Dedale : A Dedicated Testbed for Multi-Agents Problems

Cédric Herpson
Sorbonne Université, CNRS, LIP6,
F-75005 Paris, France
cedric.herpson@lip6.fr

***Abstract*—We present Dedale, an environment for studying multi-agents coordination, learning and decision-making problems under realistic hypotheses. Dedale avoids the 8 fallacies of MAS in which all previous testbed fall and offers open, dynamic, asynchronous and partially observable environments. Highly parametrizable, Dedale allows to tackle either cooperative or competitive exploration, patrolling, pickup and delivery, treasure(s) or agent(s) hunt problems with teams from one to dozens of heterogeneous agents in discrete or continuous environments. The variety of modelable multi-agents problems associated with the possibility to create a peer-to-peer network of Dedale's environments makes us believe Dedale beeing able to become a unifying environment for both MAS research and teaching communities in their goal to work and evaluate their proposals under real-life hypotheses. Feedback from more than 150 early-users comfort us in this perspective.**

***Index Terms*—Multi-agent, environment, simulation, testbed, p2p, decentralised problem solving, coordination, decision-making, learning**

Introduction

In recent years, single-agent standardised environments allowed the research community to efficiently benchmark their proposals, which led to real progress in the field [1, 5, 44]. Unfortunately, there is currently no equivalent for multi-agent coordination, learning and decision-making problems under real-life hypotheses.

Several experimental platforms have been built in the last 30 years [8, 9, 24, 25, 27, 43]. Nevertheless, they either focus on large-scale Complex Adaptive Systems (CAS) [22] and are thus restricted to synchronous environments with no [27] or few communication [41] capabilities or they assume closed-world environments where evolve up to only 4 homogeneous agents with a perfect vision of the system in zero-sum games between two teams [33], when they are not controlled from a unique omniscient entity [20].
In both cases, these platforms make unrealistic hypotheses. It thus makes their use difficult or even impossible for coordination problems, and may render the proposed solutions ineffective or inoperable in a real situation. As a result, researchers working on these issues often use their own (unpublished) toy examples environments, which can unfortunately turn out to be over-fitting the proposed algorithms or make the results difficult to reproduce.

Through the release of Dedale, we aim to facilitate and improve the experimental evaluation conditions of the developed algorithms, and to contribute to the progress of the field towards decentralised solutions able to deal with real-world hypotheses and problems.
Section 1 presents the state of the art of multi-agent testbeds and compare them to Dedale through the prism of what we define as the 8 fallacies of MAS. Section 2 presents Dedale, and more particularly the important features from the user's point of view. In Section 3, we present three class of multi-agent problems that can be studied through the use of Dedale as well as the associated open-research issues. Finally, section 4 presents the peer-to-peer capabilities of Dedale for long-term experiments in open-world before discussing its future evolutions.

I. Existing platforms and the 8 fallacies of MAS

To study decentralised multi-agents coordination, decision-making and learning problems under real-life hypotheses, a suitable testbed should avoid what we call the 8 fallacies of MAS:

1) Agents take turns executing each other
2) Agents are homogeneous and run at the same speed
3) Agents have access to unlimited resources
4) Agents are reliable
5) Agents are sure
6) Agents have a global and perfect vision of the system
7) Agents number does not change over time
8) Communication respects the 8 fallacies[1] of distributed systems [35].

To consider turns (1) make disappear all questions related to synchronized actions and drastically reduce conflicts over resources. (2,3) Homogeneous agents – both in term of capacities and (unlimited) computational capabilities – eliminates bottlenecks and operational constraints (calculation time, memory, and energy), and simplifies coordination problems (the agents are interchangeable). (4,5) In real life situations, whether deployed on servers or mobile robots, agents can experience malfunctions, face byzantine behaviours,.. Eliminating these hypotheses does not make it possible to study the robustness of the proposed solutions. (6,7) Outside of toy use-cases, open-environments and limited-percepts' agents are the norm. Assuming complete

[1]The network is secure, reliable, instantaneous, with infinite bandwidth, the topology is fixed and homogeneous, communications costs are non-existent.

TABLE I: Multi-agent testbeds facing the 8 fallacies of MAS.

Criteria \ Platform	Mages	RobotCup Rescue	RobotCup Soccer	Pommerman	Starcraft II	Gym (openAI)	Repast	Gama	Dedale
(1) Asynch. agents	✓	✗	✗	✗	✗	✗	✗	✗	✓
(2) Heterog. agents	✓	✓	✗	✗	✓	✓	✓	✓	✓
(3) Comput. limited agents	✗	✓	✗	✗	✗	✗	✗	✗	✓
(4,5) Reliability and safety	✓	✗	✗	✗	✗	✗	✗	✗	✓
(6) Uncertain & P. observable	✓	✓	✓	✓	✓	✓	✓	✓	✓
(7) Open-word	✗	✓	✗	✗	✗	✗	✗	✗	✓
(8) Asynch. communication	✓	✓	✗	✗	✗	✗	✗	✗	✓
Team-size > 2	✓	✓	✓*	✗	✓*	✗	✓	✓	✓
Nb-teams > 2	✓	✓	✗	✓	✓	✓	✓	✓	✓

observation and a fixed number of agents thus conducts proposed solutions to be non-scalable. (8) To assume a perfect, instantaneous and zero-cost communication network lead to bias the evaluation of all proposed solutions.

