Carbó, J., Patricio, M.A., Molina, J.M. (2019). Merging plans with incomplete knowledge about actions and goals through an agent-based reputation system. *Expert Systems with Applications,* 115, pp. 403-411.

DOI: 10.1016/j.eswa.2018.07.062

Merging plans with incomplete knowledge about actions and goals through an agent-based reputation system

Javier Carbo, Miguel A. Patricio, Jose M. Molina[a,b,c]

[a] *javier.carbo@uc3m.es, Depto Informatica, Univ. Carlos III de Madrid, Av Universidad 30, Campus Leganes, 28911 Madrid, Spain*
[b] *miguelangel.patricio@uc3m.es, Depto Informatica, Univ. Carlos III de Madrid, Av Universidad Carlos III, 22, Campus Colmenarejo, 28270 Madrid, Spain*
[c] *josemanuel.molina@uc3m.es, Depto Informatica, Univ. Carlos III de Madrid, Av Universidad Carlos III, 22, Campus Colmenarejo, 28270 Madrid, Spain*

Abstract

In this paper, we propose and compare alternative ways to merge the plans formed by sequences of actions of unknown similarities between goals and actions. Plans are formed by actions and executed by several operator agents, which cooperate through recommendations. The operator agents apply the plan actions over passive elements (which we call node agents) that require additional future executions of another plan after some time of use. Such ignorance of the similarities between the plan actions and the goals justifies the use of a distributed recommendation system, which will provide a useful plan to apply for a certain goal to a given operator agent. Then, this plan is generated from the known results of previous executions of different plans by other operator agents. Here, we provide the general framework of execution (the agent system), and different merging algorithms are applied to this problem.

Keywords: Recommendation, Reputation, Trust, Agents, Planning

1. Introduction

The term multi-agent planning has been used in the literature for two types of problems. On the one hand, it has been interpreted as the problem to find plans for a group of agents (Ephrati & Rosenschein, 1993) (Rosenschein, 1982). In such an approach, autonomous agents intend to coordinate their actions to satisfy different goals of each agent (Georgeff, 1983) (Muscettola &

Smith, 1987). The drawback of this problem definition is that autonomous agents, who pursue their particular goals by definition, may adopt a non-cooperative attitude if they do not have personal incentives to cooperate.

On the other hand, when the agents share a common collective goal, multi-agent planning is alternatively considered a "divide et impera" problem. Such a common goal, which is typically complex, is split into subproblems among the agents. Then, each agent solves a subproblem, and all corresponding subsolutions that are generated in parallel by the agents must be merged into a unified solution (Wilkins & Myers, 1998) (Ephrati & Rosenschein, 1993). In some cases, there is no partition of the problem, and agents only independently apply different planners to the same problem (as in scheduling for transportation systems (Fischer et al., 1995)) to more quickly converge to a solution.

1.1. General context and motivation

The problem of merging plans with no previous partition follows a similar approach to ours: each agent defines its own plans in parallel in a distributed and independent manner until a merging algorithm is applied to merge the plans into a single, unified, and supposedly better plan. This merging algorithm must identify key elements of the plans that provide improvements, incompatibilities, efficiencies and other relevant features in the method to solve the problem.

This analysis and search of the key elements of plans to be merged by autonomous agents acting as independent planners can become extremely complex. With this goal, (Bruce & Newman, 1978) proposed a structured model of actions, intentions, beliefs and states to be represented in the plans. (Rosenschein, 1982) proposed a formalization of plans to detect potential conflicts. Similarly, (Georgeff, 1983) suggested the definition of correctness and execution conditions to be satisfied in a plan. (Shieber, 1985) defined a temporal logic system to specify restrictions of plans that (Yang et al., 1992) and (Foulser et al., 1992) used to ensure an efficient merge of alternative plans that pursued a common goal. A more recent logic system for multi-agent planning was proposed by (de Weerdt et al., 2003). (Elkawkagy & Biundo, 2011) and (Brahimi et al., 2014) use preprocessing steps to generate a hierarchy of subplans that agents compute in an independent way. Therefore, for them merging has a different meaning, it is just a composing task of hierarchically pre-ordered subplans. This is not our case, since we do not assume a previously known hierarchy of subplans, we assume the knowledge

of the number and similarities of actions (not subplans) that may compose a plan. How they have to be composed is the goal of our planning computing, instead of being previously known. And (Elkawkagy & Biundo, 2011) and (Brahimi et al., 2014) merge parts of the final plan, not complete alternatives of the final plan as we do. A very recent work by (Borrajo & Fernández, 2018) provides an interesting view of distributed planning with autonomous agents focused on protecting privacy.

The use of execution conditions, logic or temporal restrictions to combine plans is based on the intrinsic characteristics of the plan and dependencies of the actions that form a plan, which are typically domain-dependent. This drawback is often ignored by assuming that they are previously known, although this cannot be the case in real problems. However, plans can also be merged using an external evaluation of the actions (instead of previously known restrictions), as the following proposals and our study suggest:

- Partial global planning (Decker & Lesser, 1992): each agent iteratively proposes an incremental prototype of a global plan: when an agent receives the global plan, it combines such a plan with its plan to create an improved new global plan (e.g., removing redundancies). This improvement is proposed to the other agents so that they can accept, modify or reject it.

- An approach based on an external estimation of the quality of a plan from (Ephrati & Rosenschein, 1993): the combination of a set of plans is suggested to detect redundant actions using the A* search algorithm and a suitable cost heuristic. A process of joint aggregation was also suggested, where the agents form an improved global plan voting joint actions, and the votes play the role of the cost in the heuristic (Ephrati et al., 1995).

Similar to our proposal, these approaches merge plans using external evaluations of the plans instead of assumed known previous restrictions. However, the improvement of these two merging proposals is obtained from an internal analysis that uses complex knowledge representation formalisms about plans. Therefore, they are domain-dependent and similar to the logical-temporal restrictions and execution conditions of the most extended approaches regarding the problem of combining plans. In contrast, the originality of our approach relies on the method of such external evaluation. We evaluate plans in a truly domain-independent manner according to the results

obtained by the past execution of the plans to be combined and not a rich (and complex) deliberation about the compatibility, efficiency and redundancies of the actions that form the plan. Then, the evaluation is extrinsic of the actual composition of the plans to be combined. We can say that the underlying inspiration of our method is closer to a neural/subsymbolic approach instead of a deliberative/symbolic approach: the valuation of the plan is the output of a black box with two inputs–own direct experience and recommendations. These inputs are two indirect methods to estimate the quality of the plan, which are not based on the expression of the positive/negative interrelationships of the actions that form the plans to be combined. Therefore, our approach does not rely on the previous, certain and complete knowledge of domain.

