Plan Recovery Process in Multi-agent Dynamic Environments

Leonardo Henrique Moreira[a] and Célia Ghedini Ralha[b]
Department of Computer Science, Institute of Exact Sciences, University of Brasília, Brazil

Keywords: Multi-agent Planning, Simulation, Dynamic Environments.

Abstract: Planning is the process that focuses on the choice and organization of actions through their expected effects. Plans can be affected by unexpected, uncontrolled, non-deterministic events leading to failures. Such challenging problem boosted works focusing agent distribution, communication mechanisms, privacy, among other issues. Nevertheless, the plan recovery process does not have a defined standard solution. Thus, in this work, we present a three-phase plan recovery process to provide resilience to agent plans by supporting a staggered solution. Whenever an action execution fails, agents try to solve individually through their own capabilities. But when not possible, agents start an interaction protocol to ask for help. Finally, when previous two phases were unsuccessful, a centralized planning process is trigged. Regardless the phase in which the solution is found, agents' plans are coordinated to guarantee cooperation maintaining information privacy. An empirical analysis applying metrics such as planning time, final plan length and message exchange was conducted. Results give statistical significant evidence that agents' autonomy is better explored in agents' loosely coupled environments. The contributions of this work include: a three-phase plan recovery process, a simulation tool for benchmarks, and a statistical robust evaluation method to multi-agent planning.

1 INTRODUCTION

In multi-agent system (MAS), agents interact with the environment and with other agents through the execution of actions towards the transformation of the environment from a initial to a desirable state. The sequence of these actions is a plan, being the result of agents' deliberation process performed before or during acting (Ghallab et al., 2014). Furthermore, planning and execution responsibilities are distributed over multiple entities. Thus, agents must coordinate continually their efforts towards the satisfaction of individual or global goals to guarantee the cooperation by defining a multi-agent planning (MAP) model. MAP applications are required in ordinary to complex tasks, such as logistic and fire rescue activities, respectively. A common point in both scenarios is the fact that agents depend on or affect other agents under different levels, regarding the interaction that their action executions induce.

In a classical MAP models, changes in the environment are expected to happen only by agents' action executions. However, those models become ineffective in scenarios where events that are neither controlled nor expected by agents can happen and update the environment state. Those events, labeled as exogenous, are common in dynamic environments. In this case, the planning process performed in an earlier and single phase (disconnected from execution) is not able to provide resilience to agents (Ghallab et al., 2014; Komenda et al., 2014; Chrpa et al., 2020).

In the related work, the MAP models are composed by two different and isolated recovery strategies: replanning and repairing. Although each strategy has its pros and cons, MAP models that apply both of them regarding agents' capabilities are missing in the literature.

Therefore, our goal is to propose a plan recovery process that combines repairing and replanning strategies, providing conditions to agents to perform a staggered solution. As soon as agents detect an action execution failure, they first try a local repair by planning individually, considering only their own capabilities. In this sense, agents keep the information privacy because they do not inform which actions they are able to perform. When the local repair phase is not possible, agents interact to ask for help. Finally, if even so no solution is returned, a centralized planning phase is triggered. Therefore, the proposed process tries to avoid message exchange in conditions where

[a] https://orcid.org/0000-0003-0479-578X
[b] https://orcid.org/0000-0002-2983-2180

Moreira, L. and Ralha, C.
Plan Recovery Process in Multi-agent Dynamic Environments.
DOI: 10.5220/0010559301870194
In *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2021)*, pages 187-194
ISBN: 978-989-758-522-7

a local solution is possible. This ability is important in environments where communication is complex or agents must deliberate quickly towards the solution of global goals, for instance, during rescue operations after catastrophic events.

This work was motivated by the hypothesis that agents' autonomy in performing local repair is better explored in environments with low levels of interaction. Therefore, we followed a research methodology that was guided by the literature review, process design, simulation of benchmarks and results analysis with enough evidence to accept or reject the research hypothesis.

Beyond the development of the plan recovery process, our contributions to the MAP area also include a simulation tool for benchmarks executions and a statistical evaluation method regarding final plan length, planning time and message exchange.

