Expert Systems With Applications 115 (2019) 403–411

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

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

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

[a] Departamento Informatica, University of Carlos III de Madrid, Av Universidad 30, Campus Leganes, Madrid 28911, Spain
[b] Departamento Informatica, University of Carlos III de Madrid, Av Universidad Carlos III, 22, Campus Colmenarejo, Madrid 28270, Spain

ARTICLE INFO

Article history:
Received 3 April 2018
Revised 19 July 2018
Accepted 29 July 2018
Available online 3 August 2018

Keywords:
Recommendation
Reputation
Trust
Agents
Planning

ABSTRACT

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

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 of finding plans for a group of agents (Ephrati & Rosenschein, 1993; Rosenschein, 1982). In such an approach, autonomous agents attempt to coordinate their actions to satisfy the different goals of each agent (Georgeff, 1983; Muscettola & Smith, 1987). The drawback of this problem definition is that autonomous agents, who by definition pursue their own particular goals, may adopt a non-cooperative attitude if they do not have personal incentives to cooperate.

Alternatively, a scenario can be considered in which the agents share a common collective goal, in which case multi-agent planning is treated as a "divide et impera" problem. The common goal, which is typically complex, is split into subproblems among the agents. Then, each agent solves one subproblem, and all corresponding subsolutions that are generated in parallel by the agents must be merged into a unified solution (Ephrati & Rosenschein, 1993; Wilkins & Myers, 1998). In some cases, there is no partitioning of the problem, and the agents simply apply different planners independently to the same problem (as in the case of scheduling for transportation systems (Fischer, Müller, Pischel, & Schier, 1995)) to more quickly converge to a solution.

1.1. General context and motivation

The problem of merging plans with no previous partitioning 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 the key elements of the individual plans that give rise to improvements, incompatibilities, efficiencies and other relevant features for solving the problem.

This analysis and search of the key elements of plans to be merged that have been developed by autonomous agents acting as independent planners can become extremely complex. Towards this goal, Bruce and Newman (1978) proposed a structured model of actions, intentions, beliefs and states to be represented in the plans. Rosenschein (1982) proposed a formalism for the representation of plans to facilitate the detection of potential conflicts. Similarly, Georgeff (1983) suggested a definition of correctness and execution conditions to be satisfied in a plan. Shieber (1985) defined a temporal logic system to specify restrictions on the plans, which was used by Yang, Nau, and Hendler (1992) and Foulser, Li, and Yang (1992) to ensure the efficient merging of alternative plans in

* Corresponding author at: Departamento Informatica, University of Carlos III de Madrid, Av Universidad 30, Campus Leganes, Madrid 28911, Spain.
E-mail addresses: javier.carbo@uc3m.es (J. Carbo), miguelangel.patricio@uc3m.es (M.A. Patricio), josemanuel.molina@uc3m.es (J.M. Molina).

https://doi.org/10.1016/j.eswa.2018.07.062
0957-4174/

pursuit of a common goal. A more recent logic system for multi-agent planning was proposed by de Weerdt, Bos, Tonino, and Witteveen (2003). Elkawkagy and Biundo (2011) and Brahimi, Maamri, and Sahnoun (2014) used preprocessing steps to generate a hierarchy of the subplans independently computed by agents. Therefore, for these approaches, merging has a different meaning; it is simply a task of composing hierarchically pre-ordered subplans. This is not true in our case, since we do not assume a previously known hierarchy of subplans; instead, we assume knowledge of the number and similarities of actions (not subplans) that may compose a plan. How they should be composed is what we wish to determine through our planning computation, instead of being previously known. Moreover, Elkawkagy and Biundo (2011) and Brahimi et al. (2014) merged parts of the final plan rather than complete alternatives for the final plan as we do. A very recent work by Borrajo and Fernández (2018) provides an interesting view of distributed planning with autonomous agents with a focus on privacy protection.

The use of execution conditions, logic or temporal restrictions to combine plans is based on the intrinsic characteristics of the plans and the dependencies among the actions that form a plan, which typically depend on the domain. This drawback is often ignored by assuming that these characteristics and dependencies are previously known, although this cannot be the case in real problems. However, plans can also be merged on the basis of 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 this plan with its own plan to create an new, improved global plan (e.g., by 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 plan quality (Ephrati & Rosenschein, 1993): a 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 has also been suggested, in which the agents form an improved global plan by voting on joint actions, with the votes playing the role of the cost in the heuristic (Ephrati, Pollack, & Rosenschein, 1995).