Finally, from an initially easy to understand and tackle team-of-two independent agents setup, the environment should allow users to progressively raise the complexity by increasing the number of agents as well as to remove simplifying but unrealistic hypotheses (either online or offline). Indeed, this progressivity is a necessary step to evaluate and validate both the robustness and scalability of a decentralised proposal.

Table I summarizes the behavior of the main simulation environments available today to study multi-agent problems with respect to the 8 fallacies of MAS.
The Repast [13, 27] and Gama [41] platforms are central when it comes to agent simulation, but, focusing on large-scale Complex Adaptive Systems, they are not designed to study coordination problems. Their objective is to simulate and study the behaviour are emerging from tens of thousands of agents for problems related to transport, climate or epidemiology within GIS-based environments. Communications between agents simply does not exist, and agents runs on a turn-by-turn basis.

On the opposite spectrum, a platform such as Gym [9], dedicated to reinforcement learning research, is focused on mono or 2 agents use-cases with unlimited computation capabilities and no explicit coordination.

If the well-known Starcraft II testbed [1] offers the possibility to confront more than two adversaries at the same time, no coordination can take place as each agent controls all the entities that compose its "team" and as no communication channel exist between entities within the game. Although technically impressive, *DeepMind* recent results [45] are no exception. The multi-agent dimension they consider is for them to specialise 3 agents in order to play the 3 available Starcraft races. The same limitation appears with the Robocup Soccer simulator [20]. It is theoretically possible to consider a team composed of more than 2 autonomous agents, but, in practice, the framework's incentive leads users to develop one central agent that efficiently control all the entities that compose a team.

Pommerman [33], a multi-agent environment based on Bomberman, assumes closed-world environments where evolve up to only 4 homogeneous agents in zero-sum games between two teams [33]. Communication is synchronous and restricted to 2 Ints in [0,8]. Focused on team-of-two agents learning, it does not currently allow users to scale-up and thus meanly attract the research community around Deep reinforcement learning.

To our knowledge, RoboCup Rescue [40] and Mages [8] platforms are the closest ones to cover all required criteria to study and evaluate multi-agent coordination, learning, and decision-making under realistic hypotheses. If Mages is no longer supported (nor available), RoboCup rescue is still actively developed. In addition to asynchronous and potentially failing communications, Robocup Rescue is the only one (with Dedale) to support explicit limitation of the computational capabilities of the agents[2]. Nevertheless, to guarantee their scenarios to scale-up to hundreds of agents, they favour simple agents' architectures and run in a turn-peer-turn unrealistic setting which removes much of the complexity inherent to decentralised problems.

We aim for the Dedale testbed to provide for the multi-agent community a plateform allowing to work under parametrizable but realistic hypotheses and environment setups without sacrificing the number nor the complexity of deployable agents.

II. The Dedale platform

Figure 1 illustrates Dedale's general architecture. From a synthetic standpoint, it combines the well-known Jade framework and the Inter-Platform Mobility Service (IPMS) [6] with the GraphStream (GS) [18] and jMonkeyEngine (JME3) [26,34] libraries. While Jade is in charge of the MAS management, GraphStream and Jme are respectively handling the two types of environments currently provided by Dedale : discrete dynamic graphs and continuous 3D-environments. Users' agents will then have to evolve within them to accomplish their goals.
Note that Dedale and its components are all open-source. Our website [21] contains all required information on how to install, configure and run it.

[2]This was done in order for the users to take into account the scarcity and unreliability of available resources in disaster-like situations

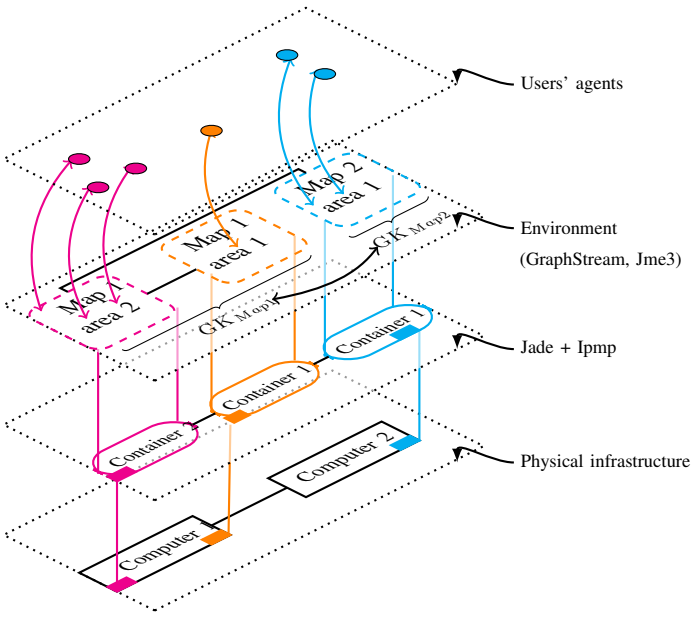


Fig. 1: Dedale's architecture overview : The Jade platform combined with the IPMP component is used as a baseline to create a distributed environment. One GateKeeper's agent (*GK*) is deployed on each computer and is in charge of the local environment's creation and management. It maintain a view of the distant locations that can be reached from its local map and takes care of incoming and exiting migration requests made by users' agents intending to complete their missions.