The rest of the document is structured as follows. In Section 2, we present the MAP concepts. In Section 3, we detail the proposed plan recovery process along with the simulation tool, while in Section 4 we describe the experiments and then, we discuss the results following statistical evaluation method. Finally, we present conclusion and future work in Section 5.

2 BACKGROUND

In Section 2.1, we present MAS and MAP formal definitions. In Section 2.2, we detail plan recovery strategies described in related work.

2.1 MAS and MAP Concepts

A MAS is a set of software entities able of sensing (sensors) and modifying the environment state using their actuators (Weiss, 2013). Thus, a MAS is composed by autonomous agents that interact in a shared environment through a communication protocol. To interact agents need also a coordination model that can put together competitive or cooperative agents. A negotiation protocol is required when competitive agents are defined. But, when cooperative agents interact in the same environment, a planning protocol is necessary to define the individual and group goals, avoiding conflicts in the shared resource usage.

Furthermore, MAP can be understood as the planning and executing process distributed over multiple agents (Torreño et al., 2017). The agent distribution characteristic focuses on the number of agents and the roles they adopt during the process of finding a solution for the problem. The agents involved in the reasoning stage of synthesizing the sequence of actions (plan) are the planning entities. Executors are agents committed to execute actions, such as a robot or a software entity in a simulator.

The MAP models can be specialized into many different models and approaches regarding the assumptions made about actions: deterministic, hierarchical, temporal, non-deterministic and probabilistic. In this work, we assume some premises to deal with MAP similarly to related work (Borrajo and Fernández, 2019): the environment is fully observable, agents are collaborative, actions are unit cost and instantaneous, communication process is free of failures.

An operator θ is a schema that defines actions using parameters and is represented by: $name(\theta)$, an identification to the operator; $pre(\theta)$, the set of precondition that stands for literals required to apply the operator; and $eff(\theta)$ which is formed by a set of effects that stands for literals which are added (eff^+) or deleted (eff^-) from the state of the world after executing the operator.

The set of operators, types and parameters defines the planning domain. An action is a particular operator instantiation, where all parameters are replaced by objects. So, the set of available actions, AG_i, is formed by the combination of the every parameter value. A tuple that is formed by all available actions, logical propositions, the environment initial state I and the goal G is defined as the planning problem.

In a multi-agent environment, agents' plans coexist. Thus, a MAS requires coordination when agents execute their plans simultaneously, because they can compete for some resources or even undo the effects of each other's actions. This coordination requirement derives from the fact that actions can be public or private. According to Definitions 1 and 2 from (Brafman and Domshlak, 2008).

Definition 1. *An action is public whenever some propositions of its preconditions or effect appears in an action that belongs to other agents. The set of all public actions is defined by:*

$$A^{Pub} = \{\alpha | \exists i, j : i \neq j, \alpha \in A^i, \alpha' \in A^j, \text{ and } (pre(\alpha) \cup eff(\alpha)) \cap (pre(\alpha') \cup eff(\alpha')) \neq \emptyset\}.$$

Definition 2. *An action is private whenever it does not affect nor depends on actions that belongs to other agents. The set of all private actions is defined by:*

$$A^{Priv} = A \setminus A^{Pub}.$$

When planning concerns only about private actions, it can be performed locally since actions do not depend or are not dependent of other agent's actions (Komenda et al., 2014). Under this condition, a coordination process is not necessary because there is no

competition or cooperation issues. Therefore, the coordination complexity is formalized as a function of the number of actions executed by an agent that affect or depend on other agents. Complexity derives from the agents' coupling level or the interactions, that are required to control the dependency among agents (Brafman and Domshlak, 2008).

Regarding action classification (Definitions 1 and 2) and coordination complexity, planning problems can be defined as follows. In loosely-coupled domains, most of the available actions are private leading to low interaction level. In tightly-coupled, agents need to interact to satisfy goals because of the public behavior of actions.

