

Towards Alternative Approaches to Reasoning about Goals

Patricia H. Shaw and Rafael H. Bordini

Department of Computer Science,
University of Durham, U.K.
{p.h.shaw,r.bordini}@durham.ac.uk

Abstract. Agent-oriented programming languages have gone a long way in the level of sophistication offered to programmers, and there has also been much progress in tools to support multi-agent systems development using such languages. However, much work is still required in mechanisms that can reduce the burden, typically placed on programmers, of ensuring that agents behave *rationally*, hence being effective and as efficient as possible. One such mechanism is *reasoning about declarative goals*, which is increasingly appearing in the agents literature; it allows agents to make better use of resources, to avoid plans hindering the execution of other plans, and to be able to take advantage of opportunities for reducing the number of plans that have to be executed to achieve certain combinations of goals. In this paper, we introduce a Petri-net based approach to such reasoning, and we report on experimental results showing that this technique can obtain comparable improvements on an agent's behaviour to other existing approaches (our experiments do not yet cover reasoning about resource usage). Our long-term goal is to provide a number of alternative approaches for such reasoning, evaluate and compare their performances under different configurations, and incorporate them into interpreters for agent-oriented programming languages in such a way that the most appropriate approach is used at given circumstances.

1 Introduction

Recent years have seen an astonishing progress in the level of sophistication and practical use of various different agent-oriented programming languages [3]. These languages provide constructs that were specifically created for the implementation of systems designed on the basis of the typical abstractions used in the area of autonomous agents and multi-agent systems, therefore of much help for the development of large-scale multi-agent systems. However, the burden of ensuring that an agent behaves *rationally* in a given application is left to programmers (even though the languages do offer some support for that task).

Clearly, it would make the work of multi-agent systems developers much easier if we could provide (semi-) automatic mechanisms to facilitate the task of ensuring such rationality, provided, of course, that they are sufficiently fast to be used in *practical* agent programming languages. One important issue for a

rational agent is that of *deliberation* — that is, deciding *which goals to adopt* in the first place (see [15, 9, 2] for some approaches to agent deliberation in the context of agent programming languages). Besides, once certain goals have been adopted, the particular choice of plans to achieve them can cause a significant impact in the agent’s behaviour and performance, as particular plans may interfere with one another (e.g., through the use of particular resources, or through the effects they have in the environment). The general term that has been used to refer to the reasoning that addresses these issues, which requires *declarative goal* representations [25, 24], is *reasoning about goals*.

Much work has been published recently introducing various approaches which contribute to addressing this problem [7, 21–23, 11, 16]. In most cases, in particular in the work by Thangarajah *et al.* and Clement *et al.*, the idea of “summary information” is used in the proposed techniques for reasoning about goals. However, the size of such summary information can potentially grow exponentially on the number of goals and plans the agent happens to be committed to achieve/execute [8]. It remains to be seen how practical those approaches will be for real-world problems.

In our work, we are interested in mechanisms for goal reasoning which do not require such summary information, yet can reproduce the reasoning that has been proposed in the literature. Avoiding the use of summary information, of course, does not guarantee that those alternative techniques will be more efficient than the existing approaches. In fact, our approach is to try and use well-known formalisms with which to attempt to model the goal reasoning problem, then experimentally evaluating the various different approaches. We aim, in future work, to combine those approaches in such a way that agents can use one mechanism or another in the circumstances where each works best, if that turns out to be practically determinable.

So far, we have been able to model the goal reasoning problem using two different approaches, neither of which requires summary information as in the existing literature on the topic (the next section gives a detailed description of such work). First, we have modelled goal-adoption decision making as a reachability problem in a Petri net [14]. Then, using the idea and method suggested in [18, 17] for translating a Hierarchical Task Network (HTN) plan into a Constraint Satisfaction Problem (CSP), we have also developed a method for, given an agent’s current goals and plans (possibly including a goal the agent is considering adopting), generating an instance of a CSP which can produce a valid ordering of plans — if one exists — to help the agent avoid conflicts (and take advantage of opportunities) when attempting to achieve all its goals.