The use of Jade allows us to make real-life hypotheses on the agent's functioning. Each agent is a thread, their respective execution speed can thus vary. Furthermore, there is no bound on the transmission time of a message sent between agents (messages can be lost, delayed, or have a maximum reach depending on the users' choices). We are thus considering both system and communication asynchronism. From these choices, we avoid the 1^{st},2^{nd} and 8^{th} fallacies previously introduced. To this point, we made implementation choices guaranteeing to avoid the 5 remaining fallacies of MAS.

The types of continuous environments we currently offer through the JME game-engine increases the complexity of the environment and the planning needs for the agents without intrinsically changing the properties of the problems considered. Thus, for sake of clarity, we will focus in the present document on the case of discreet environments, referring to the continuous case only when the agents' behaviour is significantly impacted.

In the discrete case (see figure 2), the environment is a graph. Nodes are rooms associated with a unique identifier where the agents will find objects to interact with, and edges are non-oriented corridors between them. The topology can be hand-made, generated or imported from OpenStreetMap [15]. The number and the initial location of both objects and agents can also be either generated or hand-picked. By default, only one agent can be on a given node at any given time. However, the user can decide otherwise and individually set higher capacities to nodes.

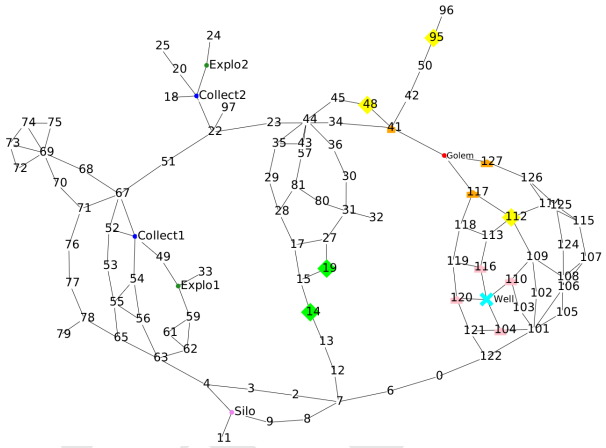


Fig. 2: Part of a 400-nodes hand-made environment integrating treasures (gold and diamonds), wells, golems and several teams of 10 heterogeneous agents. 1 Tanker, 2 Explorers and 2 Collectors are visible.

A. Activable elements within the environment

1) Wells: They are dangerous nodes where any agent coming-in falls and dies. Wells generate wind in their surrounding nodes (the distance is parametrizable), giving a warning to any passing agent.

2) Treasures: A treasure is stored within a safe, and is defined by its type (gold or diamonds), its value and its lock (a set of skills required to open it combining strengh and lockpicking expertise).

When an agent (see next section) is in a room where a treasure is, and if its backPack and expertise allow it, then the agent can grab it.

- If the safe is locked and the agent does not possess sufficient expertise to open it, he needs to call for help. Several agent can indeed pool their forces to meet the lock requirements.
- If the agent backPack is full or the treasure is a different type than his, then he grabs nothing.
- If the agent cannot grab all the treasure of an opened safe, the remaining treasure stays on the map. Another agent can come and take it.

The system is designed so that an agent drops a portion of the treasure by picking it up. The more picking actions are made on a treasure, the more the loss is important. This will therefore conduct the agents to opt for picking up a single treasure in a minimum of actions.

3) Benchmark agents (Golems): They are "native" Dedale's agents activable within the environment. Dedale currently offers three types of Golem's behaviours :

- `DummyMovingGolem` : As he moves, the Golem will shift all or part of the treasures he discovers.

- `CollectingGolem` : He will explore the environment and collect the treasures he finds.
- `EscapingGolem` : He will try to move around indefinitely.

Golems release a stench which allow other agents to detect them before encountering them. In Fig. 2, the emission radius is set to 1. As we will see in section III, Golem(s) significantly increase the complexity of the tasks to be performed by user's agents.

B. User's agents

Dedale's agents can be of 3 different types :

- `Explorer` : It cannot collect, only explore
- `Tanker` : It possesses a nearly unlimited storage capacity, but cannot pick anything
- `Collector` : It possess a backpack specific to a given type of treasure, and a limited carrying capacity. It can pick a treasure, and drop it in the Tanker if the latter is in reach

The scenario designer can decide to activate one or several type of agents.