In a multi-agent plan actions scheduled to be started at the same instant must be independent of each other to guarantee cooperation and avoid competition. A multi-plan ρ, with $\pi[i,t] = \alpha_t^i, i \in AG, t \geq 1$ is a sequence of actions that can be executed in parallel by different agents at the same instant t. Thus, the set of actions of a plan ρ to be executed at the instant t is $A_{\rho_t} = \{\alpha_t^i | \forall i, j \in AG : i \neq j, \alpha_t^i \in \rho, \varphi(\alpha_t^i, \alpha_t^j) = \emptyset\}$. Data representation of multi-agent plan ρ is:

$$\rho = \begin{bmatrix} \alpha_1^1 & \alpha_2^1 & \varepsilon \\ \alpha_1^2 & \varepsilon & \alpha_2^2 \\ \varepsilon & \alpha_1^3 & \varepsilon \end{bmatrix}$$

An agent updates the environment state by executing an action. The transition caused by the action α applied in a state s is defined by Equation 1.

$$\gamma(s,\alpha) = (s \setminus eff^-(\alpha)) \cup eff^+(\alpha) \quad (1)$$

2.2 Plan Recovery Strategies

Problems may happen during the execution of a plan because of different reasons. Actions may be not executed because some of their requirements (preconditions) are not held. For instance, a previous action did not performed as planned. The set formed by these uncontrolled, unexpected and non-deterministic facts are labeled in literature as exogenous events. Hence, there must recovery strategies that enable agents with an adaptation ability insomuch they can overcome a failure as soon (Ghallab et al., 2014).

Basically, there are two types of recovery strategies and they differ from their results (final plan). In order to illustrate the difference between the replanning results and repairing strategies, after the occurrence of a single failure, consider the conditions presented in Table 1.

Let the failure be the impossibility of executing action α_3 because of some exogenous events. The replanning strategy provides a new plan replacing the suffix $(\alpha_{3'}, \ldots, \alpha_{m'})$ of the initial plan, starting from action α_3, by a new sequence of actions $\alpha'_3, \ldots, \alpha'_m$. Here, nothing can be assumed about the final plan length, which can be bigger or smaller than the initial one. On the other hand, the repairing strategy tries to return the environment state to the expected conditions that support the execution of α_3. Thus, a new item β, that can be either a single action or a sequence of actions, after α_2 and then preserves the suffix $\alpha_3, \ldots, \alpha_m$. In this case, the final plan is greater than the initial one.

Table 1: Difference in results of recovery strategies.

Strategy	Initial plan	Final plan
Replanning	$[\alpha_1, \ldots, \alpha_m]$	$[\alpha_1, \alpha_2, \alpha_{3'}, \ldots, \alpha_{m'}]$
Repairing	$[\alpha_1, \ldots, \alpha_m]$	$[\alpha_1, \alpha_2, \beta, \alpha_3, \ldots, \alpha_m]$

In (Komenda et al., 2014), authors propose a multi-agent plan repair in dynamic environments where a failure is caused by a state perturbation or by an action removal from the plan. Three algorithms are described: back on track (BoT), lazy repair (LR) and repeated lazy repair (RLR).

The Hierarchical Iterative Plan Repair (HIPR) approach, proposed by (Mohalik et al., 2018), combines an architecture and an algorithm to support hierarchical agent teams to replan after a hazard occurrence. Agents try to repair its current plan locally or sent a signal and try to recovery from failures related to preconditions or actions.

The DRA* is an extension of A* algorithm suited for the repairing of sequential plans. The goal-set modifications and actions' costs updates are addressed (Gouidis et al., 2018).

In (Cashmore et al., 2019), authors tackled the problem of replanning for robots using temporal planning problem. Actions have well-defined duration and during their execution it is likely to replan in order to recover a failure or to avoid wasting resources, such as time and battery.

3 PLAN RECOVERY PROCESS

The plan recovery process proposed in this work combines three MAP dimensions related to dynamic environments: planning, coordination and execution. This combination was motivated by the literature review that highlighted an important conclusion: the need to investigate planning and execution in dynamic environments where exogenous events occur and render plans useless (Torreño et al., 2017).