For reasons of space, in this paper we focus on presenting the Petri-net based technique only, and we also give initial experimental analysis of an agent’s performance when using such goal reasoning in two different scenarios; the results of the CSP-based technique will be reported in a separate paper. The remainder of this paper is organised as follows. Section 2 gives an overview of the types of goal reasoning and various approaches to that problem that have appeared in the literature. Then, in Section 3, we look at how such reasoning can be repre-

sented as a Petri net. Section 4 provides an experimental analysis of the Petri-net based reasoning. Finally, we give conclusions and a summary of future work in Section 5.

2 Reasoning About Goals

There are multiple types of conflicts that rational agents need to be aware of; these can be internal to the individual agent, or external between two or more agents [10]. While conflicts can occur in social interactions, when attempting to delegate or collaborate over a set of given tasks [5], the main focus of this paper is to look at conflicts between goals within an individual agent.

The conflicts arise within a single agent when it has taken on two or more goals that are not entirely compatible [10]. The conflicts may be caused if there is a limited amount of resources available [23, 16], or it may be due to the effects the actions involved in achieving the goals have on the environment; the actions in the plans being executed to achieve concurrent goals can cause effects which can hinder, or even prevent altogether, the successful completion of some of those plans [21, 22].

In all the work by Thangarajah *et al.* referred above, a Goal-Plan Tree (GPT) is used to represent the structure of the various plans and sub-goals related to each goal (see Figure 1). In order for a plan within the tree to be completed, all of its sub-goals must first be completed. However, to achieve a goal or a sub-goal, only one of its possible plans needs to be executed. At each node on the tree, *summary information* is used to represent the various constraints under consideration. The reasoning done in their approach is solely internal to an individual agent.

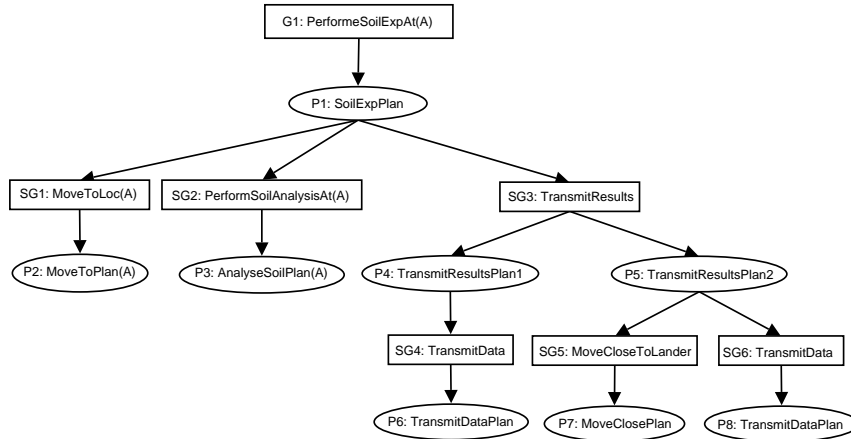


Fig. 1. Goal-Plan Tree for a Mars rover as used by Thangarajah *et al.* Goals and sub-goals are represented by rectangles, while plans are represented by ovals.

Reasoning about effects of actions needs to consider both positive and negative impacts in relation to other plans, and causal links that may exist between goals. In the first paper by Thangarajah *et al.* where reasoning about effects is considered, they show how to detect and avoid negative interference between goals [21]. By using additional types of summary information, similar to those developed in [7], such as summaries for definite or potential pre-conditions and in-conditions along with post-conditions or effects, they monitor the causal links between effects produced by a plan which are used as pre-conditions of another to ensure these are not interfered with. To derive these effects, a formal notation based on set theory is defined, to allow agents to produce the summary information in order to reason about conflicting actions between its current goals and any new goals the agent might consider adopting.

When conflicts occur, often they can be handled by scheduling the plan execution so as to protect the causal links until they are no longer required. Also in [21], the authors determine a sequence of steps for an agent to schedule plan execution so as to avoid interference, including checks that need to be performed before an agent can accept to adopt a new goal. Empirical results from experiments using the reasoning described in that paper are given in [19], comparing the performance of an agent with and without such reasoning, varying the level of interaction between goals and the amount of parallelism. The results show the improvement in number of goals successfully achieved, and only slight increase in time taken to perform the additional reasoning.

