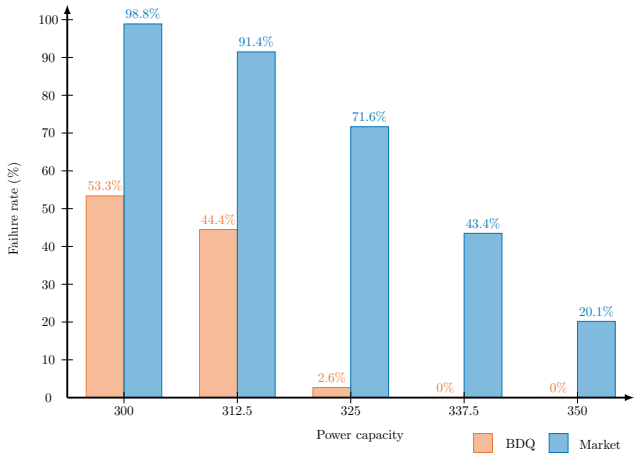

Fig. 11. Failure rate vs initial battery level for task set 1

more reliable by displaying a failure rate of 0 % for an initial battery power capacity greater than 325 W.h. Furthermore, the USVs from the BDQ solution with 25 W.h less initial power are as efficient as from the classical market approach.

☐ Does a high success rate mean maximized resource use?

C. Computing load

Figure 12 shows the average load rates (CPU and memory) of the MRS during a mission. The results of the 'MA-S' solution are missing for sets n.2 and n.3 because it failed to complete them. We note the following three points. First, the RL solution uses less resources than the market-based with the 'MA-T' being the most parsimonious. Second, the use of unbalanced sets restricts the systems ability to mobilize their resources which explains the reduction in the overall performance.

☐ Does the observed high resource usage by the market approach reflects poor task efficiency?

D. Task efficiency

Table IX shows the number of tasks executed on average. The mission task efficiency is optimal when it is performed with five workers, without penalty and respecting the dependency chain. In this case, only 40 tasks are needed in each category to complete the mission. With a number of executed tasks ranging from 45.9 to 60, the solutions show an efficiency rate varying from 87.1 % to 66.7 % for an average of 78 %. The static market approach does not suffer for mobilizing

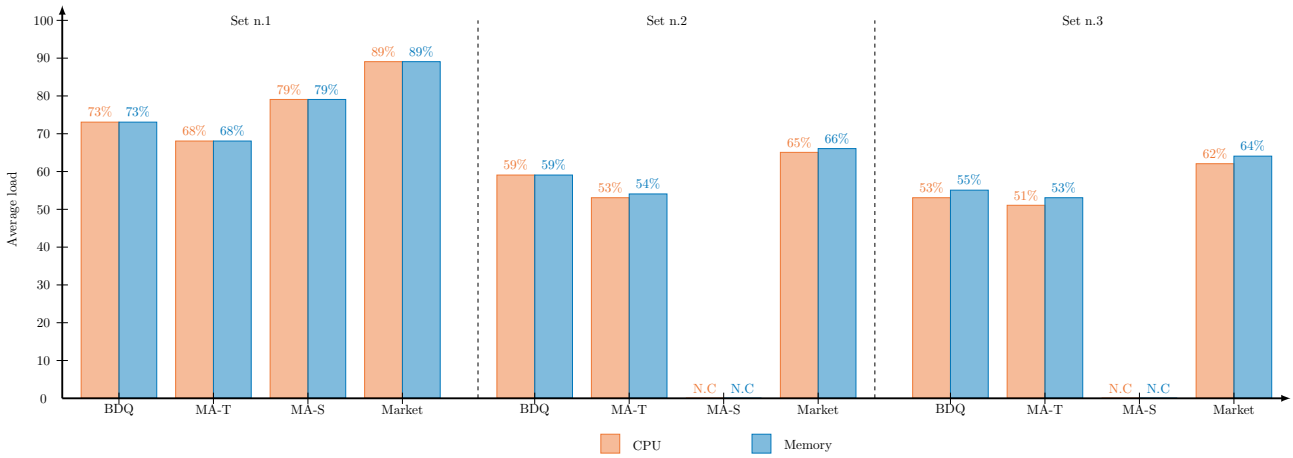

Fig. 12. Resource usage of each approach according to the task set used.

more resources. Indeed, this approach offers similar efficiency rate (average of 77.7 %), but perform more regular tasks explaining the use of more resources. However, the regular tasks do not require movement and therefore have little effect on the system energy consumption. Although showing similar performance, the 'BDQ' and 'MA-T' solutions focus on different regular tasks. Indeed, the 'BDQ' approach performs 22,4 % more path planning tasks and the 22 % fewer merging tasks indicating the use of different strategies. Things change when using unbalanced sets and these two approaches show a similar execution number for regular tasks. In addition to constraining the resources mobilization, the difficulty induced by these sets reduces the number of viable strategies leading to this similarity. Finally, the 'MA-S' solution displays the best efficiency rate for its mission tasks while performing the fewest regular tasks. This efficient and rational behavior does not reflect a lack of convergence. In this context, how can the observed performance differences be explained ?