In the proposed process, there are two types of entities. The coordinator is responsible for handling a

pair of files that represents planning domain and problem. Those files are described according to Multi-Agent Planning Domain Definition Language (MA-PDDL). Then, the coordinator searches for a planning problem solution. This initial (and centralized) plan is transformed to a set of single-agent plans. In such plans, actions are scheduled to a common step whether they can be carried out simultaneously by their executors. Then, these plans are sent to the coordinator to start a monitoring loop.

The second type is defined as agents that play planning and executing roles. Therefore, each agent has autonomy to run a deliberation process, whenever it needs. Agents have the commitment of executing the planned actions. Moreover, agents also performs coordination activities to guarantee an environment free of conflicts.

Regarding these premises, the dimensions are handled in staggered solution when agents can try different strategies regarding their capabilities. The process design can be summarized as a three-phase sequence that agents first try to recover from a failure using a local planning. If in this phase is impossible to find a solution, the agent that detected the problem interact with other agents asking for help. Whether some agents return positive answers, the caller will compare the solutions and choose the best (plan with the smallest number of actions) and coordinate with all agents the new execution condition. Otherwise, the caller agent triggers a centralized planning process that is performed by a coordinator agent.

The plan recovery process is presented in Figure 1, where the pipeline of each entity type is described in individual lanes. The process activities are detailed in Sections 3.1-3.11 highlighted by section numbers.

3.1 Problem Instance

The planning domain and problem files are parsed to identify the initial state, goals and operators. Then the available actions are computed and the literals that are updated in action's effects are listed. Those literals are special because they define the search space (relevant facts). The other literals are rigid facts because they are not affected by action results. Both classification are important since only relevant facts must remain after an encoding phase with the purpose of reducing the search space to be explored in the planning phase.

3.2 Centralized Planning

The first planning activity is carried out centralized by the coordinator. In this step, agents are considered as resources. In addition, to compute the plan, this activity is also important to select agents to be committed to execution.

In this sense, the centralized planning provides a solution that minimizes the number of actions required to turn the environment initial state to the goal state. Thus, only the executors of those actions are granted to join the next activities. In case the planning problem has no solution, no execution is triggered and no further activities are performed.

3.3 Plan Coordination

After the definition of the initial plan, the coordinator starts to build the multi-agent plan ρ. First, each agent action is separated in individual lists. Then, a loop is started where the first action of each list is checked about the possibility to be carried out simultaneously from the initial state I. The actions that satisfy the conditions are popped from their agents' list. Otherwise, an empty action (idle state) is defined to the respective executor. Those actions are placed in multi-agent plan and simulated to compute the next expected environment state. The loop finishes when every action of the initial plan is added to ρ.

3.4 Sending Problem and Plans

As soon as the multi-agent plan is defined, coordinator sends the single-agent plans (ρ matrix rows) to their owners. Moreover, it sends a fragment of the planning problem that is formed only by the initial state, goals and available actions of each agent. Thus, information privacy is kept because no agent knows about other agents' capabilities.

3.5 Environment Monitoring

At this point, coordinator starts to monitor the environment with a double concern. First, it controls the plan execution of every agent by receiving messages when they finish their tasks. When there are no more actions to be performed, the coordinator runs its second verification, namely, goal satisfaction. This is an important activity because of the possibility of exogenous events that impair the plan execution leading to failures. If the coordinator detects a problem, it starts a new centralized planning and follows with the next pipeline activities.

3.6 Plan or Coordination Message

The first activity in the agent's pipeline is receiving message about plan and problem. Now, each agent knows its sequence of actions, that was checked and