In [22], Thangarajah *et al.* focus on exploiting positive interaction between goals. This is where two or more plans cause the same effect, so rather than executing both, it might be possible to merge the two plans, thereby improving the agents' performance. To represent this form of reasoning, they again use the goal-plan tree with summary information including *definite* and *potential* effects of the plans and goals; they also define a particular method to derive such summaries. They then describe how an agent can decide if it is feasible to merge the plans, and how to avoid waiting too long if one of the two plans selected for merging is reached considerably sooner than the other or the second plan is never reached, in case it was a "potential" merge rather than a "definite" merge. Results from experiments using this type of reasoning are once again presented in [19].

Horty and Pollack also consider positive interaction between plans [11]. In their work, an agent evaluates the various options it has between its goals within the context of its existing plans. They use estimates for the costs of plans, and where there is some commonality between some plans, those plans will be considered for merging. If the estimated merged cost is less than the sum of the two separate estimated costs, then the plans are actually merged. The example they give to illustrate this is an "important" plan for going to a shopping centre to buy a shirt, while also having a less important goal of buying a tie. Both plans involve getting money and travelling to a shopping centre, so if the overall cost of buying the tie at the same time as the shirt is less than that of buying the tie separately, then the plans will be merged, even though the goal of having a tie

is not as important. In this way, they look for the least expensive execution of plans involved in achieving the goals.

When referring to reasoning about resource usage in a GPT [23], Thangarajah *et al.* consider both reusable and consumable resources. For example, a communication channel is a reusable resource, while energy or time is consumed so they cannot be reused. Summaries of the resource requirements are passed up the tree towards the goal, describing which resources are *necessary* in order to achieve the goals, and also which resources are used only *potentially*. They introduce a notation, based on set theory, allowing the derivation of summaries for the resource requirements of each goal and plan with sub-goals. These can then be used to reason about where conflicts might occur, so that they can be avoided by choosing suitable alternative plans or appropriately ordering plan execution. An algorithm is given to compute whether it is feasible to add a new goal to the existing set of goals without rendering them unachievable. The initial formulation of the goal-plan tree and summary information for an agent is produced at compile time, and the highlighted conflicts are then monitored at runtime in an attempt to avoid conflict.

Empirical results from experiments done using such reasoning are given in [20]. They consider goal-plan trees of depth 2 and depth 5, varying the amount of parallelism between multiple goals, and the amount of competition for the resources either by reducing their availability or increasing the number of goals competing for the same resources. The reasoning is implemented as an extension to the JACK agent development system [4]; the extended system is called X-JACK. The performance of X-JACK is compared against the performance of JACK without any of the additional reasoning, and shows an improvement in performance regarding the number of goals successfully achieved, typically with only a half-second time increase in the computation cost.

In comparison, [16] consider the use of limited resources when deliberating and performing actions in a multi-agent environment, where coordination and negotiation with the other agents is required. In their attempt to address the problem of limited resources within meta-level control, they make use of reinforcement learning to improve the agents' performance over time.

To our knowledge, while Thangarajah *et al.* have reported on experimental results for reasoning separately about each of the types of interactions between plans and goals as well as resource usage, no results appear in the literature showing what is the performance obtained when an agent is doing all those forms of reasoning simultaneously. All results are given for the individual types, to demonstrate the sole effects from the individual reasoning and the (typically very small) amount of added computational costs associated with it. The lack of combined results seem to suggest the possibility of there being interference between the different forms of reasoning presented in their approach. For example, if one reasoning suggests that performing a particular plan will cause one type of conflict (say, lack of resources), while another reasoning suggests that the only alternative plan for that goal will also cause a conflict (say, a negative interference with another goal), the agent may be unable to decide between the

two without some additional overriding reasoning. It also remains unknown if their approach is still equally efficient when the various types of reasoning are combined.

The results were also limited in the depth of trees tested. In the real world, it is likely the plans (and hence the goals) would be far more complex, leading to trees of significantly greater sizes. However, using the summary information, as a goal-plan tree grows, the amount of summary information to handle could potentially grow exponentially [8], which would have a significant impact on the performance of the agent for larger problems.