- ☐ Does the high failure rate of the 'Market' approach and the 'MA-S' approach inability in completing the missions for the sets n.2 and n.3 come from poor distribution of energy demanding tasks?

E. Energy management

We can analyze the distribution of energy-consuming tasks by observing the remaining power capacity of USVs' batteries. Figure 13 shows the battery levels (min, max and avg) at the end of a mission depending on the solution and the task set used. On the one hand, we clearly see the failure causes of the 'MA-S' solution, which prove unable to distribute the energy-consuming tasks. On the other hand, the 'Market' solution does not present any fault in its distribution. Indeed, if the 'Market' levels prove to be slightly lower, its allocation mechanism seems efficient. Indeed, the disparities displayed are low and similar to those of the 'BDQ' and 'MA-T' approaches attesting to a good distribution.

The reason why the market approach fails to deliver good results is not straightforward. Indeed, its behavior seems relevant since tasks are perform efficiently while distributing the energy intensive ones. Admittedly, it performs more regular tasks, but they do not involve any specific movement and therefore the consumption is low. In addition, these executions are not carried out to the detriment of mission tasks since their observed efficiency indicates good distribution. Finally, this solution makes multiple small inaccuracies that only a dynamic and adaptive approach can detect and avoid. On a final note, it would be tempting to estimate that these differences in results arise from the arbitrary choice of starting battery capacity. However, as shown in Fig. 11, the BDQ solution manages to provide better results with 25 W.h less power capacity (starting at 300 W.h) ruling out this hypothesis.

VII. Conclusion

In this study, we investigated the use of reinforcement learning as an overlay to a market-based approach for the management of a robotic cluster. The case study is a search and rescue context. Managing a robotic cluster is complex since any use of resources impacts the other cluster members. In addition, the amount of available resources depends on the load of each member and therefore inevitably fluctuates during the mission. The high dynamic degree of the problem is simulated with the following four points:

1) A dependency chain between mission tasks conditioning the relevance of their execution.
2) A dynamic penalty that evolves according to the actions taken by the system.
3) A task selection mechanism, based on dynamic probabilities about drone local understanding of the mission state, with no direct control from the system.
4) A strong energy constraint drives the mission success.

The results obtained by simulation show the contribution of reinforcement learning to auction conduct. Indeed, learning considerably improves the performances by reducing the failure rate (from 69 % to 40 % according to the task set). It then becomes possible to use learning to make either the system more reliable or to use drones with lower battery capacity. As shown, the highly dynamic nature of this problem makes the

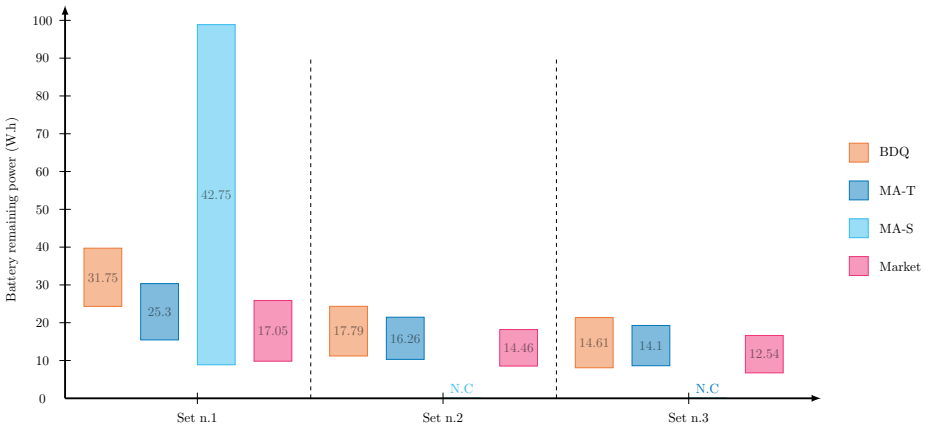

Fig. 13. Distribution of USVs energy after successful missions: rectangles indicate maximum and minimum values while numbers are the average values.

design of an efficient static solution very complex since many uncertain and evolving parameters must be considered.