Similar to our proposal, these approaches merge plans using external evaluations of the plans instead of previous restrictions that are assumed to be known. However, these two merging approaches achieve plan improvement by means of an internal analysis that relies on complex formalisms for the representation of knowledge about the plans. Therefore, they are domain-dependent approaches with requirements similar to the logical-temporal restrictions and execution conditions of the previously discussed approaches for combining plans. In contrast, the originality of our approach lies in the method of external evaluation. We evaluate plans in a truly domain-independent manner on the basis of results obtained from past executions of the plans to be combined rather than any rich (and complex) deliberation about the compatibility, efficiency and redundancies of the actions that form the plans. Thus, the evaluation is extrinsic to the actual composition of the plans to be combined. We can say that the underlying inspiration for our method is closer to a neural/subsymbolic approach than to a deliberative/symbolic approach: the evaluation of a plan is the output of a black box with two inputs—direct experience and recommendations. These inputs represent two indirect methods of estimating the quality of a plan, which are not based on expressions of the positive/negative interrelationships among the actions that form the plans to be combined. Therefore, our approach does not rely on previous, certain and complete domain knowledge.

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

To endow the agents with this ability, their past experiences must be represented in a context-sensitive manner by means of an integration mechanism based on utility control for these experiences. In simple terms, the agents must be able to distinguish (contextualize) which of their experiences are similar to one another and to the new situation; this is, in fact, an estimation process. This problem has been addressed in planning systems that use case-based reasoning, e.g., Veloso (1994), Redmond (1990), Goel, Ail, Donnellan, de Silva Garza, and Callantine (1994), and Plaza and McGinty (2005). Solving the problem of combining past experiences requires the use of contextual information; in our case, the relevant past experiences are previously executed plans. This contextual information is an estimate of the applicability of a particular plan to the current situation faced by the agent, even if this estimate is roughly computed from indirect sources of information (the aforementioned two inputs to the black box).

1.2. Assumptions

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

We must also make the following assumptions about the nature of the autonomous agents that perform the plan combinations:

- They are self-interested and mainly focus on achieving 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 conflicts 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 states.

In planning systems, plans are often considered partially or completely ordered in accordance with given restrictions on their actions, where partial ordering allows correct plans to consist of different sequences of actions, whereas a completely ordered plan represents 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 corresponding to each step of the plan. We also assume that the actions are independent: the actions that should be executed before and after a particular action do not alter the suitability of that action to be assigned to the correct step.

We assume that the final combined plan is executed by an active ‘operator’, which can have different levels of expertise, and is applied to a ‘node’ or physical passive element that requires the execution of successive plans with a frequency that depends on the success of the previously executed plan. For example, a major city can consider the infrastructure elements of the city, which deteriorate over time, as nodes that thus require maintenance over time. In our urban example, the operators can be human resources that perform maintenance tasks (plans) on the infrastructure elements,

which require a certain expertise level and a correct sequence of operations (plan actions) to succeed. Success implies that a long time period will elapse before the next maintenance task is required.

To represent the 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, nodes with minor differences can be 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 a given node. Therefore, we define an action by 3 values that represent its suitability with respect to the node to which it is to be applied:

- the corresponding type of the node on which the action is to be executed,
- the corresponding subtype of the node on which the action is to be executed, and
- the corresponding time (sequence order in the plan) at which the action is to be executed.

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

Additionally, we assume that the number of steps in a plan is public, previously known and limited (small). Hence, the suitability of an action for assignment to a given plan step can be computed in accordance with the placement of that action in the plan (when the action is placed near or in the correct plan step).

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

2. Defining the agent system

Both types of agents (node and operator agents) are implemented in JADEX (Braubach, Pokahr, & Lamersdorf, 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 to agent reasoning structured into three levels of knowledge: beliefs, desires and intentions (henceforth, BDI). Many other agent platforms also implement this standard and BDI reasoning (Kravari & Bassiliades, 2015). Nonetheless, a survey from 2015 explicitly shows that JADEX provides a higher level of robustness and performance than 24 alternative agent platforms do. More recent platforms that claim to be more efficient, such as SPADE[1], can be considered, but JADEX has been largely successfully applied in research, teaching and industrial applications, and we have previous expertise in developing applications with it. Furthermore, we have published a deeper explanation of how agents internally use beliefs, desires and intentions in this problem in Carbo, Molina, and Patricio (2016).

- The beliefs of the agents and the contents 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 (where an ordered sequence of such operations forms a plan), which is defined by the operation type; node type and subtype to which an action is to be applied; and expertise, which represents the ability of the operator that executes the operation plans. The beliefs of a node agent include the current operator agent and the expected time until a new operation plan must be executed on this node. An operator agent holds the following beliefs: the availability of the operator, corresponding to the number of node(s) it is currently able to operate on; the node type(s) in which it specializes; its expertise; its most recent operation plan executed on each type of node; and the other operator agents that act as recommenders (identifiers and expertise) and their corresponding last recommendations.
- The intentions of the agents correspond to the actions performed by an operator or node agent in pursuit of a given goal (conceptually, an instantiated desire). Typically, they involve receiving a message, accessing the content of that message and the 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 the agents are implemented in JADEX as goals that can be triggered by external events (reception of a message) or internal events (specific conditions on the beliefs of the agents are satisfied). Such goals in our agent system implementation consist of the complete execution of the corresponding role (initiator/participant, in FIPA terms) in a protocol.