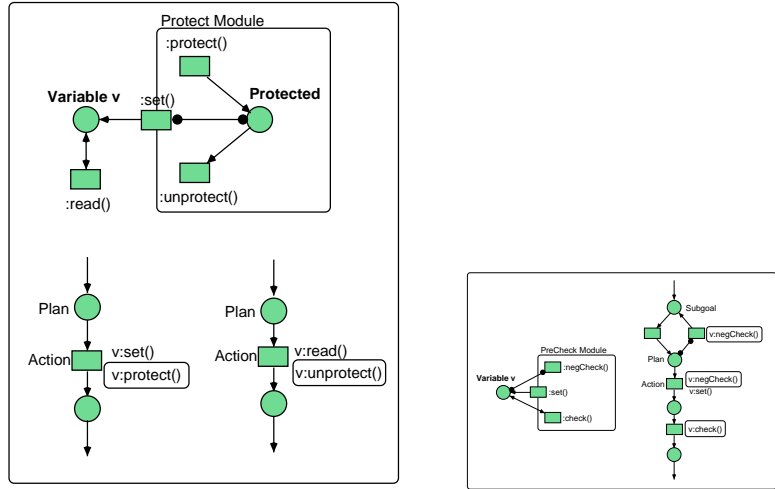
Prior to the time that the work by Thangarajah *et al.* was published, the Distributed Intelligent Agents Group led by Edmund Durfee, produced some similar research for modelling — and reasoning about — plan effects, extending their work to cover multi-agent systems rather than individual agents [6–8]. In their work, they were interested in reasoning about conflicts to coordinate the actions of agents that use HTN planning, while the work by Thangarajah was based around BDI agents (focusing on individual agents instead). In [7], Clement *et al.* present the summary information for pre-, in-, and post-conditions of plans, which is adopted by Thangarajah *et al.* and used within goal-plan trees to reason about both resources and effects.

3 Reasoning About Goals using Petri Nets

Petri nets are mathematical models, with an intuitive diagrammatic representation, used for describing and studying concurrent systems [14]. They consist of *places* that are connected by *arcs* to *transitions*, with *tokens* that are passed from place to place through transitions. Transitions can only *fire* when there are sufficient tokens in each of the input places, acting as pre-conditions for the transition. A token is then removed from each input place, and one is placed in each of the output places. Places are graphically represented as circles, while transitions are represented as rectangles.

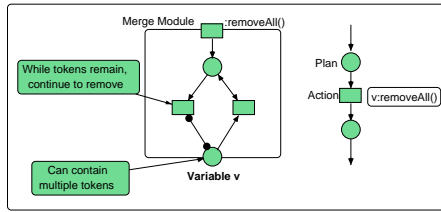
There are many variations on the basic Petri net representation, and many of these have been used in a variety of agent systems [13, 1]. Arcs can have weights associated with them, the default weight being one. Greater weights on arcs either require the place to have at least that many tokens for the transition to fire, or the transition adds to the output place that number of tokens as its output. Coloured Petri Nets are able to hold tokens of different types, representing for example different data types. The weightings on the arcs then match up and select the relevant tokens to fire. Reference nets allow nets to contain sub-nets. *Renew* is a Petri net editor and simulator that is able to support high-level Petri nets such as coloured and reference nets [12].

We have developed a method to represent an agents' goals and plans using Petri nets. Essentially, we are able to represent the same problems as expressed by goal-plan trees in the work by Thangarajah *et al.* (see Figure 2 for an example). According to the method we have devised, goals and plans are represented by a series of places and transitions. A plan consists of a sequence of actions that



(a) Protect module for negative interference.

(b) Pre-check module for positive interaction.



(c) Merge module for positive interaction.

Fig. 3. Petri-Net Representation of Modules for Reasoning about Goals.

change. When an agent executes a plan that produces an effect in the environment, and that effect will be required by a later plan, the effect is immediately marked as protected until it is no longer required. This is done by using a **protect** module (Figure 3(a)) that adds a set of transitions and places to the Petri nets so that when the relevant effect takes place, a transition is fired to protect it, then when it is no longer needed another transition is fired to **release** the protected effect. If another plan attempts to change something that will impact on the protected effects, then it will be blocked and forced to wait until the effects are no longer protected (i.e., until the release transition fires).

In the Mars Rover example, negative interference occurs when two or more goals require taking samples at different locations and after having moved to the first location, a second goal interferes to take the rover to another location before the sample is taken to satisfy the first goal. To avoid this, the causal link is identified based on the effects and preconditions of the plans when Petri nets are generated, and a **protect** module is added to ensure other goals and plans

cannot interfere with the causal link until the necessary plans have executed. In the Petri nets, the `protect` module is implemented by adding a place that holds a token to indicate if a variable (i.e., effect) is protected or not, with transitions that the plan fires to protect the variable at the start of the causal link, then another transition to `unprotect` the variable when it is no longer required.