However, the complexity of the problem leads to a curse dimensionality making the use of classical Deep Q-learning methods impossible. In order to overcome this limitation, we resorted to two little explored methods: the new BDQ architecture and the multi-agent division. These two solutions have comparable performance, but the multi-agent approach suffers from several limitations. Indeed, as expected, this approach encounters difficulties in converging with the increase in number of sub-agents . We solved this problem by clustering together sub-dimensions (having each sub-agent process several sub-dimensions). However, although efficient in our case, this method should face difficulties to converge with the increase in sub-dimension number or fall again into a curse of dimensionality with the increase in possible action per sub-dimension or/and larger clusterings.

Overall, the BDQ method appears more suitable since it is perfectly scalable (for both increase in sub-dimensions and in action number by sud-dimension), does not suffer from convergence issue (due to the common state value) and, unlike grouping, does not require specific problem knowledge.

The proposed architecture of the Deep Q-learning solution is a new BDQ type that scales with the dimensionality issue and makes possible the estimation of the Q-values for each sub-dimension.

Despite a growing need to locally distribute the computational load of MRS, the robotic cluster paradigm has been barely explored. Therefore we believe that our promising results can promote this idea. It will pave the way for future improvements based on the following future works.

First, an S&R type mission environment turns out to be highly complex and we had to simplify it to run the thousands of simulations that were required by our study. The next step is to consider spatial modeling making it possible to link the USVs and tasks to positions. Secondly, the approach must be tested with the deployment of a real USVs but prior a real deployment, a hardware-in-the-loop simulation should be performed with a dynamic 3D environment such as Gazebo.

REFERENCES

[1] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, Sep. 2004. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364904045564

[2] A. Torreño, E. Onaindia, A. Komenda, and M. Štolba, "Cooperative Multi-Agent Planning: A Survey," *ACM Computing Surveys*, vol. 50, no. 6, p. 32, Nov. 2017.

[3] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, pp. 309–320, 2019.

[4] M. Afrin, J. Jin, A. Rahman, A. Rahman, J. Wan, and E. Hossain, "Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 842–870, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9360855/

[5] A. Marjovi, S. Choobdar, and L. Marques, "Robotic clusters: Multi-robot systems as computer clusters," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1191–1204, Sep. 2012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889012000632

[6] B. D. Gouveia, D. Portugal, D. C. Silva, and L. Marques, "Computation Sharing in Distributed Robotic Systems: A Case Study on SLAM," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 2, pp. 410–422, Apr. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6913567/

[7] P. Gautier, J. Laurent, and J.-P. Diguet, "Comparison of Market-based and DQN methods for Multi-Robot processing Task Allocation (MRpTA)," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. Taichung, Taiwan: IEEE, Nov. 2020, pp. 336–343. [Online]. Available: https://ieeexplore.ieee.org/document/9287887/

[8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017. [Online]. Available: http://www.nature.com/articles/nature24270

[9] A. Tavakoli, F. Pardo, and P. Kormushev, "Action Branching Architectures for Deep Reinforcement Learning," *arXiv:1711.08946 [cs]*, Jan. 2019, arXiv: 1711.08946. [Online]. Available: http://arxiv.org/abs/1711.08946

[10] X. Jia and M. Q. Meng, "A survey and analysis of task allocation algorithms in multi-robot systems," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2013, pp. 2280–2285.

[11] S. Koenig, C. Tovey, M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, and J. Sonal, "The Power of Sequential Single-Item Auctions for Agent Coordination," in *Proc. of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*. Boston, Massachusetts, USA: AAAI Press, Jul. 2006, pp. 1625–1629. [Online]. Available: http: //www.aaai.org/Library/AAAI/2006/aaai06-266.php

[12] Z. Beck, L. Teacy, and A. Rogers, "Online Planning for Collaborative Search and Rescue by Heterogeneous Robot Teams," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, May 2016, pp. 1024–1033.