This idea is strongly inspired by our previous works on recommendation systems, specifically the agent-based reputation management (Carbo et al., 2003) Gomez et al. (2007) (Carbo & Molina, 2010). In these contributions, autonomous agents use the ability to dynamically combine past experiences to face new situations and select and mix them to improve the expected results.

To provide this ability to the agents, such past experiences must be represented in a context-sensible manner, which includes an integration mechanism based on a utility control of these experiences. In simple terms, they must be able to distinguish (contextualize) which of these experiences are similar (it is in fact an estimation) to one another and to the new situation. This problem has been addressed by planning systems that use case-based reasoning, e.g., (Veloso, 1994) (Redmond, 1990) (Goel et al., 1994) (PLAZA & MCGINTY, 2005). Therefore, solving the problem of combining past experiences requires using context information; in our case, experiences are previously executed plans. This contextual information is the estimation of the applicability of a particular plan to the actual situation faced by the agent, even if this estimation is roughly computed by indirect sources of information (the aforementioned two inputs of the black box).

1.2. Assumptions

We must accept several assumptions to apply our approach, which drive us to reflect some unrealistic requirements. For example, all plans are executed in a completely deterministic world. We assume that each action always produces the same consequence (no given uncertainty), and the execution of plan actions is time-independent, i.e., identical results are produced

any time the plan is executed.

We must also assume the next statements about the nature of autonomous agents that perform the plan combinations:

- They are self-interested and mainly focus on reaching their own goals.
- They are consistent; the specification of their internal state must be an accurate representation of the external world.
- The agents have no conflict among themselves; they are cooperative and not competitive.
- The agents are benevolent, i.e., they have a predisposition to help each other.
- The agents are honest; the information that they share with others is a precise representation of their own internal state.

In planning systems, plans are often considered partially or completely ordered according to given restrictions over their actions, where partial ordering allows that the correct plans are open to different sequences of actions. Completely ordered plans denote a unique solution in the form of a particular sequence of actions. Our approach to the problem assumes completely ordered plans, and we assume that there is only one correct action in each step of the plan. We also assume that the actions are independent: the actions that should be executed before and after do not alter the suitability of a particular action ordered in the correct step.

We assume that the final combined plan is executed by an active 'operator', which can have different expertise, over a 'node' or physical passive element that requires the execution of successive plans with a frequency, which depends on the success of the previously executed plan. For example, a major city can consider the infrastructure elements of the city that deteriorate with time as nodes, and they require maintenance over time. In our urban example, the operators can be human resources that perform maintenance tasks (plans) of the infrastructure elements, which require an expertise level and a correct sequence of operations (plan actions) to succeed. A success will imply a long time period until the next maintenance task is required.

To represent different types of actions that may form a plan, we consider that some nodes have certain similarities (i.e., they belong to the same "type") in two different grades; then, minor differences among the nodes are

represented as nodes of the same type but different subtypes. An operator that accurately notices such differences will select the correct action to perform on the node. Therefore, we define an action by 3 values that represent its suitability according to the node to be applied:

- corresponding type of the node to be executed.
- corresponding subtype of the node to be executed.
- corresponding time (sequence order of the plan) to be executed.

Furthermore, we assume that the types and subtypes of the nodes are public and previously known, and there is a limited (low) number of them. We assume that the agents may compare different nodes to obtain a similarity estimation, which depends on their type and subtype. Then, they can compare the similarity of a node, over which a plan was executed, with the current node, which requires a new combined plan to be executed. Thus, the suitability of an action over a given node can be computed according to the similarity of the type and subtype of that node with the type and subtype of the node to which the action belongs. If a plan is effectively suitable, the node will reduce the expected time before the successive plan must be executed over that node.

Additionally, we assume that the number of steps of a plan is public, previously known and limited (small). Hence, the suitability of an action in a given plan step can again be computed according to where such action in the plan is placed (when such action is placed near or at the correct plan step).

Finally, we assume that weights are associated to the relative relevance of these three suitability criteria (type, subtype of the node and time of execution); thus, the final suitability of the plan is computed as their weighted sum.

2. Defining the agent system

Both types of agents (node and operator agents) are implemented in JADEX (Braubach et al., 2004), which is a JAVA-based academic agent platform that complies with the IEEE-FIPA standard (fip, 1997) and facilitates the implementation of a deliberative approach of agent reasoning structured into three levels of knowledge: beliefs, desires and intentions (in advance,

BDI). Many other agent platforms implement such standard and BDI reasoning (Kravari & Bassiliades, 2015). Nonetheless, this survey from 2015 explicitly shows that JADEX provides a higher level of robustness and performance than 24 other alternative agent platforms. More recent and claim-to-be-more-efficient platforms as SPADE [1] can be considered, but JADEX has been largely successfully applied in research, teaching and industrial applications, and we have previous expertise developing applications with it. Furthermore, we published a deeper explanation of how agents internally used beliefs, desires and intentions in this problem in (Carbo et al., 2016).

- The beliefs of agents and content of the exchanged messages (concepts, actions and predicates of an ad hoc ontology) are JAVA classes that represent such knowledge. The concepts in the ontology are as follows: node identifier; node type; operation to be performed (their ordered sequence forms a plan), which is defined by the operation type; node type and subtype to be applied; and expertise, which represents the ability of the operator that executes the operation plans. The beliefs of the node agents include the current operator agent and the expected time until a new operation plan must be executed over this node. Operator agents have the next beliefs: availability of the operator, which are the currently operated nodes; type of specialized node; its expertise; its last operation plans over each type of node; and final operator agents that act as recommenders (identifiers and expertise) and their corresponding last recommendation.

- The intentions of agents correspond to the actions performed by an operator or node agent that looks for the satisfaction of a given goal (conceptually, an instantiated desire). Typically, they involve receiving a message, accessing the content of the message and internal beliefs of the agent, performing computations over these data, and building and sending a response message to another node/operator agent.