The positive interaction reasoning checks whether the desired effects have already been achieved (such as a Mars rover going to a specific location to perform some tests), or whether multiple goals can all be achieved by a merged plan rather than a plan for each goal, such as the Mars Rover transmitting all the data back in one go instead of transmitting separately individual results obtained by separate goals. When two or more plans achieve the same effect, only one of the plans has to be executed. This can greatly reduce the number of plans that are executed, especially if one of the plans has a large number of subgoals and plans. As a result, this can speed up the completion and reduce the costs of achieving the goals, particularly if there is a limited amount of resources.

In the Mars rover example, positive interaction can take place in both ways. First, when moving to a different location the rover may have several goals all of which required going to the same location; however, only one plan needs to be actually executed to take the rover there. In the Petri nets, this is handled by a `pre-check` module (Figure 3(b)) that first checks whether another plan is about to, or has already, moved the rover to the new location, and if not it then fires a transition to indicate that the rover will be moving to the new location so the similar plans for other parallel goals do not need to be executed.

The second form of positive interaction is the direct merging of two or more plans. In the Mars rover scenario, this can occur when two or more goals are ready to transmit the data they have collected back to the base station. A `merge` module (Figure 3(c)) is added to indicate that when a goal is ready to transmit data back, it also checks to see if other goals are also ready to transmit their data. If so, all data that is ready is transmitted by the one plan rather than each goal separately executing individual plans to transmit the data.

4 Experimental Results and Analysis

We have used two different scenarios in our evaluation: the first is an abstract example and the other is the simple Mars rover example.

Scenario 1: Abstract Example

In this scenario, the goal structure in Figure 4 was used for each of the goals that were initiated. In the experiments reported here, we have opted for not considering varying structures, but this will be considered in future experiments. The experiments we conducted with Scenario 1 aimed to match, to the extent we could understand and reproduce, the settings of the experiments conducted in [19] to evaluate the GPT and summary information method that they introduced, in particular their experiments to compare the performance of JACK and X-JACK.

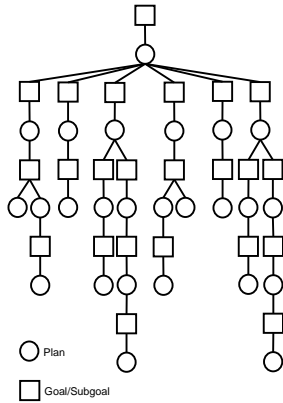


Fig. 4. Goal-Plan Tree Used for all Goals in Scenario 1.

In our experiments using Scenario 1, ten goal types were defined adjusting the selection of plans within the goal plan tree that would interact with those of other goals. The interaction was modelled through a set of common variables to which each goal was able to assign values. The variables and values are used to represent the different effects that plans can have in the environment.

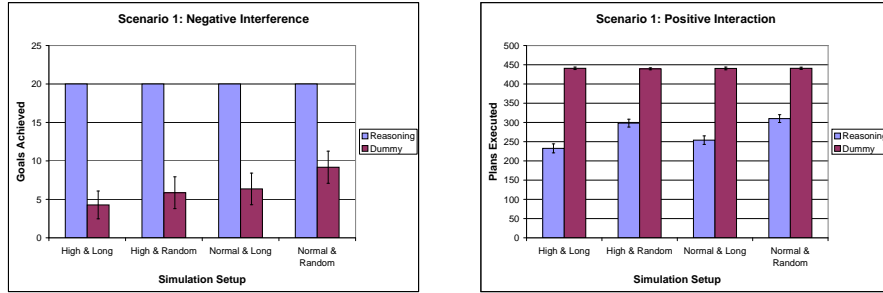
To stress-test the Petri nets, tests were set up that involved high levels of interaction, using a set of 5 variables, or low levels of interaction, using a set of 10 variables. Out of the 10 goal types, 5 of the goal types used 3 variables, while the remaining 5 goals types only altered 1 variable. During testing, 20 instantiations of the 10 possible goal types were created at random intervals and running concurrently. The Petri nets were implemented using Renew 2.1 [12], and each experiment was repeated 50 times.