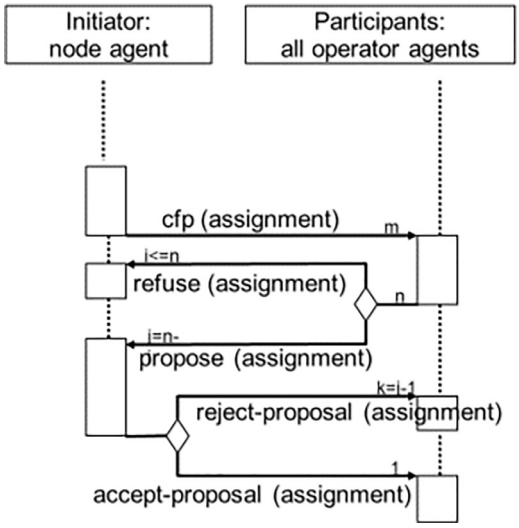

Fig. 1. CFP interaction protocol between a node agent and an operator agent.

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

- The first protocol launched is a call-for-proposal (CFP) protocol (see Fig. 1), in which a node agent acts as the initiator and an operator agent acts as a participant. This protocol is associated with the goal of the node agent to be assigned to an operator agent, and it is triggered by an internal event of the node agent, namely, its beliefs show no current operator agent.
- Next, an interaction proposal protocol is launched, with the node agent as the initiator and the operator agent as the participant; the associated goal is triggered by an internal event in which the node agent observes that the belief regarding the

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

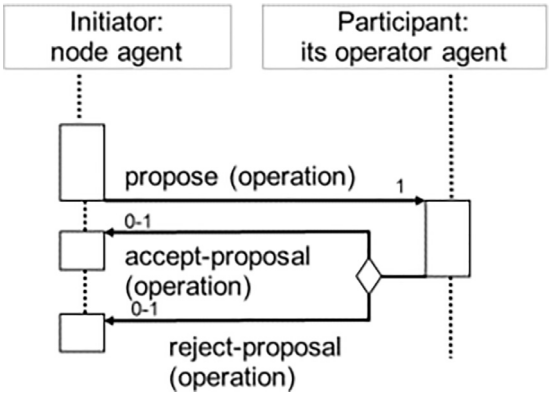

Fig. 2. Interaction proposal protocol between a node agent and an operator agent.

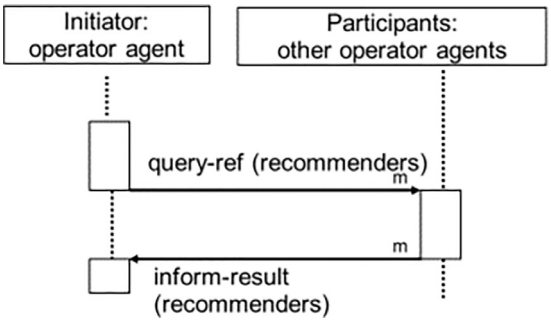

Fig. 3. Recommender query interaction protocol between two operator agents.

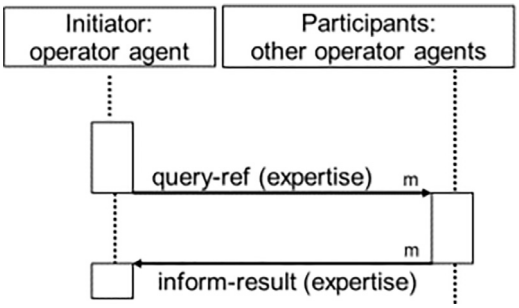

Fig. 4. Expertise query interaction protocol between two operator agents.

expected time at which a new operation plan is required to be executed corresponds to the current time (see Fig. 2).

- Then, a query interaction protocol is launched, with an operator agent as the initiator and another operator agent as the participant. This query protocol identifies the recommenders that are specialized in a particular node type. It is triggered by an internal event of the initiating operator agent in which its beliefs show that it must generate an operation plan for a node agent of a given type and it has no recommenders for this node type among its beliefs (see Fig. 3).
- Another query protocol is launched, in which an operator agent, acting as the initiator, asks another operator agent, which acts as a participant, about its expertise on a given node type to determine the suitability of the participating operator agent as a potential recommender. This protocol is triggered by an internal event of the initiating operator agent in which its beliefs show that it must generate an operation plan for a node agent and that it must update the expertise of recommenders for this node type among its beliefs (see Fig. 4).
- Subsequently, another query protocol is initiated by the operator agent to ask another operator agent, which acts as a participant, what operation plan the participating operator agent recommends for a particular type of node. This protocol is triggered by an internal event of the initiating operator agent in which its beliefs show that it must generate an operation plan for a node agent of a given type and that it must update the operation plan recommended by recommenders for this node type among its beliefs (see Fig. 5).