- The desires of agents are implemented in JADEX as goals that can be fired by external events (reception of a message) or internal events (specific conditions over the beliefs of the agents are satisfied). Such goals in our agent system implementation consists of the complete ex-

[1] https://pypi.org/project/SPADE/

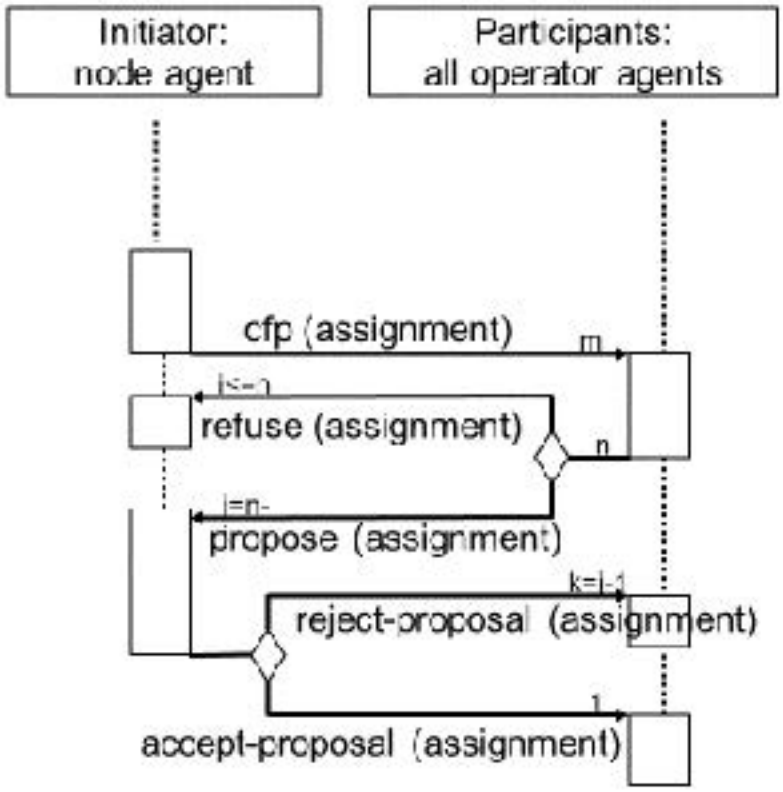

Figure 1: CFP interaction protocol between a node and an operator agent

ecution of the corresponding role (initiator/participant in FIPA terms) in a protocol.

The messages between the operator and the node agents are instances of pre-defined IEEE-FIPA protocols. Therefore, we have the next sequence of protocols in a typical iteration cycle of our agent system:

- The first protocol to launch is a call for proposal protocol (see Figure 1), where the node agent acts as an initiator, and the operator agent acts as a participant. This protocol is associated with the goal of a node to become assigned to an operator agent, and it is fired by an internal event of a node agent because its beliefs show no current operator agent.