1) Agent-centered representation: At the beginning, agents are deployed (randomly or not) on the map. The agents do not know the topology beforehand and thus have to discover and learn it.

a) Limited percepts: When an agent is in a room, it can only perceives :

- The id – unique – of the room.
- The names of the others agents in the room, if any.
- The occurrence, if any, of a treasure, its type, value and opening conditions.
- The links to the neighbouring rooms (and associated observations, if any).

b) Limited communication range: Agents can communicate through messages to share their knowledge, coordinate and optimize their actions. Nevertheless, the reach is limited. An Agent can only communicate with the ones in the neighbourhood (a distance that can vary with each agent). The coordination solutions should thus take into account this fact.

2) Agent observations and actions: In the absence of treasures, 4 actions are available to the agents, 9 otherwise.

a) Observation, communication and motion:

- `getCurrentPosition()` : Returns the current position of the agent
- `observe()` : Returns the set of observables that can be perceived from the agent current position as a list of couple (position,list(ObservationType,Value))
- `moveTo(myDestination)` : Makes the agent move to myDestination (if reachable)
- `sendMessage(msg)` : Send a message and manage the communication radius. It can contain any information. If the recipient is not in range, the message is lost.

Additionally, if its a treasure-type scenario, then the following actions are available to the agents :

b) Treasures and safes:

- `getMyTreasureType()`: Type of treasure that the agent can grab (only one type per agent)
- `getMyExpertise()`: Expertise of the agent (strengh and lockpicking) to open safes
- `openLock()`: Open the safe present in the current room if the required expertise is provided.
- `pick()`: Grab all of any of the treasure available on the current position (according to agent type, capacity and safe state)
- `EmptyMyBackPack(agentTankerName)`: Allow the agent to transfer its backpack within the Tanker agent in the vicinity (if any).

C. Performance analysis

To properly compare the performances of different proposals to a given problem instance, several evaluation metrics are available.

- The number of messages exchanged between the agents
- The number of actions executed
- The overall time needed to complete a task

In the case of a treasure hunt, the quantity of collected resources is also stored. Finally, for both analytical and debugging purposes all of the agents' actions are logged.

III. Multi-agents use-cases

In this section, we present the main classes of multi-agent problems that can be currently studied through the use of Dedale as well as the associated open research issues.

A. Distributed exploration

Consider an unknown finite graph, a set of agents randomly deployed, a limited communication range and the impossibility for two agents to be on the same node at the same time. How to efficiently explore the environment ? This classical problem was initially formulated by Shannon [37] and has been extensively studied since then (see [16, 32, 38]).

If uniquely labelled nodes and no harmful situations allow a smooth exploration through the use of Distributed DFS-like algorithms, the activation of the well(s) introduced in section II-A1 significantly change the problem. Indeed, any agent visiting a "well-node" is instantaneously destroyed. In this context, the complete exploration of the environment require for at least one agent to survive knowing all edges leading to the well(s). The location of the harmful node(s) must thus be based on a coordinated exploration and communication strategy.

- What is the minimal number of agents required to locate all wells and finish the exploration ?
- For a given number of agent and wells, what is the quickest possible strategy and the minimal number of messages exchanged ?

In synchronous cases, when the agents know the topology beforehand or when they start from the same location and

have unlimited communication range, some proposal and complexity bound have been established [14, 17, 19]. Nevertheless, to our knowledge, no algorithm seems to have so far been proposed in the general setting above-mentioned.

Another interesting variation offered by Dedale is to study the cooperative exploration of an unlabelled graph. Once again, under the previously introduced hypotheses, no optimal strategy seems to currently exist [10].

B. Cooperative patrolling and pursuit-evasion games

The second setup is the well known patrolling problem and its related pursuit-evasion game. Consider an unknown finite environment E, a set of agents A, a set of intruders I and limited communication ranges C.

- What is the minimal size of A in order to guarantee that there exist a finite solution to detect and block all the intruders in E ?
- For a given number of agent (and intruders), what is the best possible strategy ?

To determine the minimal number of agents required for any graph is an NP-Hard problem [3, 12], but several heuristics have been proposed [4]. Regarding the optimal strategy for a given A and I, if its easy on a (finite) binary tree with unlimited communication range and only one intruder, it becomes more difficult when you set a limited communication range, use a complex environment such as Fig. 3a and annoyingly clever intruder(s). In this context existing proposals mainly tend to decrease the size of the subgraph that the evader(s) can safely reach instead of minimizing the distance between pursuers and evader(s).

As illustrated in Fig 4, the key problem is that an already explored (and thus considered as cleared) area can be contaminated by an intruder moving behind the agents' back, thus requiring to maintain containments areas whose size depend of both the graph structure, the number of patrolling agents and of their respective communication ranges.

Numerous work have been published regarding multiagent patrolling, either from the robotic [2, 30] or mas [7, 11, 31] communities. But little attention has been paid to the impact of asynchronism, limited communication range or dynamic environments on the agents cooperation [28]. We believe Dedale to be a suitable testbed in these directions.