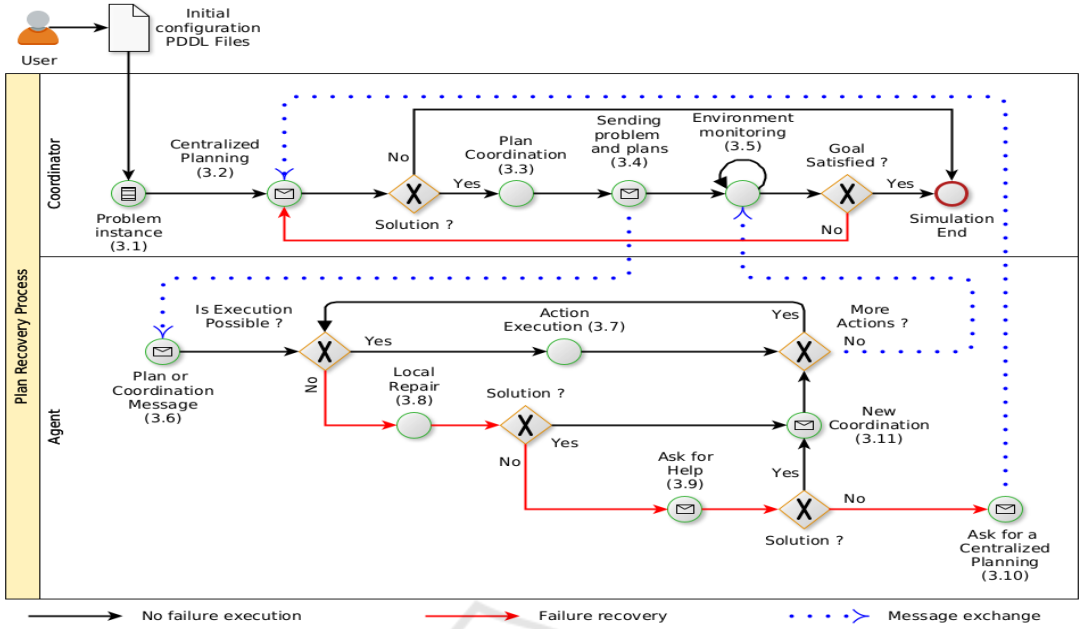


Figure 1: Plan recovery process.

scheduled by to coordinator to provide an execution phase free of conflicts.

This activity can also be triggered when another agent needed to updated its plan (recovery) and then send a message to inform its efforts (number of actions) to overcome a failure. Hence, all the receivers adjust their plans, adding waiting steps (empty actions), as a new coordination phase. We highlight that the coordinator is not warned by the sender because it only needs to control the end of agents' plans.

3.7 Action Execution

The execution phase starts with agents' evaluations about the conditions to run their actions. Each agent analyses the current environment state (s_t) and verifies the possibility to execute the next action. When the preconditions are held, the agent executes the action α, turning s_t to s_{t+1} according to Equation 1, and returns to the evaluation step.

After the execution of the last action, agents send a message to the coordinator to inform that the tasks are done. At the evaluation step, the agent may detect a failure when its next action can not be carried out because of error in the preconditions. Thus, it starts the recovery process.

3.8 Local Repair

The first step of the plan recovery process is performed by the agent that has just detected the failure. Then, it starts a local repair activity. The agent applies the repairing strategy and tries to find a solution that leads the environment state to a condition where the preconditions of the failed actions are satisfied.

If the agent finds a possible solution, it updates its own plan by adding the actions in the beginning of its list, keeping the suffix of the plan from the failed action. In the sequence, agent informs other executors that it needs to run more actions to bring the environment to expected state. However, it is likely that the agent does not find a solution because of a lack of capabilities. Therefore, the next attempt is to ask other agents for help.

3.9 Ask for Help

When an agent asks for help, it shares the conditions the environment state needs to satisfy to guarantee the execution of the failed action. As soon as other agents receive the message, they try a local repair to send back the results. In order to keep the information privacy, agents only share the number of actions they need to recover, instead of sharing the actions themselves.

When just one agent returns a positive answer, this is the solution. However, in the presence of two or more answers, the agent evaluates the possibilities and chooses the best solution considering the smallest number of actions. Then, the selected executor is warned to update its plan by adding the solution, while other agents just receive a coordination message

about the numbers of actions that the chosen agent needs to perform.

3.10 Ask for a Centralized Planning

When the previous phases (Local Repair and Ask for Help) fail in finding a solution, the next activity of the agent that detected the failure is to send a message asking the coordinator for a centralized planning. While the earlier attempts applied the repairing strategy, now the solution is tried through replanning.

As soon as the coordinator receives the message, it runs a centralized planning. But different from the first round, the coordinator plans from the current state rather than the initial one. If it finds a solution to reach the goal from that state, it follows the pipeline (Plan Coordination - Sending Problem and Plans - Environment Monitoring), otherwise no further activity is carried out.