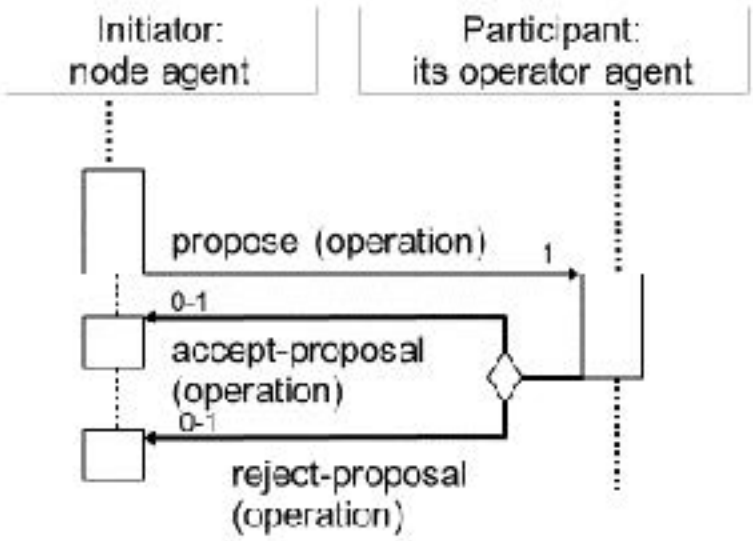

Figure 2: Propose interaction protocol between a node and an operator agent

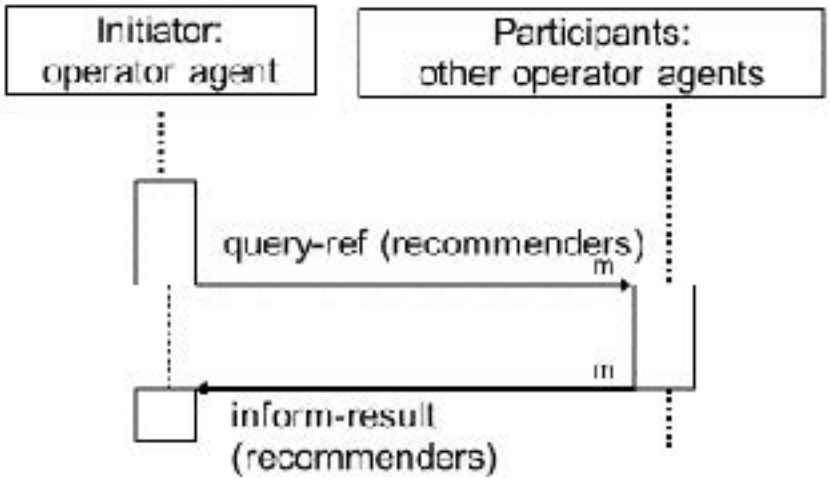

Figure 3: Query recommender interaction protocol between two operator agents

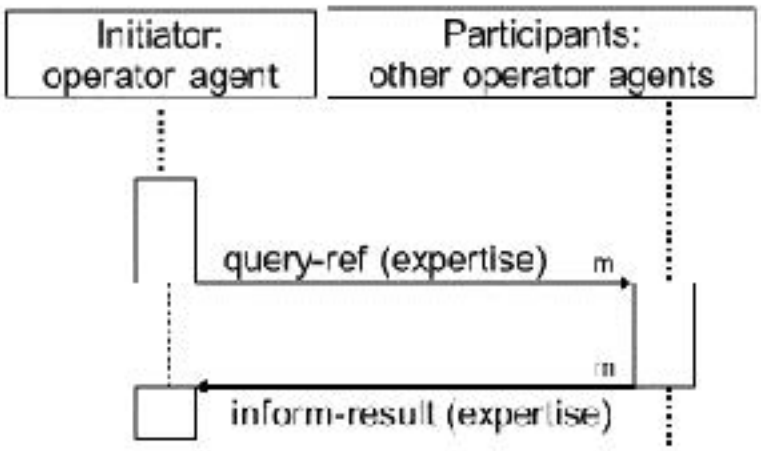

Figure 4: Query expertise interaction protocol between two operator agents

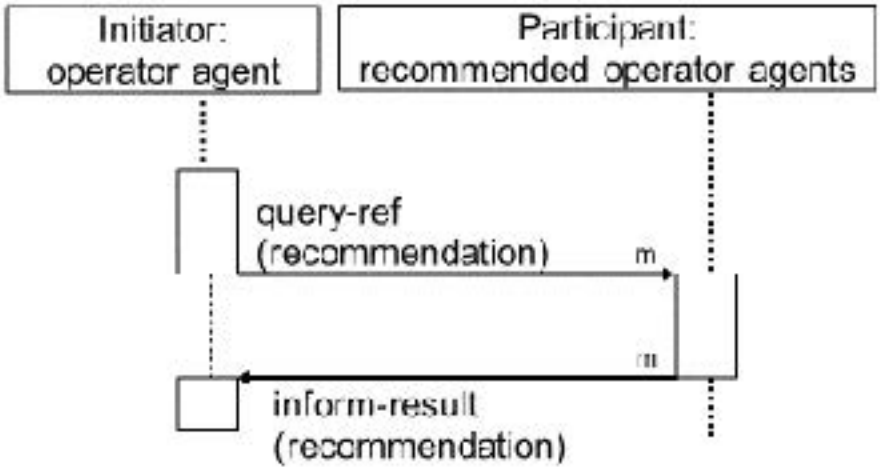

Figure 5: Query recommendation interaction protocol between two operator agents

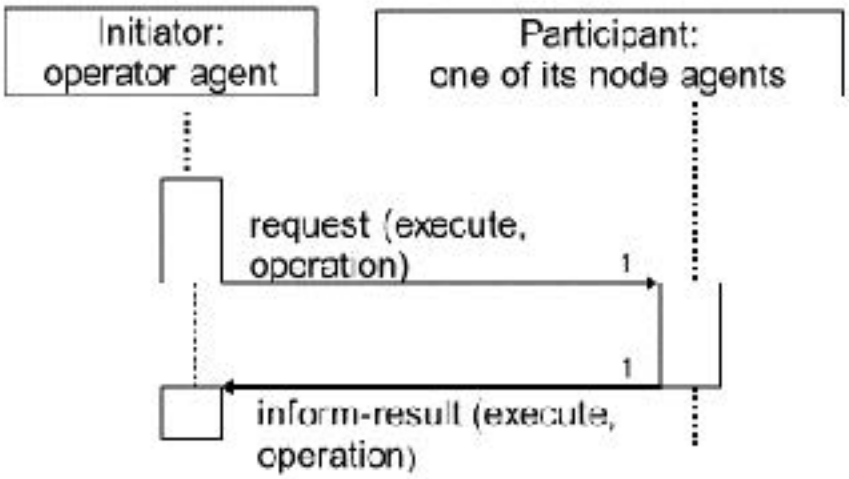

Figure 6: Request plan execution interaction protocol between an operator and a node agent

- Next, a proposed interaction protocol is launched with the node agent as the initiator and the operator agent as the participant; the associated goal is fired by an internal event when the node agent observes that the belief that corresponds to the expected time of a new required operation plan points out such circumstance (see Figure 2).

- Then, a query interaction protocol occurs with an operator agent as the initiator and another agent as the participant of the query protocol. This query protocol identifies the recommenders that are specialized in a particular type of node. It is fired by an internal event of an operator agent because its beliefs show that it must generate an operation plan for a node agent of a given type, and it has no recommenders of this node type among its beliefs (see Figure 3).

- Again, another query protocol is launched where an operator agent that acts as an initiator asks about its expertise in a type of node to another operator, which acts as a participant in the protocol, as a potential recommender. It is fired by an internal event of an operator agent because its beliefs show that it must generate an operation plan for a node agent, and it must update the expertise of recommenders of this node type in its beliefs (see Figure 4).

- Additionally, another query protocol initiated by an operator agent asks about the operation plan that another operator agent, which acts as a participant in the protocol, recommends for a particular type of node. It is fired by an internal event of an operator agent because its beliefs show that it must generate an operation plan for a node agent of a given type, and it must update the recommended operation plan of the recommenders of this node type in its beliefs (see Figure 5).

- Finally, a request protocol is launched by the operator agent as the initiator and a node agent as the participant of the protocol. There, the operator agent requests the node agent to execute an operation plan, which is generated by combining the received recommendations and the last operation plan of such operator agent for that type of node. It is fired by an internal event of an operator agent because its beliefs show that it must generate an operation plan for a node agent of a given type, and all of its beliefs, which correspond to the expertise

and recommended operation plans of this node type, are updated (see Figure 6).

3. Merging Plans

As shown in the description of the agent interaction protocols, the combination of plans will occur immediately before the execution of the last protocol. Specifically, when the operator agent has updated its beliefs, the subsequent knowledge includes the following:

- expertise about the given node type of the operator agents that act as recommenders
- operation plan that the operator agents recommend for the given node type
- previous own expertise about the node type
- previous own operation plan for the node type

Our merging method uses these four inputs to produce a new operation plan, which will be sent to the node agent. The corresponding answer from the node agent will include the time to spend before a new operation plan is required. This time is computed according to the level of success obtained with the operation plan suggested by the operator agent, and it will be used by the operator agent to update its own expertise in such node type.

We will obtain this level of success based on the equations in the Agent Reputation Testbed (ART) platform (Fullam et al., 2005). This platform has been used for experimentation in several publications (LukeTeacy et al., 2007) Yann Krupa & Vercouter (2009) (Diniz Da Costa et al., 2008) (Munoz et al., 2009) and several international competitions in International Conference on Autonomous Agents and Multi Agent Systems (AAMAS).

$$resul = \Phi(\|e - t\|, s) \tag{1}$$

$$s = (1 - E + 1/c) \times \|e - t\| \tag{2}$$

The ART testbed generates the error in a service estimation through a normal distribution centred in the difference between the estimated (noted as e) and the true value of the service (noted as t); the standard deviation s

denotes the noise that avoids an optimal perception of the true value of the service in real life (Equation 1). s takes its value from Equation 2. In this equation, E denotes the expertise of the agent in providing such service, c is the cost for the agent to provide the service, i.e., the time invested by the agent (which represents the effort of the agent). Thus, stronger expertise, more invested effort, and larger success of the estimation correspond to less noise applied to the normal distribution.