C. Treasure(s) hunt & pickup and delivery problems

This last setup, the treasure hunt, is the more complex one as it can include the two previous and create an additional level of difficulty through the agents' heterogeneity and the problems induced. Note that the pickup and delivery problem [36] is a subset of the treasure hunt problem where the pickup and delivery locations are *a priori* known.

Consider an unknown finite environment with treasure chests, a set of heterogeneous agents randomly deployed and limited communication ranges. Assuming that the team's score only takes into account the treasures stored within the tankers at the end of the time limit:

- What is the best strategy to maximise the overall treasure value within the Tankers' backpacks ?

As highlighted in II-A2, the agents have to minimize the number of picking to maximise the amount of treasure grabbed. The agents goals are thus to : (1) locate the treasures, (2) collectively decide on the coalitions required to efficiently open them, (3) determine the optimal picking order as well as (4) quickly and repetitively find the tankers to unload their backpacks.
As agents initally start from distinct locations and do not posses an unlimited communication range, steps 2 to 4 require to solve the *rendez-vous* problem [29]. Moreover, steps 3 and 4 will make the interlocking problem arise.

1) Path planning and the interlocking problem: Interlocking are non-existent during the exploration phase: When two agents meet each others they only need to share their respective knowledge to get what they were immediately looking for and solve their momentary issue. In the case of a treasure hunt, the heterogeneity of the agents will on contrary generate non-interchangeable objectives and a dynamical hierarchy between agents. Fig. 5 illustrates a typical situation.

A_1 wants to go the node where A_3 is and conversely. At the same time A_2 is moving in towards its goal, node 7, increasing the interlocking complexity. Here, the locked agent (A_3) should become prioritary over A_1 as it has no escaping path to let A_1 going through. This priority should not be of one-action duration, otherwise A_1 and A_3 will oscillate "infinitely", soon joined by A_2. The obvious solution is, in the current example, for A_1 and A_2 to move towards nodes 1 and 2 and wait until A_3 leaves the sub-tree and inform them of this fact before leaving trough node E.

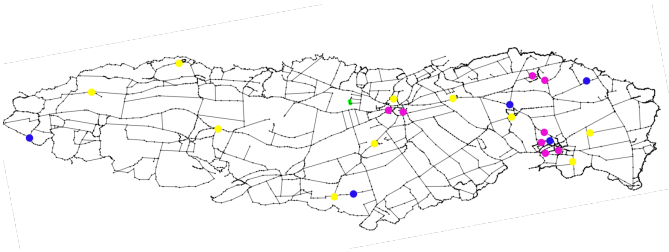
In the general case, multi-agent path planning has been shown to be a PSPACE-hard problem [23]. Even when the information are available, a global search is only tractable by aggregating the agents [39]. In real conditions, from air traffic control to computer games, decentralised replanning and heuristic procedures are the preferred methods to break the occurring interlocks [46].

Capable of transparently support a hundred of agents on a standard computer[3], Dedale offers a flexible testbed to study and compare the efficacy, the scalability, and the robustness of those approaches.

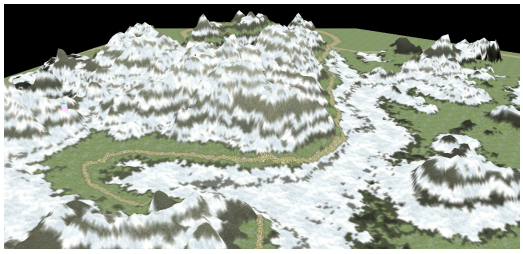
2) Dynamicity and decision-making trade-off: If we release the Golem(s) or allow several adversarial teams on the same map, the environment becomes dynamic.

- In the case of the Golems, they will move the treasures. Consequently, two agents meeting each others might face inconsistency while sharing their knowledge. Until now, the value of a treasure on a given position could only diminish. Now it can also increase. Without the use

[3]The number of agents that Dedale can manage without slowing down depends on their inner complexity as well as on available processing cores.



(a) Groix island, France. Topology imported from *OpenStreetMap* and used as default testbed for real-world discrete case-study for mas patrolling, treasures hunt or pickup and delivery problems. Pick-up locations are represented as treasures (yellow) and delivery locations as Tanker agents (purple). The collector agents (blue nodes) are the delivery vehicles.



(b) Mountainous area, Auvergne, France. Heightmap imported from *terrain.party* and used as default testbed for 3 dimensional and continuous environments. Only grounded agents are currently considered.

Fig. 3: Dedale's discrete and continuous default testbeds for real-word environments. Users can easily choose the topology they want to consider for their experiments, either it is generated or based on a real geographical area.