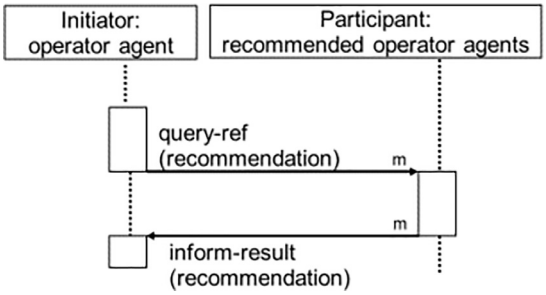

Fig. 5. Recommendation query interaction protocol between two operator agents.

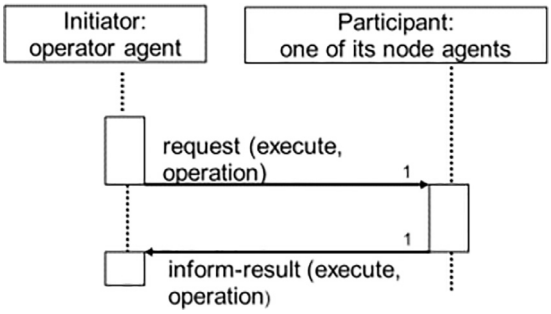

Fig. 6. Plan execution request interaction protocol between an operator agent and a node agent.

- Finally, a request protocol is launched by the operator agent as the initiator, with a node agent as the participant. The operator agent requests the node agent to execute an operation plan that is generated by combining the received recommendations with the most recent operation plan of this operator agent for that type of node. This protocol is triggered by an internal event of the operator agent in which its beliefs show that it must generate an operation plan for a node agent of a given type and that all of its beliefs corresponding to the expertise and recommended operation plans for this node type have been updated (see Fig. 6).

3. Merging plans

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

- expertise about the given node type from the operator agents that act as recommenders,
- the operation plans that the operator agents recommend for the given node type,
- its own previous expertise about that node type, and
- its own previous operation plan for that 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 response from the node agent will include the time expected to elapse before a new operation plan is required. This time is computed on the basis of the level of success achieved with the operation plan suggested by the operator agent, and it will be used

by the operator agent to update its own expertise on the current node type.

We quantify this level of success based on the equations in the Agent Reputation Testbed (ART) platform (Fullam et al., 2005). This platform has been used in several previously published experiments (Diniz Da Costa, Lucena, Torres Da Silva, Azevedo, & Soares, 2008; LukeTeacy et al., 2007; Munoz, Murillo, Lopez, & Busquets, 2009; Yann Krupa & Vercouter, 2009) and in several international competitions at the International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

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

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

The ART platform generates the error on a service estimate from a normal distribution centred on the difference between the estimated value (denoted by e) and the true value (denoted by t) of the service; the standard deviation s represents the noise that prevents optimal perception of the true value of the service in real life (Eq. (1)). s takes its value from Eq. (2). In this equation, E denotes the expertise of the agent in providing such service, and c is the cost to the agent for providing the service, i.e., the time invested by the agent (which represents its effort). Thus, higher expertise, more invested effort, and higher estimation quality correspond to less noise applied to the normal distribution.

We can compute the difference between the optimal plan and the suggested plan in a similar manner (the node agents will compute this difference). We assume that such an optimal plan is available, that plans have a limited number of steps, and that the possible operations that may be applied in each step are limited and depend on the node type and subtype. Thus, along with the number of steps, the numbers of types and subtypes, and the weights for making a mistake in each step, all node types and subtypes are setup parameters of the agent system.

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

Each operation in the optimal plan is defined by three dimensions: the correct type of node to which that operation should be applied, the correct node subtype and the correct time step in which the operation should be applied. The difference between an operation in a plan suggested by an operator agent and the optimal operation is computed as a weighted sum of the distances in terms of the node type, node subtype and time step (sequence order), as shown in Eq. (3).

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

Once the node agent knows the level of success achieved with the plan suggested by the operator agent, the node agent must compute the expected time that will elapse before the execution of the next plan will be required. This time value is computed as shown in Eq. (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 as a static-valued parameter in the setup of the agent system). Thus, a higher level of success of the plan implies that less time will elapse before the required execution of 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 is associated with two values:

- Global expertise: the number of times that an operator agent has executed plans (regardless of their success) for a given node type. The global expertise value will increase up to a given threshold, which is defined as a setup parameter; then, after this maximum number of plan executions, the global expertise will remain fixed at the maximum value. The global expertise is used instead of the invested time in Eq. (2), under the assumption that all operator agents spend an identical amount of time in the execution of plans but that operator agents with more global expertise are more efficient within that time period.
- Local expertise: the ability to suggest a plan for a given node type. The local expertise value is computed as the inverse of the standard deviation s; thus, it has a recursive definition, with past local expertise affecting the computation of the current local expertise, as s is defined in Eq. (5) as depending on the local expertise.