Similarly, we may compute the difference between the optimal plan and the suggested plan (the node agents will compute such difference). Hence, we assume that such optimal plan is available, plans have a limited number of steps, and the possible operations to be applied in each step is limited and dependent on the type and subtype of the nodes. Therefore, with the number of steps, number of types and subtypes, and weights of making a mistake in the step, all types and subtypes are setup parameters of the agent system.

$$\begin{aligned} \|e - t\| = \sum_{i=1..maxtimestep} & (w_{type} \times \|e_{i,type} - t_{i,type}\| \\ & + w_{subtype} \times \|e_{i,subtype} - t_{i,subtype}\| \\ & + w_{timestep} \times \|e_{i,timestep} - t_{i,timestep}\|) \end{aligned} \tag{3}$$

Then, each operation of an optimal plan is defined by three dimensions: the correct type of the node to apply, correct subtype of the node and correct time step to apply. The difference between an operation of the plan suggested by an operator agent and the optimal operation is computed as a weighted sum of the distance of the node type, subtype and timestep (sequence order), as shown in Equation 3.

$$time = resul \times maxtime \tag{4}$$

When we have the level of success obtained with the suggested plan by the operator agent, the node agent must compute the required time for the next plan execution. It would take its value from Equation 4, where *resul* is the level of success of the suggested plan and *maxtime* is the maximum time that any node may spend without the execution of a new plan (given static value as parameter in the setup of the agent system). Thus, a better level of success of the plan implies less required time to execute a new plan.

$$s = (1 - E_l + 1/E_g) \times \|e - t\| \tag{5}$$

Finally, the node agent must compute the new expertise of the operator agent after the suggested plan is executed. In our design, the concept of expertise has two associated values:

- Global expertise: the number of times that an operator agent has executed plans (regardless of their success) for a given node type. The number of times will increase until a threshold, which is defined as a setup parameter; then, after the number of plan executions, the global expertise is fixed at the maximum value. The global expertise is used instead of the invested time of Equation 2, assuming that all of our operator agents spend identical amount of time in the execution of plans, but operator agents with more global expertise are more efficient in identical time periods.

- Local expertise: the ability in suggesting a plan for a given node type. It is computed as the opposite of the standard deviation s; then, it has a recursive definition, where past local expertise affects the computation of the current local expertise, as s was defined in Equation 5 as dependent of local expertise.

According to these two features of expertise, Equation 2 from the ART testbed becomes redefined in our agent system as Equation 5.

Meanwhile, operator agents must weight the recommended plans from other operator agents to combine them into a new single plan to suggest to the node agent. The reputation of the operator agents in our domain is contextual, i.e., it takes a different value for each node type. It is computed from the global and local expertise of the operator agent using Equation 6, where E_l is the local expertise, E_g is the global expertise, and Thr is the threshold of the global expertise.

$$rep = 1 - (1 - E_l) \times (Thr/E_g) \tag{6}$$

Next, we will show 4 methods to use all of these elements to combine plans. The first method is only a no-merging method used as a benchmark to compare the improvement in level of success of the operator agents that suggest a plan to the node agent by combining plans.

3.1. Merging method 0: no merging and ignoring recommendations

$$newplan = oldplan_j, j = \arg\max_{i=1..n} s_i \tag{7}$$

The operator agent ignores all received recommendations, and it executes the plans that are exclusively updated from its previous execution of plans as shown in Equation 7, where s is the success of the execution of previous plans. This method is used as the benchmark, and the relative benefits of applying the other methods can be measured by the improvements in results obtained with respect the results of this no-cooperation merging method.

3.2. Merging method 1: no merging considering recommendations

$$newplan = recommendedplan_j, j = \arg\max_{i=1..n} rep_i \tag{8}$$

The operator agent select the best plan (according to the computed reputation of the operator agents) among all possible options (recommendations and previous plan for that node type) with no combination as shown in Equation 8, where rep is the reputation of the received recommendation.

3.3. Merging method 2

$$newplan = \arg\max_i \sum_{j=1}^{numrec_{k,i}} rep_{i,j}, k \in [1, numtimesteps] \tag{9}$$

The operator agent builds the new plan by combining the operations with more reputation (inherited by the operator agent that suggests this operation in this plan step in its recommended plan) and popularity (number of times that this operation is suggested by the operator agents). In other words, we sum all reputations of each possible operation for each plan step; thus, repeated operations and operations from operators with good reputation have more options to become the suggested operation in the new plan step. This process is summarized in Equation 9, where *numoper* is the number of different recommended operations for a given time step j, and *numrec* is the number of recommenders that suggest that operation i is applied in time step j.

Table 1: General Parameter Setup

Parameter	Value
Number of node agents	1
Number of operator agents	20
Number of recommenders an operator may ask	20
Number of plan steps	5
Number of types of a node	1
Number of subtypes of a node	2
Weights of errors mis-predicting types, subtypes and operations	1
Initial expertise of operators	1
Number of replacements of merging method 3	1
Generation of previous plans of operator agents	random

3.4. Merging method 3

$$newplan = \begin{cases} best_k & \text{if } best_k \neq worst_k, k \in [1, numtimesteps] \\ \arg\max_i \sum_{j=1}^{numrec_{k,i}} rep_{i,j} & \text{otherwise}, i \in [1, numoper_k) \end{cases} \tag{10}$$

The operator agent modifies the recommended plan from the operator agent with the most reputation. The modification consists of replacing a number of the operations that this best plan according to the reputation shares in the same plan step with the worst plan. These operations are replaced by different suggested operations by other recommended plans (selected in decreasing order of reputation of the operator agent). The number of replacements is fixed as a parameter setup. Therefore, this combination method is similar to evolutive algorithms, where plans are the individuals to be crossed, and the reputation plays the role of a fitness function. This process is summarized in Equation 10, where i goes from 1 to the number of operations recommended for timestep k-1, which is the operation that the best and the worst plans share.