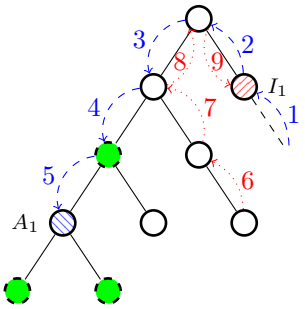


Fig. 4: The containment problem : Agent A_1 is looking for the intruder I_1. Its field of view allow him to check the neighbouring nodes. While A_1 explores the graph (arrows 1 to 5), the intruder can move behind its back and join a previously secured area (dotted arrows 6 to 9).

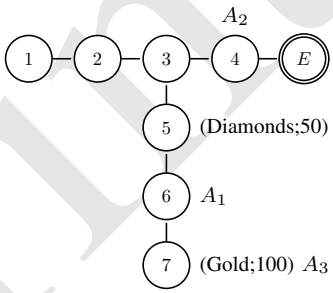


Fig. 5: Interlocking problem : Let 3 agents A_1,A_2 and A_3 on a known graph and their communication range set to 2. Node E represents the exit, any agent moving in will transfer its backpack to the tanker. Agents A_1 and A_2 are of gold type, and their backpack capacities are respectively 80 and 10. A_3 is of diamond type and posses a 51 carrying capacity. Each agent's goal is to reach the node possessing their resource type. When picking, 50% of the remaining resources is definitely lost. No prior coordination have been made.

of common clock, the agents are no longer capable of identifying the most recent information.

- In the case of adversarial teams, in addition to collecting resources some adversaries may block the access to a subgraph without the possibility for a member of a different team to trigger interlocking procedures. The topology of the accessible graph will thus change over time.

In both cases the agents will have to significantly adapts their strategy to this situation. When a treasure is discovered, each agent have to choose between picking it up (if possible), protecting it or looking for his team before someone else discovers it. The chosen decision-making process (robust, risk-adverse,..) will greatly influence the result of the team and we believe cooperative learning to be perfectly fitted for the emergence of innovative and adaptive strategies in this context.

IV. From a local testbed to a peer-to-peer environment for long-term experiments in open-world

We have so far introduced the different characteristics of Dedale and highlighted its advantages, inherent to the fact that the 8 fallacies of MAS are taken into account. What we believe to be an additional stand-out factor comparatively to the literature is its ability to work as a node in a network of Dedale's environments.

Indeed, after conscientiously designing their environment and developing the capabilities of their agents, each user can decide to open incoming or outgoing access to their instance. This offers for the user's agents the possibility to explore foreign and unknown environments, and for foreign agents to discover and test their robustness facing an environment – and potentially the agents's – the local user designed. The size, the complexity and the richness of the "networked" environment accessible to the agents therefore becomes virtually unlimited. This is made possible by the fact that Dedale relies on the JADE multiagent platform [6], which supports inter-platform migration. When activated, this feature gives two additional actions to the agents :

- `getNeighbours()`: The list of Dedale's nodes connected to this environment.

- `migrateTo(myDestination)`: Makes the agent migrate to the distant environment it is willing to be deployed in (if the authorisation is granted).

For the sake of clarity we will only present here the key points associated with this peer-to-peer feature from the agent perspective.

A. The Gatekeeper

To allow foreign agents to dynamically move-in and out of a Dedale instance require to manage two critical elements : security and deployment. Both of them are at the Gatekeeper's expense. As illustrated on Fig. 6, its a supervision agent that authorises or refuses the access to the environment it manages according to the local user preferences as well as on the locally available computational and memory capacities.

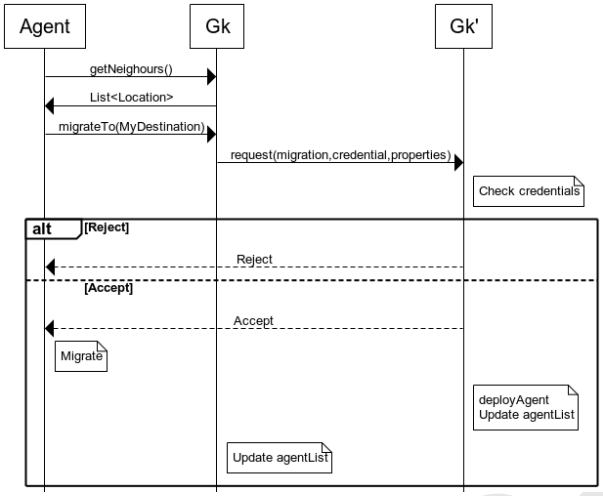


Fig. 6: Migration process : Gk is the local gatekeeper, Gk' the distant one.

Once the authorisation received, the local gatekeeper initiates the agent transfer with the distant one following a transaction-type process. All classes that constitute the agent are serialized and sent over the network[4]. A reference to the environment managed by the local gatekeeper is given to the agent while him and his previous knowledge are instantiated. Once the migration achieved, the original agent is transparently destroyed by its former gatekeeper.