In terms of these two aspects of expertise, Eq. (2) from the ART platform is redefined in our agent system as shown in Eq. (5).

Meanwhile, an operator agent must weight the plans recommended by other operator agents to combine them into a single new plan to suggest to the node agent. The reputation of an operator agent 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 values of the operator agent using Eq. (6), where E_l is the local expertise, E_g is the global expertise, and *Thr* is the threshold for the global expertise.

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

Next, we will present 4 methods of using all of these elements to combine plans. The first method is a no-merging method, which is used only as a benchmark against which to evaluate the improvements in the levels of success of operator agents that combine plans to formulate the plans to be suggested to node agents.

3.1. Merging method 0: no merging, ignoring recommendations

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

The operator agent ignores all received recommendations, and it executes plans that are updated exclusively on the basis of its previous executions of plans, as shown in Eq. (7), where s is the success of the execution of a previous plan. This method is used as the benchmark, and the relative benefits of applying the other methods can be measured by the improvements in the results obtained with respect to the results of this non-cooperative 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 selects the best plan from among all possible options (recommendations and previous plans for that node type) according to the computed reputations of the operator agents, with no combination, as shown in Eq. (8), where *rep* is the reputation of the operator agent from which the corresponding recommendation was received.

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 a new plan by combining the operations with the highest reputation (inherited from the operator agent that suggested this operation in this plan step in its recommended plan) and popularity (number of times that this operation was suggested by the operator agents). In other words, we sum all reputations of each possible operation for each plan step; thus, operations with multiple recommendations and operations from operators with good reputations are more likely to become the suggested operation in the corresponding step of the new plan. This process is summarized in Eq. (9), where *numoper* is the number of different recommended operations for a given time step j and *numrec* is the number of recommenders that suggested that operation i be applied in time step j.

3.4. Merging method 3

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

The operator agent modifies the plan recommended by the operator agent with the highest reputation. The modification consists of replacing some number of the operations in this best plan (according to operator reputation) that it shares with the worst plan in the same plan step. These operations are replaced by different suggested operations from other recommended plans (selected in decreasing order of the reputation of the recommending operator agent). The number of replacements is fixed as a setup parameter. Therefore, this combination method is similar to an evolutionary algorithm, in which the plans are the individuals to be crossed and the reputation plays the role of a fitness function. This process is summarized in Eq. (10), where i goes from 1 to the number of operations recommended for time step $k-1$, which is initially assigned an operation that is shared between the best and worst plans.

4. Experimentation

Since our merging proposals are based on an external evaluation, as stated in Section 1, they substantially differ from the classic approaches to merging plans; these classic approaches rely on logical domain-dependent restrictions, whereas our approach does not. Therefore, because our inputs and assumptions are different, we do not present any comparisons with other merging methods in a shared agent framework, as Zaghetto, Aguiar, Zaghetto, Ralha, and Vidal (2017) did with tracking algorithms, since they are different in nature. We also do not report any simulations based on realistic data from any problem domain, since our proposal is intended to be generic and independent of the domain. Instead, we focus on testing the four merging methods with respect to the relative contributions of the cooperative recommendations and the evolution due to past experience. We define two simple simulations, 1 and 2, in which the isolated improvements due to evolution and cooperation are considered, and simulation 3, in which both evolution and cooperation are jointly applied. We consider the parameter setup in Table 1 for all simulations. The weights of the mis-prediction errors for types, subtypes and operations are all set to 1 since using different weights would introduce the possibility of giving more or less relevance to one of the three categories over the others. Different weights can be useful if we know in advance that in a given domain, errors in one of the three categories will have more costly consequences. Here, however, we assume no previous knowledge of the problem domain.

Table 1
General parameter setup.

Parameter	Value
Number of node agents	1
Number of operator agents	20
Number of recommenders an operator may query	20
Number of plan steps	5
Number of node types	1
Number of node subtypes	2
Weights of mis-prediction errors for types, subtypes and operations	1
Initial expertise of operators	1
Number of replacements in merging method 3	1
Generation of previous plans of operator agents	Random

Table 2
Parameter setup for simulation 1.

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

Table 3
Parameter setup for simulation 2.

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

First, we consider a simple test, called simulation 1, with the parameter setup given in Table 2. In simulation 1, the node agent first selects one of the 20 operator agents as its operator. Next, the selected operator agent asks for recommendations from the other 19 operators on the basis of their previous experiences. Finally, the resulting plan is generated by combining the 19 responses with the previous experience of the selected operator (all of which are false, randomly generated experiences). Because there is only one node agent, the other 19 operator agents do not have any real "experience"; they act only as "dumb" recommenders (always sending the same, constant, randomly generated recommendations). 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.