4. Experimentation

Since our merging proposals are based on an external evaluation as section 1 stated, they substantially differ from the classic approach of merging plans, they use logical domain-dependent restrictions and we do not. Therefore, as

Table 2: Parameter Setup of Simulation 1

Parameter	Value
Number of node agents	1
Number of iterations	2
Number of operator agents	20
Number to simulation runs	4 (one for each merging method)

our inputs and assumptions are different, we do not intend to make any comparison with other merging methods in a shared agent framework as (Zaghetto et al., 2017) did with tracking algorithms, since they are of different nature. We also do not intend to test any simulation based on realistic data from any problem domain, since our proposal intends to be generic and domain independent. Instead, we focus on testing the four merging methods according to the relative contribution of the cooperation provided by recommendations and the evolution because of past experience. We define two notably simple simulations, 1 and 2, where the improvement because of the evolution and cooperation is considered isolated, and simulation 3, where both evolution and cooperation jointly occur. We consider the parameter setup in Table 1 for all simulations. All weights of error mis-predicting types, subtypes and operations were set to 1, since different weights introduce the possibility of giving more or less relevance to any of the three categories over the others. Different weights can be useful if we know in advance that in a given domain, the errors in any of the three categories imply more costly consequences. However, we assume no previous knowledge of the domain problem.

First, we consider a notably simple test, noted as simulation 1, with the parameter setup in Table 2. In simulation 1, first, the node agent selects one of 20 operator agents as its operator. Next, the selected operator agent asks for recommendations from the other 19 operators about its previous experience. Finally, the resulting plan is generated by combining the 19 responses and the previous experience of the selected operator (all of which are fake, randomly generated experiences). Because there is only one node agent, the other 19 operator agents do not have real "experience"; they only act as "dumb" recommenders (always sending the same, constant, randomly generated recommendation). Therefore, there is no reason to run more than 2 iterations. Each iteration consists of the execution of all described sequences of protocols with the same set of agents and remaining beliefs.

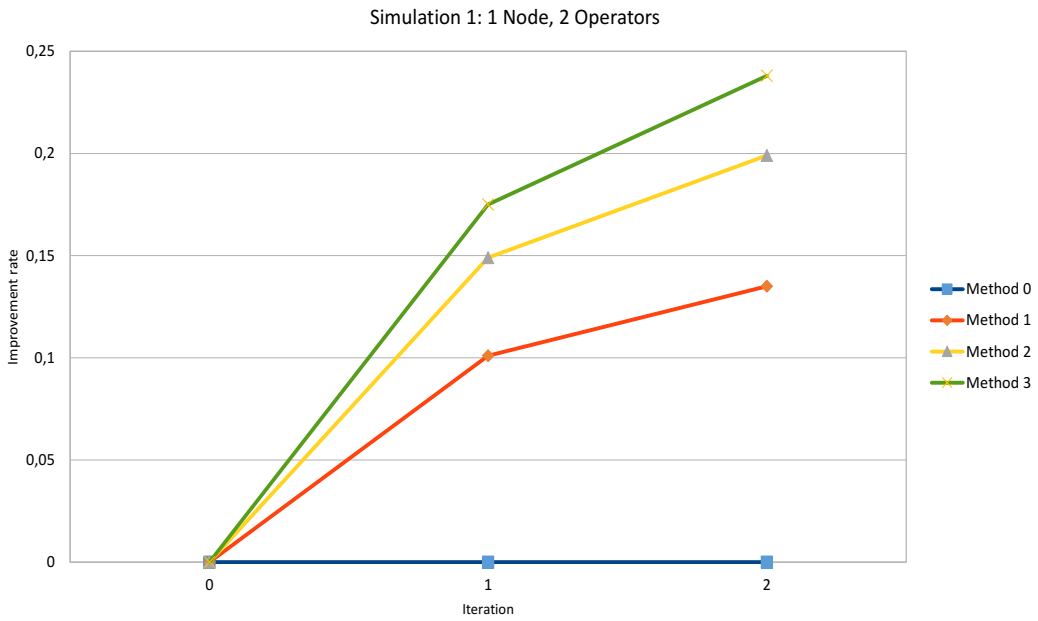

Figure 7: Improvement rate (0-1, Y axis) because of the merging methods (0,1,2,3) with 20 operators and 1 node in 2 iterations (X axis)

Table 3: Parameter Setup of Simulation 2

Parameter	Value
Number of node agents	2
Number of iterations	10
Number of operator agents	2

Simulation 1 yields the results of Figure 7. The vertical axis of Figure 7 shows the improvement of the combined plan of each merging method (approaching the optimal plan), whereas the horizontal axis shows the evolution along 2 iterations. The resulting lines of each merging method appear to confirm the order in which they obtain the best possible plan from the same available information through cooperation (in the form of recommendations) but with no evolution of the available information of recommendations.

Simulation 2 with the initial parameter setup in Table 3 decreases the level of cooperation (because there are 2 operator agents instead of 20), but the information provided by recommendations is not constant or random because it is based on the evaluation of the corresponding plan. Therefore, the recommendation is improved in each iteration. Again, each operator agent is linked to only 1 node agent because the parameter of their availability remains equal to 1, and the two operator agents share their relative success and failures in 10 iterations.

Figure 8 shows the information relative to simulation 2 in both axes, and the lines show the evolution of the average improvement of both operator agents. Here, we observe that the relative improvements are much less significant ($0.05 << 0.25$) when the operator agents combine their plan with only one recommendation (instead of 19 in the previous simulation). Thus, the merging methods have much better performance when they have fewer plans to combine, even when they are not accurate or updated (they were constant in the previous simulation). Additionally, the differences among 'serious' merging methods (methods 1-3 because method 0 cannot be considered a merging method) are notably small in these circumstances.

Finally, we run simulation 3 with the parameter setup in Table 4, 10 node agents and 10 operator agents along 10 iterations to observe the evolution because of the changes in previous experiences and cooperation through accurate recommendations. Each node agent is linked with a single operator agent, and each operator agent asks for recommendations from the other 9 operator agents in each iteration.