[13] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang, "UAV Autonomous Target Search Based on Deep Reinforcement Learning in Complex Disaster Scene," *IEEE Access*, vol. 7, pp. 117 227–117 245, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8787847/
[14] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "LSAR: Multi-UAV Collaboration for Search and Rescue Missions," *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8695011/
[15] C. Yoon, S. Lee, Y. Choi, R. Ha, and H. Cha, "Accurate power modeling of modern mobile application processors," *Journal of Systems Architecture*, vol. 81, pp. 17–31, Nov. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1383762117301947
[16] CLEARPATH, "Heron user manual," 2020. [Online]. Available: www.generationrobots.com/media/clearpath_heron_usermanual.pdf
[17] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *IEEE/RSJ Int. Conf.s on Intelligent Robots and Systems (IROS)*, 2018, pp. 4238–4245, ISSN: 2153-0866.
[18] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *IEEE 16th Int. Conf. on Automation Science and Engineering (CASE)*, 2020, pp. 1461–1466, ISSN: 2161-8089.
[19] P. Wenzel, T. Schön, L. Leal-Taixé, and D. Cremers, "Vision-based mobile robotics obstacle avoidance with deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 14 360–14 366, ISSN: 2577-087X.
[20] L. Jiang, H. Huang, and Z. Ding, "Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 1179–1189, 2020, conference Name: IEEE/CAA Journal of Automatica Sinica.
[21] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent Cooperation and Competition with Deep Reinforcement Learning," *arXiv:1511.08779 [cs, q-bio]*, Nov. 2015, arXiv: 1511.08779. [Online]. Available: http://arxiv.org/abs/1511.08779
[22] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020, conference Name: IEEE Transactions on Vehicular Technology.
[23] Y. Xiao, J. Hoffman, T. Xia, and C. Amato, "Learning multi-robot decentralized macro-action-based policies via a centralized q-net," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 695–10 701, ISSN: 2577-087X.
[24] B. Park, C. Kang, and J. Choi, "Cooperative multi-robot task allocation with reinforcement learning," *Applied Sciences*, vol. 12, no. 1, p. 272, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/12/1/272
[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602v1*, Dec. 2013, arXiv: 1312.5602v1. [Online]. Available: https://arxiv.org/abs/1312.5602v1
[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://www.nature.com/articles/nature14236
[27] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," *arXiv:1511.05952 [cs]*, Feb. 2016, arXiv: 1511.05952. [Online]. Available: http://arxiv.org/abs/1511.05952
[28] H. V. Hasselt, "Double Q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.
[29] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *arXiv:1509.06461 [cs]*, Dec. 2015, arXiv: 1509.06461. [Online]. Available: http://arxiv.org/abs/1509.06461
[30] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," *arXiv:1511.06581 [cs]*, Apr. 2016, arXiv: 1511.06581. [Online]. Available: http://arxiv.org/abs/1511.06581
[31] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson, "Discrete Sequential Prediction of Continuous Actions for Deep RL," *arXiv:1705.05035 [cs, stat]*, Jun. 2019, arXiv: 1705.05035. [Online]. Available: http://arxiv.org/abs/1705.05035
[32] C. Yang, D. Wang, Y. Zeng, Y. Yue, and P. Siritanawan, "Knowledge-based multimodal information fusion for role recognition and situation assessment by using mobile robot," *Information Fusion*, vol. 50, pp. 126–138, 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1566253517302038
[33] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Trans. on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.
[34] Y. Yue, P. Senarathne, C. Yang, J. Zhang, M. Wen, and D. Wang, "Probabilistic fusion framework for collaborative robots 3d mapping," in *21st Int. Conf. on Information Fusion (FUSION)*, 2018.
[35] L. Camargo-Forero, P. Royo, and X. Prats, "Towards high performance robotic computing," *Robotics and Autonomous Systems*, vol. 107, pp. 167–181, Sep. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S092188901830232X
[36] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, Feb. 2012.

Paul Gautier is an Assistant Professor at the University Bretagne Sud and works at the CNRS Lab-STICC. His research interests include robotic cluster, machine learning in robotics and reinforcement learning. He received the Ph.D. degree in computer science from the University Bretagne Sud, France, in 2021.

Johann Laurent is an Associate Professor at the University Bretagne Sud and works at the CNRS Lab-STICC (Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance). His research interests include software consumption estimation for embedded systems and IA tools for cyber physical systems. He received a Ph.D. in electronics from the University Bretagne Sud, France, in 2002 and his Habilitation à Diriger les Recherches in 2015.

Jean-Philippe Diguet is a CNRS director of research. He has been A/Prof. at UBS University, research visitor at IMEC/Belgium and UQ/Australia, invited Prof. at Tohoku Univ./Japan and USP/Brasil. At Lab-STICC he has led the team method and tools for SoC and embedded system (MOCS) from 2008 to 2016 and then the ICT and Drones program. Since 2021 his is the director of CROSSING, an International CNRS Lab in Adelaide, Australia dedicated to Human/Automous-Agents teaming. His research work focused initially on various aspects of SoC and embedded system design including self-adaptation and now addresses different levels of embedded and distributed intelligence.