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

In simulation 2, with the initial parameter setup given in Table 3, the level of cooperation is decreased (because there are 2 operator agents instead of 20), but the information provided by the recommendations is neither constant nor random because it is based on the evaluation of corresponding plans. Therefore, the recommendations are improved in each iteration. Again, each operator agent is linked to only 1 node agent because their availability parameter remains equal to 1, and the two operator agents share their relative successes and failures over 10 iterations.

Fig. 8 shows the information obtained from simulation 2 on the two 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

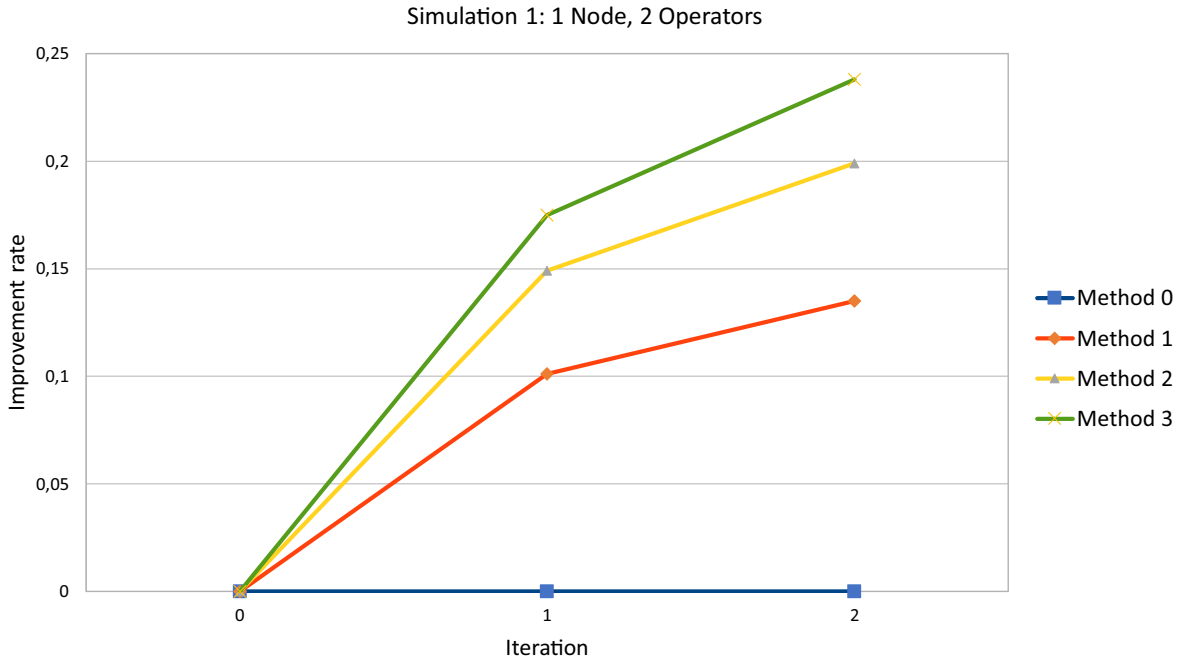

Fig. 7. Improvement rates (0–1, *Y*-axis) achieved with merging methods 0, 1, 2, and 3 with 20 operator agents and 1 node agent over 2 iterations (*X*-axis).