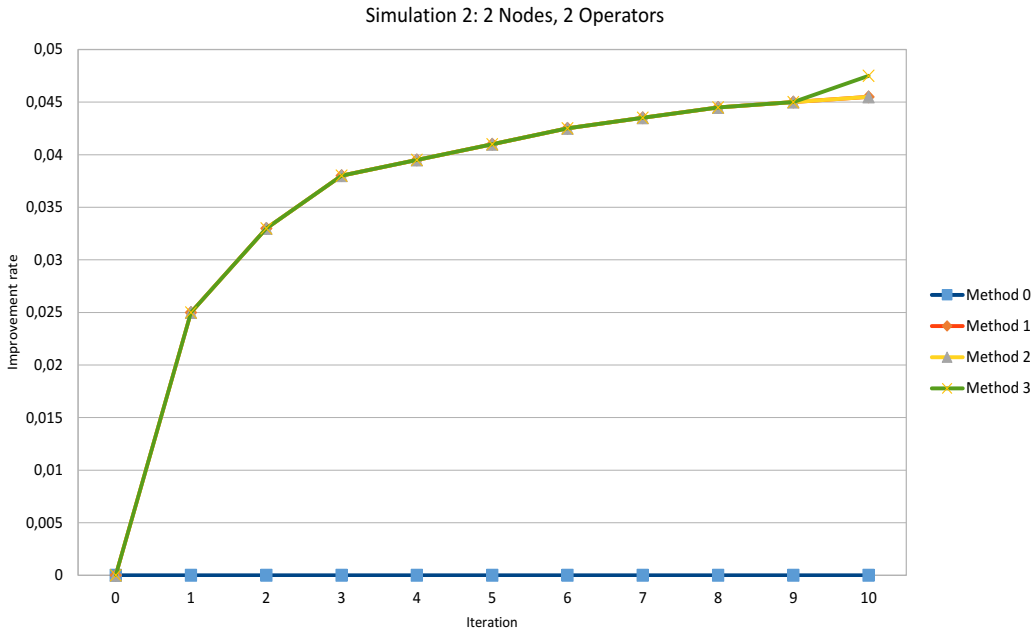

Figure 8: Improvement rate (0-1, Y axis) because of the merging methods (0,1,2,3) with 2 operators and 2 nodes in 10 iterations (X axis)

Table 4: Parameter Setup of Simulation 3

Parameter	Value
Number of node agents	10
Number of iterations	10
Number of operator agents	10

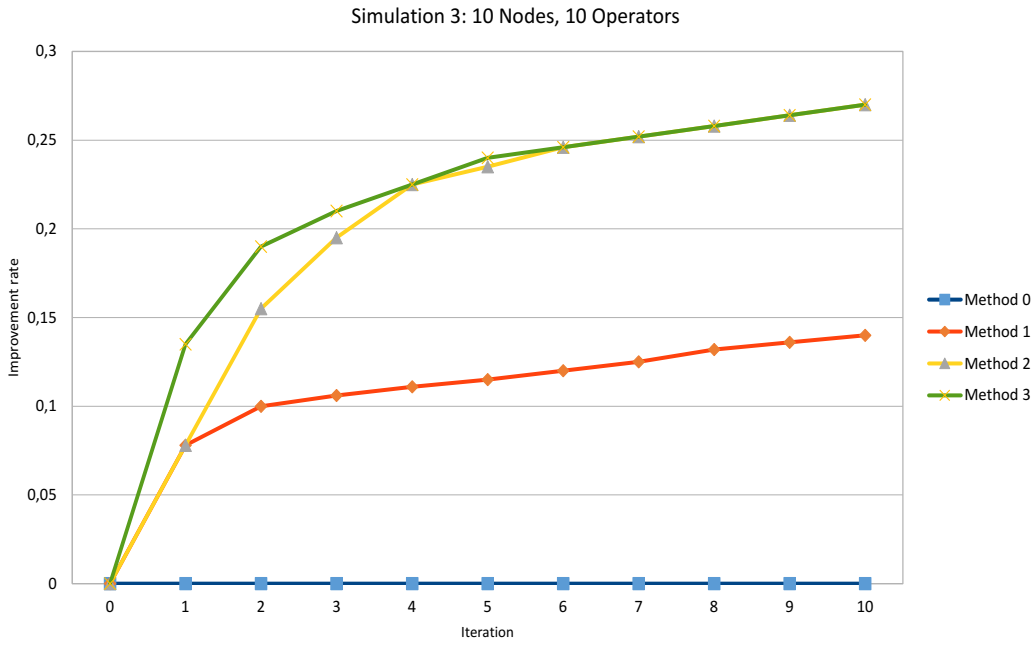

Figure 9: Improvement rate (0-1, Y axis) because of merging methods (0,1,2,3) with 10 operators and 10 nodes along 10 iterations (X axis)

Figure 9, which corresponds to simulation 3, shows the differences of the merging methods, which was present in the first simulation. In addition, the relevance of the improvement in the combined plan generated in each iteration has a similar scale ($0.3 \approx 0.25$). Merging method 1 appears to generate combined plans not so close to the optimal one, whereas merging methods 2 and 3 show similar results, although merging method 3 shows slightly better results in the first iterations (faster convergence). Considering the simulations, we can estimate that most of the improvement is a result of the cooperation of agents in the form of recommendations (method 1 is close to methods 2 and 3) instead of the combination of plans. Therefore, although a combination method to merge plans is required to take advantage of recommendations, most of the improvement is expected from the research on issues from the recommendation system domain (computing the reputation of sources and selecting the correct partners for cooperation) instead of the issues from the planning domain relative to the merging plans. Finally, the set of simulations of our agent system is limited, and many other initial setups can be considered. We intended to maintain a simple agent system to observe the relative contribution of merging plans and the use of recommendations with more clarity.

5. Conclusions

When we must merge plans and do not have sufficient knowledge on the actions and goals, a domain-independent merging method will provide a helpful alternative, which is the motivation of this work. Specifically, in this paper, we have accomplished the following goals:

- we have defined and implemented an agent system that enables the comparison of alternative methods to merge plans based on the roles of the nodes and operators using the BDI paradigm.
- we have assumed a given number of conditions (mainly independent actions) that enable an evaluation of the actions in the plan, which do not depend on intrinsic domain-dependent restrictions over the actions.
- we have justified the use of recommendations and defined a way to weight them according to the past execution of the plans and several simple methods to merge them.

- we have tested the performance of the merging algorithms with simple simulations and observed that the contribution from the recommendations is much greater than that from the plan-merging methods.

We are aware of the limited scope of applicability of our proposal, where many assumptions must be jointly present to justify its use. However, our contribution is innovative because it is located in the boundaries of planning and research issues of the recommender systems and relevant because it broadens the applicability of merging-plan algorithms to problems that were not available (they previously depend on domain-dependency logic). Our method also provides new paths to be explored by other researchers, since our agent system is an open framework (and available in sourceforge [2]) to include many other merging algorithms, including different methods to compute the reputation of recommenders or a simple test with no other initial setup.