B. Supervising distant agents

To allow its agents to leave the nest represents both an opportunity and a risk. An opportunity for learning agents to improve their knowledge and performances, a risk for all to die if the new environment is dangerous or the platform unstable. Either way, to supervise and monitor distants agents is challenging. Dedale currently offer no default backup for the agents, their logs are stored within them and only written on disk when and *where* the agent dies. Nevertheless, through its local gatekeeper, the user keep a minimalistic control over its agents and can send two predefined priority-orders to each of them : `goHome` and `terminate`. Thus, a misbehaving but living learning agent can still be repatriated and its knowledge and behaviours analysed.

[4]All of the agent's classes and libraries should thus be serializable.

We believe Dedale's peer-to-peer network feature to be suited for multi-agent coooperative learning and research on long-term autonomy. The user can at the same time tailor its environment to meet its experiments requirements and let its agents face the unexpected, guaranteeing Dedale to be a challenging environment for years to come.

Conclusion

Avoiding what we introduced as the 8 fallacies of MAS, we opensource [21] the first testbed exclusively designed for studying truly *multi*-agents problems under real-life hypotheses : asynchronous agents evolving in open, dynamic, and partially observable continuous and discrete environments. From distributed exploration to cooperative pick-up and delivery, including pursuit-evasion games, Dedale offers a wide range of multi-agents use-cases which represent as many open research challenges.

Our next steps will be : (1) To set up a public and automated ranking of the best strategies found for each configuration. (2) To propose a peer-to-peer network management algorithm allowing the automatic placement of Dedale's nodes according to their reliability, thus offering a classification and a cartography of relatively (un)safe areas for the agents to explore.

Dedale was successfully tested by more than 150 early-users, either researchers or post-graduate students on distributed AI. The very positive feedbacks confirm its ease of use and the progressiveness offered. In its educational dimension, its led in particular to the publication of a press article by one of its users on the richness and difficulties inherent to multi-agent systems [42].

More generally, we believe that Dedale offers a challenging but accessible – because fully parametrizable – combination of coordination, decision-making and learning situations, both for research and teaching activities, outside of the too restrictive multi-agents team-of-2 setup. Its peer-to-peer feature also opens new perspectives with the possibility for researchers to study agents' capabilities to remain autonomous on a long term basis at minimal cost. Finally, if easily reproducible experiments with teams of dozens or hundreds of agents are not common today, we believe that Dedale provides a step towards this goal.

References

[1] Khan Adil, Feng Jiang, Shaohui Liu, Worku Jifara, Zhihong Tian, and Yunsheng Fu. State-of-the-art and open challenges in rts game-ai and starcraft. International Journal of Advanced Computer Science & Applications, 8(12):16–24, 2017.

[2] Noa Agmon, Sarit Kraus, and Gal A Kaminka. Multi-robot perimeter patrol in adversarial settings. In 2008 IEEE International Conference on Robotics and Automation, pages 2339–2345. IEEE, 2008.

[3] Brian Alspach. Searching and sweeping graphs: a brief survey. Le matematiche, 59(1, 2):5–37, 2004.

[4] Adonis Antoniades, H Jin Kim, and Shankar Sastry. Pursuit-evasion strategies for teams of multiple agents with incomplete information. In 42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475), volume 1, pages 756–761. IEEE, 2003.

[5] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, 2013.

[6] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. Developing multi-agent systems with JADE, volume 7. John Wiley & Sons, 2007.

[7] Aurélie Beynier. Cooperative multiagent patrolling for detecting multiple illegal actions under uncertainty. In 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), pages 9–16. IEEE, 2016.

[8] Thierry Bouron, Jacques Ferber, and Fabien Samuel. Mages: A multi-agent testbed for heterogeneous agents. Decentralized Artificial Intelligence, 2:195–214, 1991.

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

[10] Jérémie Chalopin, Shantanu Das, and Adrian Kosowski. Constructing a map of an anonymous graph: Applications of universal sequences. In International Conference On Principles Of Distributed Systems, pages 119–134. Springer, 2010.

[11] Yann Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004.(IAT 2004)., pages 302–308. IEEE, 2004.

[12] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. Autonomous robots, 31(4):299, 2011.

[13] Nick Collier. Repast: An extensible framework for agent simulation. The University of Chicago's Social Science Research, 36:2003, 2003.

[14] Colin Cooper, Ralf Klasing, and Tomasz Radzik. Searching for black-hole faults in a network using multiple agents. In International Conference On Principles Of Distributed Systems, pages 320–332. Springer, 2006.

[15] Copyright OpenStreetMap contributors. Dump retrieved from https://www.openstreetmap.org, 2019.

[16] Humbert Fiorino. Damien Pellier. Coordinated exploration for labyrinthine environments with application to the pursuit-evasion problem. In Workshop on Cooperative Robotics (IROS), pages 50–58, 2002.

[17] Shantanu Das, Paola Flocchini, Shay Kutten, Amiya Nayak, and Nicola Santoro. Map construction of unknown graphs by multiple agents. Theoretical Computer Science, 385(1-3):34–48, 2007.

[18] Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007), 2007.