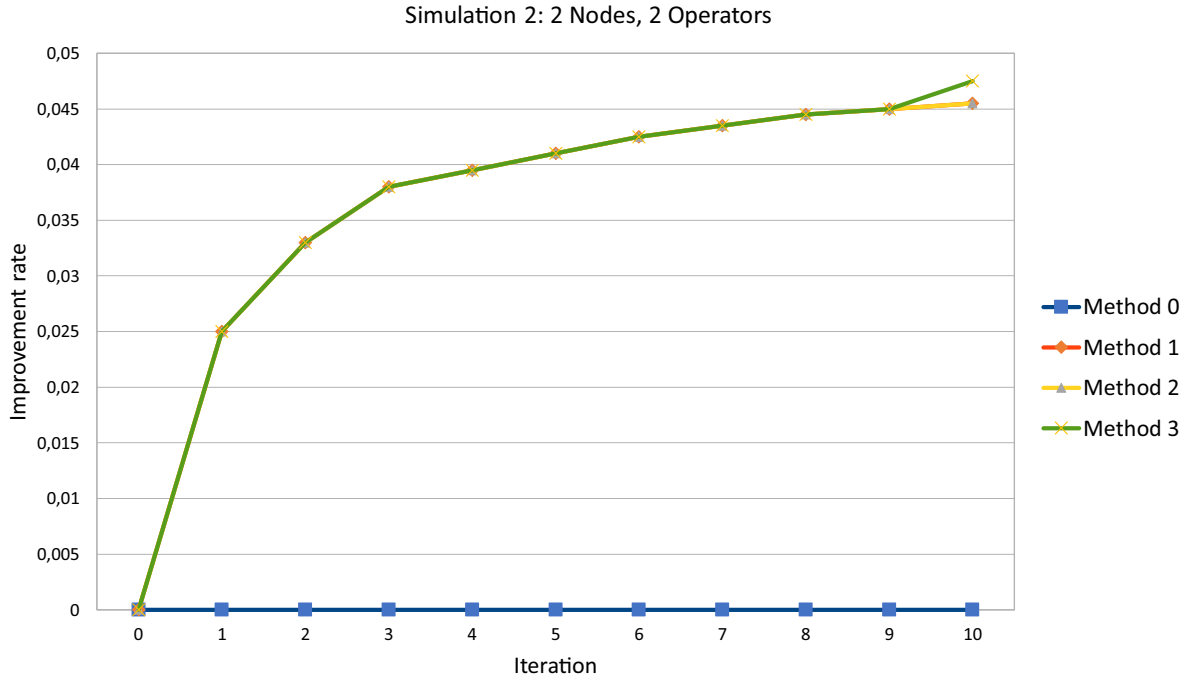

Fig. 8. Improvement rates (0–1, *Y*-axis) achieved with merging methods 0, 1, 2, and 3 with 2 operators and 2 nodes over 10 iterations (*X*-axis).

the operator agents combine their plans with only one other recommendation (instead of 19, as in the previous simulation). Thus, the merging methods exhibit much better performance when they have fewer plans to combine, even when these plans are not accurate or updated (they were constant in the previous simulation). Additionally, the differences among the 'true' merging methods (only methods 1–3, as method 0 cannot be considered a merging method) are notably small in these circumstances.

Finally, we run simulation 3 with the parameter setup given in Table 4, with 10 node agents and 10 operator agents over 10 iterations, to observe the evolution due to changes in previous experiences in combination with 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.

Table 4
Parameter setup for simulation 3.

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

Fig. 9, which presents the results of simulation 3, shows differences among the merging methods that are similar to those

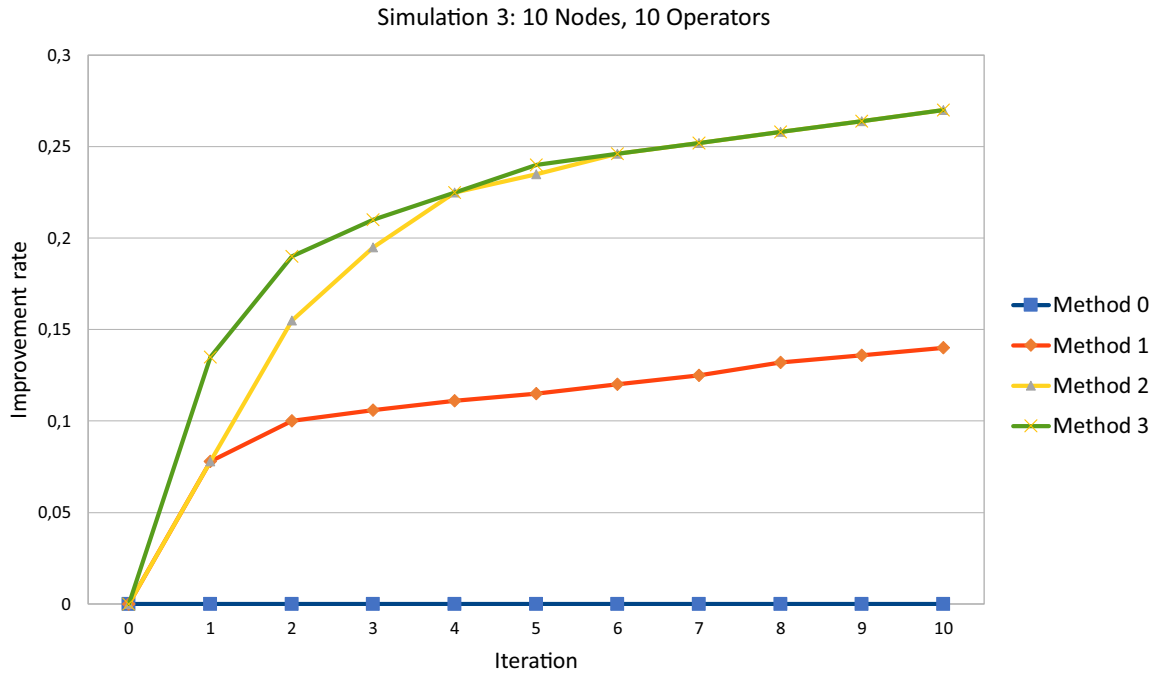

Fig. 9. Improvement rates (0–1, *Y*-axis) achieved with merging methods 0, 1, 2, and 3 with 10 operators and 10 nodes over 10 iterations (*X*-axis).