Acknowledgements

This work was supported in part by Project MINECO TEC2017-88048-C2-2-R

References

(1997). *Foundations for intelligent phisical agents specification*. Geneve, Switzerland.

Borrajo, D., & Fernández, S. (2018). Efficient approaches for multi-agent planning. *Knowledge and Information Systems*, .

Brahimi, S., Maamri, R., & Sahnoun, Z. (2014). Partially centralized hierarchical plans merging. In A. Badica, B. Trawinski, & N. T. Nguyen (Eds.), *Recent Developments in Computational Collective Intelligence* (pp. 59–68). Springer International Publishing.

Braubach, L., Pokahr, A., & Lamersdorf, W. (2004). Jadex: A short overview. In *Main Conference Net.ObjectDays 2004* (pp. 195–207).

[2] https://sourceforge.net/projects/mpik/

Bruce, B. C., & Newman, D. (1978). Interacting plans. *Cognitive Science*, (pp. 195–233).

Carbo, J., Molina, J., & Davila, J. (2003). Trust management through fuzzy reputation. *International Journal of Cooperative Information Systems*, *12*, 135–155.

Carbo, J., Molina, J., & Patricio, M. (2016). Asset management system through the design of a jadex agent system. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *5*.

Carbo, J., & Molina, J. M. (2010). An extension of a fuzzy reputation agent trust model (afras) in the art testbed. *Soft Computing*, *14*, 821–831.

Decker, K. S., & Lesser, V. R. (1992). Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, *1*, 319–346.

Diniz Da Costa, A., Lucena, C. J., Torres Da Silva, V., Azevedo, S. C., & Soares, F. A. (2008). Trust in agent societies. chapter Art Competition: Agent Designs to Handle Negotiation Challenges. (pp. 244–272). Berlin, Heidelberg: Springer-Verlag.

Elkawkagy, M., & Biundo, S. (2011). Hybrid multi-agent planning. In F. Klügl, & S. Ossowski (Eds.), *Multiagent System Technologies* (pp. 16–28). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ephrati, E., Pollack, M. E., & Rosenschein, J. S. (1995). A tractable heuristic that maximizes global utility through local plan combination. In *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA* (pp. 94–101).

Ephrati, E., & Rosenschein, J. S. (1993). Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 423–429). Chambery, France.

Fischer, K., Müller, J. P., Pischel, M., & Schier, D. (1995). A model for cooperative transportation scheduling. In *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA* (pp. 109–116).

Foulser, D. E., Li, M., & Yang, Q. (1992). Theory and algorithms for plan merging. *Artificial Intelligence Journal*, *57*, 143–182.

Fullam, K., Klos, T., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K. S., Rosenschein, J., Vercouter, L., & Voss, M. (2005). A specification of the agent reputation and trust (art) testbed: Experimentation and competition for trust in agent societies. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)* (pp. 512–518).

Georgeff, M. P. (1983). Communication and interaction in multi-agent planning. In M. R. Genesereth (Ed.), *AAAI* (pp. 125–129). AAAI Press.

Goel, A. K., Ail, K. S., Donnellan, M. W., de Silva Garza, A. G., & Callantine, T. J. (1994). Multistrategy adaptive path planning. *IEEE Expert*, *9*, 57–65.

Gomez, M., Carbo, J., & Benac, C. (2007). Honesty and trust revisited: the advantages of being neutral about other's cognitive models. *Journal Autonomous Agents and Multi-Agent Systems (JAAMAS)*, *15*, 313–335.

Kravari, K., & Bassiliades, N. (2015). A survey of agent platforms, . *18*.

LukeTeacy, W., Huynh, T., Dash, R., Jennings, N., Patel, J., & Luck, M. (2007). The art of iam: The winning strategy for the 2006 competition. In *Procs. of Trust in Agent Societies WS Procs., AAMAS 2007*.

Munoz, V., Murillo, J., Lopez, B., & Busquets, D. (2009). Strategies for exploiting trust models in competitive multi-agent systems. In L. Braubach, W. van der Hoek, P. Petta, & A. Pokahr (Eds.), *Multiagent System Technologies* (pp. 79–90). Springer Berlin / Heidelberg volume 5774 of *Lecture Notes in Computer Science*.

Muscettola, N., & Smith, S. F. (1987). A probabilistic framework for resource-constrained multi-agent planning. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 2* IJCAI'87 (pp. 1063–1066). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

PLAZA, E., & MCGINTY, L. (2005). Distributed case-based reasoning. *The Knowledge Engineering Review*, *20*, 261–265.

Redmond, M. (1990). Distributed cases for case-based reasoning: Facilitating use of multiple cases. In *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, July 29 - August 3, 1990, 2 Volumes.* (pp. 304–309).

Rosenschein, J. S. (1982). Synchronization of multi-agent plans. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 115–119). Pittsburgh, Pennsylvania.

Shieber, S. M. (1985). Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *23rd Annual Meeting of the Association for Computational Linguistics, 8-12 July 1985, University of Chicago, Chicago, Illinois, USA, Proceedings.* (pp. 145–152).

Veloso, M. M. (1994). *Planning and Learning by Analogical Reasoning* volume 886 of *Lecture Notes in Computer Science*. Springer.

de Weerdt, M., Bos, A., Tonino, H., & Witteveen, C. (2003). A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence*, *37*, 93–130.

Wilkins, D. E., & Myers, K. L. (1998). A multiagent planning architecture. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems, Pittsburgh, Pennsylvania, USA, 1998* (pp. 154–162).

Yang, Q., Nau, D. S., & Hendler, J. A. (1992). Merging separately generated plans with restricted interactions. *Computational Intelligence*, *8*, 648–676.

Yann Krupa, J. F. H., & Vercouter, L. (2009). Extending the Comparison Efficiency of the ART Testbed. In M. Paolucci (Ed.), *Proceedings of the First International Conference on Reputation: Theory and Technology - ICORE 09, Gargonza, Italy*.

Zaghetto, C., Aguiar, L. H. M., Zaghetto, A., Ralha, C. G., & Vidal, F. d. B. (2017). Agent-based framework to individual tracking in unconstrained environments. *Expert Syst. Appl.*, *87*, 118–128.