[19] Pierre Fraigniaud, Leszek Gasieniec, Dariusz R Kowalski, and Andrzej Pelc. Collective tree exploration. Networks: An International Journal, 48(3):166–177, 2006.

[20] Matthew Hausknecht, Prannoy Mupparaju, Sandeep Subramanian, Shivaram Kalyanakrishnan, and Peter Stone. Half field offense: An environment for multiagent learning and ad hoc teamwork. In AAMAS Adaptive Learning Agents (ALA) Workshop, 2016.

[21] Cédric Herpson. Dedale : Will your agents succed ? - https://dedale.gitlab.io/, 2018.

[22] John H Holland. Studying complex adaptive systems. Journal of systems science and complexity, 19(1):1–8, 2006.

[23] John E Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem". The International Journal of Robotics Research, 3(4):76–88, 1984.

[24] Hiroaki Kitano and Satoshi Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. AI magazine, 22(1):39–39, 2001.

[25] Kalliopi Kravari and Nick Bassiliades. A survey of agent platforms. Journal of Artificial Societies and Social Simulation, 18(1):11, 2015.

[26] Ruth Kusterer. jMonkeyEngine 3.0 Beginner's Guide. Packt Publishing Ltd, 2013.

[27] Michael J North, Nicholson T Collier, Jonathan Ozik, Eric R Tatara, Charles M Macal, Mark Bragen, and Pam Sydelko. Complex adaptive systems modeling with repast simphony. Complex adaptive systems modeling, 1(1):3, 2013.

[28] Mehdi Othmani-Guibourg, Amal El Fallah-Seghrouchni, Jean-Loup Farges, and Maria Potop-Butucaru. Multi-agent patrolling in dynamic environments. In 2017 IEEE international conference on agents (ICA), pages 72–77. IEEE, 2017.

[29] Andrzej Pelc. Deterministic rendezvous in networks: A comprehensive survey. Networks, 59(3):331–347, 2012.

[30] David Portugal and Rui Rocha. A survey on multi-robot patrolling algorithms. In Doctoral Conference on Computing, Electrical and Industrial Systems, pages 139–146. Springer, 2011.

[31] Cyril Poulet, Vincent Corruble, Amal El Fallah Seghrouchni, and Geber Ramalho. The open system setting in timed multiagent patrolling. In 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, volume 2, pages 373–376. IEEE, 2011.

[32] Jennifer Renoux. Contribution to multiagent planning for active information gathering. Phd thesis, Normandie Université, September 2015.

[33] Cinjon Resnick, Wes Eldridge, David Ha, Denny Britz, Jakob Foerster, Julian Togelius, Kyunghyun Cho, and Joan Bruna. Pommerman: A multi-agent playground. arXiv preprint arXiv:1809.07124, 2018.

[34] Rafaela V Rocha, Rodrigo V Rocha, and Rb Araújo. Selecting the best open source 3d games engines. In Proceedings of the brazilian Symposium on Games and Digital Entertainment, Florianópolis, Santa Catarina, brazil, 2010.

[35] Arnon Rotem-Gal-Oz. Fallacies of distributed computing explained. URL http://www. rgoarchitects. com/Files/fallacies. pdf, 20, 2006.

[36] Martin WP Savelsbergh and Marc Sol. The general pickup and delivery problem. Transportation science, 29(1):17–29, 1995.

[37] CE Shannon. Presentation of a maze solving machine. cybernetics: Circular, casual, and feedback mechanisms in biological and social systems. In Transactions Eighth Conference, von Foerster, H., Mead, M. and Teuber, HL (eds). New York, Josiah Macy Jr. Foundation, pages 169–181, 1952.

[38] Das Shantanu. Graph Explorations with Mobile Agents. In Distributed Computing By Mobile Entities, volume 11340 of Lecture Notes in Computer Science, chapter 18, pages 403–422. January 2019.

[39] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence, 219:40–66, 2015.

[40] Cameron Skinner and Sarvapali Ramchurn. The robocup rescue simulation platform. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, pages 1647–1648. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[41] Patrick Taillandier, Benoit Gaudou, Arnaud Grignard, Quang-Nghi Huynh, Nicolas Marilleau, Philippe Caillou, Damien Philippon, and Alexis Drogoul. Building, composing and experimenting complex spatial models with the gama platform. GeoInformatica, 23(2):299–322, 2019.

[42] Mickael Trazzi. Why coding multi-agent systems is hard - https://hackernoon.com/why-coding-multi-agent-systems-is-hard-2064e93e29bb, 2018.

[43] Jelle Van Gompel, Bart Tuts, Rutger Claes, Mario Henrique Cruz Torres, and Tom Holvoet. Mas-discosim 4 pdp: a testbed for multi-agent solutions to pdps. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pages 1–2, 2010.

[44] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. DeepMind Blog, 2019.

[45] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, pages 1–5, 2019.

[46] Ko-Hsin Cindy Wang, Adi Botea, et al. Fast and memory-efficient multi-agent pathfinding. In ICAPS, pages 380–387, 2008.