observed in the first simulation. In addition, the improvement in the combined plan generated in each iteration is of a similar scale ($0.3 \approx 0.25$). Merging method 1 appears to generate combined plans that are not especially close to the optimal one, whereas merging methods 2 and 3 yield similar results, although merging method 3 shows slightly better results in the early iterations (faster convergence). Considering the simulations, we can estimate that most of the improvement can be attributed to the cooperation among the agents in the form of recommendations (the results of method 1 are close to those of methods 2 and 3) instead of the combination of plans. Therefore, although a combination-based method of merging plans is required to take full advantage of recommendations, greater improvement can be expected from research on issues related to the recommendation system domain (computing the reputations of sources and selecting the correct partners for cooperation) than from research on issues related to the planning domain concerning the merging of plans. Finally, this set of simulations of our agent system is limited, and many other initial setups could be considered. We intend to maintain and extend our simple agent system to more clearly observe the relative contributions of merging plans and considering recommendations.

5. Conclusions

When we must merge plans and do not have sufficient knowledge of the corresponding actions and goals, a domain-independent merging method can serve as a helpful alternative to previously developed methods, which is the motivation for 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 of merging plans based on the roles of the node and operator agents using the BDI paradigm.
- We have assumed certain conditions (mainly the independence of actions) that enable an evaluation of the actions in a plan without relying on intrinsic domain-dependent restrictions on the actions.
- We have justified the use of recommendations and have defined a way to weight them in accordance with past executions of the recommended plans and several simple methods for merging them.
- We have tested the performance of the proposed merging algorithms through simple simulations and have observed that the contribution from the cooperative recommendations is much greater than that from the plan-merging methods.

We are aware of the limited scope of applicability of our proposal, which requires many assumptions to justify its use. However, our contribution is innovative because it is located on the boundary between planning and research issues related to recommender systems, and it is relevant because it extends the applicability of plan-merging algorithms to problems that such algorithms could not previously address (they previously depended on domain-dependent logic). Our method also provides new paths to be explored by other researchers, since our agent system is an open framework (and available on SourceForge[2]) that can be extended to include many other merging algorithms, including different methods of computing the reputations of recommenders or simple tests with no other initial setup.

Acknowledgments

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

References

fip,(1997). Foundations for intelligent physical agents specification. Geneve, Switzerland.

Borrajo, D., & Fernández, S. (2018). Efficient approaches for multi-agent planning. *Knowledge and Information Systems*. https://link.springer.com/article/10.1007/s10115-018-1202-1#citeas.

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.

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

Braubach, L., Pokahr, A., & Lamersdorf, W. (2004). Jadex: A short overview. In *Main conference Net.ObjectDays 2004* (pp. 195–207).
Bruce, B. C., & Newman, D. (1978). Interacting plans. *Cognitive Science, 2*(3), 195–233.
Carbo, J., Molina, J., & Davila, J. (2003). Trust management through fuzzy reputation. *International Journal of Cooperative Information Systems, 12*(1), 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*(2), 1–14.
Carbo, J., & Molina, J. M. (2010). An extension of a fuzzy reputation agent trust model (afras) in the art testbed. *Soft Computing, 14*(8), 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). Chapter Art Competition: Agent Designs to Handle Negotiation Challenges. In *Trust in agent societies* (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*(2-3), 143–182.
Fullam, K., Klos, T., Muller, G., Sabater, J., Schlosser, A., Topol, Z., et al. (2005). A specification of the agent reputation and trust (art) testbed: Experimentation and competition for trust in agent societies. In *Proceedings of 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*(6), 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*(3), 313–335.
Kravari, K., & Bassiliades, N. (2015). A survey of agent platforms, *18*, 1–11.
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 *Proceedings 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*. In *Lecture Notes in Computer Science: 5774* (pp. 79–90). Springer Berlin / Heidelberg.
Muscettola, N., & Smith, S. F. (1987). A probabilistic framework for resource-constrained multi-agent planning. In *Proceedings of the tenth international joint conference on artificial intelligence – volume 2*. In *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*(03), 261–265.
Redmond, M. (1990). Distributed cases for case-based reasoning: Facilitating use of multiple cases. In *Proceedings of the eighth 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 *Proceedings of the twenty-third annual meeting of the association for computational linguistics, 8–12 July 1985* (pp. 145–152).
Veloso, M. M. (1994). Planning and learning by analogical reasoning. *Lecture Notes in Computer Science*: 886. 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*(1), 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 Systems with Applications, 87*(C), 118–128.