3.11 New Coordination

Agents may receive messages about a new coordination phase. These messages are sent in two conditions. First, when one agent runs a local repair activity and finds a solution. Then, the others agents needs to adjust their plan regarding that new solution. Second, a local repair fails, but after asking for help, the agent receives one positive answer. In this case, all agents, but the chosen one, updates their plans to wait for the execution of that solution.

4 EXPERIMENTS

In this section, we detail the experiment setup (Section 4.1) and discuss the results (Section 4.2).

4.1 Experiment Setup

In order to evaluate the proposed plan recovery process, we used open source software to build a simulation tool. As a solution for the parser and planning issues (Sections 3.1, 3.2, 3.8 and 3.9), we decided to use the PDDL4J[1] JAVA library (Pellier and Fiorino, 2018). Regarding to simulation engine, we chose the Repast Symphony[2] platform for supporting agent-based modeling and simulation (North et al., 2013). The simulation tool is available in an repository[3].

[1]https://github.com/pellierd/pddl4j

[2]https://repast.github.io/

[3]https://gitlab.com/publicrepo/lcmap-de

The case studies applied in the experiments were based on the domains and problems used in the Competition of Distributed and Multi-agent Planners (CoDMAP), which was carried out together with the workshop on Distributed and Multi-agent Planning (DMAP) at the ICAPS 2015 (cod, 2015). Regarding the most used case studies described in the related work, the domains chosen from CoDMAP were satellite, logistics and taxi.

The experiments were carried out under multiple conditions. The failure probabilities varied from 0.1 to 0.9 following a 0.1 step. Each case study had three different configurations which were simulated 30 times. The configurations were selected according to the ratio of public actions with the purpose of simulating problems with different levels of agent coupling. Under those conditions, the experiments were ran by 2430 simulations. The setup description regarding the number of agents, goals and actions in each domain, is summarized in Table 2, where values in cells stand for the minimum and maximum values.

Table 2: Setup description.

Domain	Agents	Goals	Actions	Public (%)
Satellite	3;5	6;10	497;1473	26.2;40.4
Logistics	3;5	6;10	78;308	61.5;62.3
Taxi	4;7	4;7	28;126	100

We carried all experiments out in a single computer with a Intel Core i7-10510U CPU and 16 GB RAM. The operational system was Ubuntu 20.04.1 LTS 64-bit.

4.2 Discussion

The plan recovery process was evaluated regarding three metrics: planning time, final plan length and message exchange. The case studies were classified into three groups according to the agents' coupling level. The problems from the satellite, logistics and taxi were labeled as loosely, intermediate and tightly coupled domains, respectively. The results of each group are discussed individually and a global evaluation is presented.

The first important step towards the evaluation of the results is the definition of how many times each recovery activity (local repair, ask for help and centralized planning) was performed. The information about recovery activities, planning time, final plan length and message is presented in Figures 2 to 3.

Regarding the loosely-coupled domain simulations, the recovery activity was restricted to local repair (Figure 2(a)). This behavior is justified by the fact that agents carry out, at most, public actions. The highest level of coupling is 40.2% (Table 2), hence,

agents do not depend on or affect other agents. Thus, agents do not need to interact to solve failures. Therefore, the motivation hypothesis that agents' autonomy in performing local repair is better explored in environments with low levels of interaction is accepted.

Regarding the intermediate-coupled domain simulations, agents could not solve the failures by using the local repair activities. Indeed, they need to interact asking for help (Section 3.9). Sometimes, they also need to request for the centralized planning. Those recovery strategies are shown in Figure 2. The reason of that justifies this different behavior in the logistic domain simulations is inherited from the set of available actions. Thus, agents do not have all the capabilities that are required to solve a problem. Hence, they need to cooperate towards the search for a solution.