Four experimental setups were used, with “High & Long” in the graphs (see Figure 5) corresponding to High Levels of Negative Interference for Long Periods, down to “Normal & Random” corresponding to Normal Levels of Negative Interference for Random Length Periods. The periods are controlled by defining the levels within the GPT that the interaction occurs at; so, for example, in the positive interaction, the duration over which the positive interaction takes place can be maximised by making plans in the top levels of the GPT with the greatest depth to interact.

A dummy Petri net was set up using the same goal structure and set of goal types, but without any of the reasoning for positive or negative interaction. The results from running this against the Petri net where such reasoning was included could then be compared to show the improvements obtained by the reasoning.

Negative Interference. Each goal was given a set of 1 or 3 variables to which it was to assign a given value and then use it (recall that this represents the effects of plan execution in the environment). The positions in the goals where the variables were set and then used were varied either randomly or set to require the variables to be protected for the longest possible periods (meaning the state of the world caused by a plan is required to be preserved for longer periods before the interfering plans can be executed). The selections of plans in each goal are designed to cause interference for other goals being pursued simultaneously. This is done by ensuring a significant overlap in the variables which the goals are setting, particularly under high levels of interaction. The effect of the reasoning is measured by counting the number of goals achieved both by the “dummy” and by the “reasoning” Petri nets.

The results are shown in Figure 5(a). The graphs show the averages for the number of goals achieved by the reasoning Petri net and the dummy Petri net



(a) Experimental results for negative interference.

(b) Experimental results for positive interaction.

Fig. 5. Results for Negative Interference and Positive Interaction in an Abstract Scenario.

from the 50 runs for each of the experiment settings, also showing the standard deviation. The effects of the negative reasoning are immediately obvious by the fact that the Petri nets with goal reasoning were consistently able to achieve all the goals, while the dummy Petri nets achieved, on average, very few goals, particularly when there were high levels of interference and variables that had to be protected for a long time, where it was only able to achieve approximately 21% of the goals, on average. Even at normal levels of interaction and random depth positioning, it was still only able to achieve, on average, 46% of the goals. The standard deviation shows that the performance of the dummy Petri nets was highly variable within the 50 runs of this experiment.

Positive Interaction. To measure the effects of reasoning about positive interactions, each goal was again given a set of 1 or 3 variables, with overlap between the goals, so that we can determine a selection of plans for each goal which can potentially be achieved by just executing one of the plans. Each goal contains 25 plans (in its GPT), of which at least 21 would have to be executed if the goal was being pursued on its own. This is due to two subgoals having a choice of plans to execute in the GPT. The scenario was set up to ensure all the goals are achievable without any reasoning, so the effects of the reasoning are measured by the number of plans that are required to execute in order to achieve all the goals.

As with the negative interference, the depth of the plans within the goal-plan structure at which merging can occur is varied. Plans with more subgoals will have a greater impact on the number of plans executed when merged than plans with no or very few subgoals. The tests were set with mergeable plans either high up in the GPT, or randomly placed within the tree.

The results are shown in Figure 5(b). The graphs show the averages for the number of plans executed by an agent using the Petri net with goal reasoning and a dummy agent; the averages are taken from the 50 runs for each of the experiment setups, and the graphs also show the standard deviations. There is

clearly a major improvement between the dummy and the reasoning agents in all of the simulation settings, with the reasoning agent requiring significantly fewer plans to be executed than the dummy, whilst still achieving the same goals. For high levels of interaction and mergeable plans at high levels in the GPT, there is an average drop of 47% in the number of plans being executed. Even with lower levels of interaction, and randomly placed mergeable plans, there is still a decrease of 30% on average. This could lead to large savings in the time and resources required by an agent to achieve its goals. While the standard deviation shows there is more variance in the performance of the reasoning agent than the dummy, this is due to the variations in depth and GPT of the merged plans. Even with the variance, the reasoning consistently caused a significant improvement in the performance in comparison to the dummy agent.