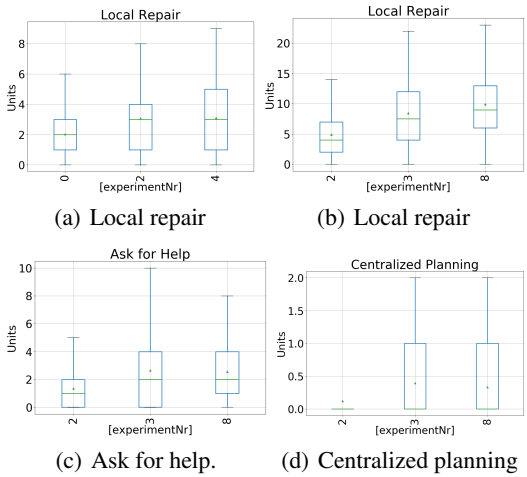
Regarding the tightly-coupled domain simulations, agents needed to request more often for a centralized planning, as shown in Figure 3. The ratio of public actions in the set of available actions was 100% of public actions (Table 2). Thus, every action either depends on or affects other actions. Therefore, it was expected that the solution for the failure would only be found by a centralized planning activity where all actions were available in a common process.

The execution of higher recovery strategies increases from loosely to tightly domains. The analysis of the averages of the results from each strategy is detailed in Table 3 by domains, which demonstrates that: (i) the frequency of local repair calls in tightly-coupled domains is 3.11× bigger that in loosely; (ii) ask for help and centralized planning activities are carried out 1.73× and 8.68× more often in tightly than in intermediate domains. Therefore, the coupling level among agents increases the complexity of the recovery process.

Table 3: Strategies calls (means) by domains.

Strategy	Loosely	Intermediate	Tightly
Local Repair	2.73	7.76	8.5
Ask for Help	0	2.18	3.78
Centralized Planning	0	0.28	2.43

Further conclusions can be drawn from the evaluation of the metrics shown in Figures 4. Regarding the final plan length (Figure 4(a)), although the intermediate-coupled domain simulations have the complexity levels (number of public actions) than satellite domains, they presented final plans with a higher number of actions. The logistics domain simulation applied repairing (local repair and ask for help) more than other simulations, and that is the reason for bigger plans. The main drawback of this strategy derives from the fact that repairing tends to



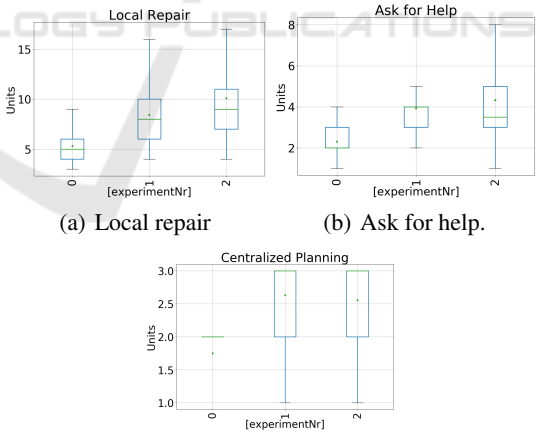
(a) Local repair (b) Local repair

(c) Ask for help. (d) Centralized planning

Figure 2: Loosely and intermediate-coupled domains.

build bigger plans. The taxi domains highlighted plans with fewer actions because of replanning strategy (Komenda et al., 2014).

Regarding the planning time analysis (Figure 4(b)), loosely domains showed irrelevant and small values when compared to the other groups where only local repair was needed and repairing strategy tends to be faster. The final planning time in logistics was higher than the values of taxi domains since plans were bigger. Hence, more actions were likely to fail and more recovery activities had to be performed.



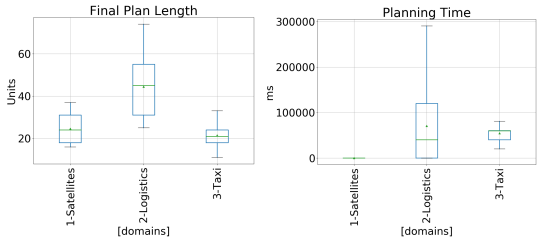
(a) Local repair (b) Ask for help.

(c) Centralized planning

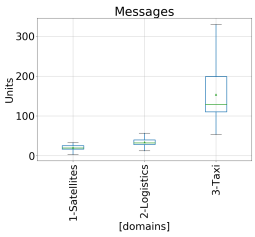
Figure 3: Recovery in tightly-coupled domains.

The message exchange (Figure 4(c)) highlights the interaction of agents after facing a failure. Since the loosely-coupled domains handle the problem with local repair strategy, the agents only need to exchange message to provide a new execution coordination. However, in the intermediate and tightly-coupled domains, the amount of messages to be exchanged tends

to be higher than the first one because agents needs to ask for help and request a centralized planning with more messages sent. Therefore, this is another evidence that the hypothesis that agents' autonomy is better explored in environments with low levels of interaction can be accepted.



(a) Final plan length. (b) Planning time.



(c) Message exchange.

Figure 4: Global evaluation.

5 CONCLUSION

In this work, a plan recovery process to dynamic environments affected by exogenous events was proposed. The process differs from related work because it combines the replanning and repairing strategies providing an staggered solution that is formed by local repair, ask for help and centralized planning.

In order to investigate the process performance, a MAP simulation tool was developed to run different problems. We evaluated case studies with different levels of interaction among agents. We accepted the motivation hypothesis that agents' autonomy in performing local repair is better explored in environments with low levels of interaction. Moreover, we showed that the coupling level among agents, inherited from the public actions ratio, increases the complexity of the recovery and the metrics related to planning time, final length and message exchange.

The contributions of this work are: a three-phase plan recovery process, a simulation tool, and a statistical evaluation method to MAP in dynamic environments. The study of other failures caused by the agents removal and the use of distributed approach to agents' coordination are suggestions of future works.

ACKNOWLEDGEMENTS

Prof. C. G. Ralha thanks the support received from the Brazilian National Council for Scientific and Technological Development (CNPq) for the research grant in Computer Science number 311301/2018-5.

REFERENCES

(2015). Competition of Distributed and Multiagent Planners (CoDMAP). Available at http://agents.fel.cvut.cz/codmap/, Accessed on: 2020-02-05.

Borrajo, D. and Fernández, S. (2019). Efficient approaches for multi-agent planning. *Knowledge and Information Systems*, 58:425–479.

Brafman, R. I. and Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 28–35. AAAI Press.

Cashmore, M., Coles, A., Cserna, B., Karpas, E., Magazzeni, D., and Ruml, W. (2019). Replanning for situated robots. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 665–673. AAAI Press.

Chrpa, L., Gemrot, J., and Pilát, M. (2020). Planning and acting with non-deterministic events: Navigating between safe states. In *Proceedings of the* 34^{th} *Conference on Artificial Intelligence*, pages 9802–9809. AAAI Press.

Ghallab, M., Nau, D., and Traverso, P. (2014). The actor's view of automated planning and acting: a position paper. *Artificial Intelligence*, 208:1–17.

Gouidis, F., Patkos, T., Flouris, G., and Plexousakis, D. (2018). Dynamic repairing A*: a plan-repairing algorithm for dynamic domains. In *Proceedings of the* 10^{th} *International Conference on Agents and Artificial Intelligence (ICAART)*, pages 363–370.

Komenda, A., Novák, P., and Pchouček, M. (2014). Domain-independent multi-agent plan repair. *Journal of Network and Computer Applications*, 37:76–88.

Mohalik, S. K., Jayaraman, M. B., Badrinath, R., and Feljan, A. V. (2018). HIPR: an architecture for iterative plan repair in hierarchical multi-agent systems. *Journal of Computers*, 13(3):351–359.

North, M. J., Collier, N. T., Ozik, J., Tatara, E. R., Macal, C. M., Bragen, M., and Sydelko, P. (2013). Complex adaptive systems modeling with repast simphony. *Complex Adaptive Systems Modeling*.

Pellier, D. and Fiorino, H. (2018). PDDL4J: a planning domain description library for java. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(1):143–176.

Torreño, A., Onaindia, E., Komenda, A., and Štolba, M. (2017). Cooperative Multi-Agent Planning: A Survey. *ACM Computing Surveys*, pages 1–32.

Weiss, G., editor (2013). *Multiagent Systems*. The MIT Press, 2^{nd} edition.