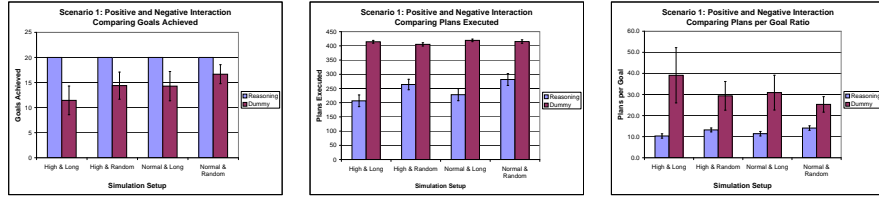
Negative and Positive Interaction. In this section, the two types of reasoning have been combined into one Petri net with a scenario that causes negative interference as well as it provides opportunities for positive interaction. To maintain exactly the same levels of interaction, both positively and negatively, the same GPT has been used again and the variables are duplicated for this abstract scenario. One set of variables is used for positive interaction, while the other is used for negative interference. This has been done, in the abstract scenario, to maintain the levels of interaction to allow for a clear comparison, but in the second scenario both forms of reasoning are applied to the same variables to represent a more realistic scenario.

Each goal is given 1 or 3 variables to assign values to for the negative interference, and we use the same number of variables for positive interaction. The number of goals achieved and the plans required are then measured to compare the expected performance of an agent that uses the Petri-net based reasoning against a dummy agent (i.e., an agent without any goal reasoning).

The four sets of tests were combined, in particular the negative interference at high levels of interaction over long periods was combined with the positive interference at high levels of interaction and at high levels within the GPT, while the negative interference at high levels of interaction over random periods was combined with the positive interference at high levels of interaction and at random levels within the GPT. The experiment for interaction at normal levels was combined in the same way.

The results are shown in Figure 6. These are broken down into three groups: 6(a) goals achieved, 6(b) plans executed, and 6(c) the ratio between plans executed and goals achieved. The standard deviations are also included in each of these graphs.

The reasoning agent is once again able to achieve all of its goals, while the dummy agent is still only able to achieve 57–83% of its goals. Not only is the dummy agent failing to achieve all its goals, it is also attempting to execute almost all its plans in an effort to achieve those goals. This means the effects of the positive interaction reasoning are also very obvious with a drop of 50% in the number of plans executed by the reasoning agent for high levels of negative



(a) Comparison of goals achieved across the four experimental setups. (b) Comparison of plans executed across the four experimental setups. (c) Comparison of ratio between plans executed and goals achieved.

Fig. 6. Experimental Results for Combined Positive and Negative Interaction in an Abstract Scenario.

interference with positive interaction for long periods in the GPT, while still maintaining a 32% decrease in plans at lower levels of interference. The *plan to goal ratio* shows that the reasoning agent only had to execute on average 10 plans at high levels of interaction, and 14 plans at lower levels of interaction, to achieve its goals, while the dummy agent had to execute on average 39 plans at high levels of interaction and 25 at normal levels. Recall that while in the GPT there are only 25 plans available to achieve the main goal on its own, the dummy agent was still executing plans in goals that failed, and the ratio shows all the plans executed compared to the goals actually achieved. The standard deviation shows that, in general, the performance of the reasoning agent is very consistent, whereas the dummy agent is highly erratic, particularly when there are high levels of interaction for long periods.

Scenario 2: Mars Rover

To show the reasoning being used in a more concrete example, a Mars rover scenario has also been used. In this scenario, the rover is given a set of locations and a set of tests (or tasks) to perform at each location. Each task at each location is represented by a separate goal, as shown in Figure 2, offering much opportunity for both negative and positive interactions. All of the plans contain a set of preconditions that must be true for it to be able to execute, and these preconditions are satisfied by the effects of other plans. So while there may be less plans involved than in Scenario 1, there is still a lot of interaction taking place. The preconditions lead to a partial ordering of the plans for the goal to be achieved. In our experiments, 2, 4, and 6 locations were used, with 5 tests carried out at each location, in order to evaluate the performance of the reasoning over different levels of concurrency, specifically 10, 20, or 30 goals being simultaneously pursued.

For the interests of comparison, the negative and positive reasoning have again been separated out before being combined together in the final set of experiments.

Negative Interference. Negative interference is caused when the rover goes to a location ready to perform its tasks, but is then interrupted by another goal that required going to a different location before the tasks required at the first location by the previous goal had been completed. The effects of the reasoning is again measured by the number of goals achieved. The results are shown in Figure 7(a).

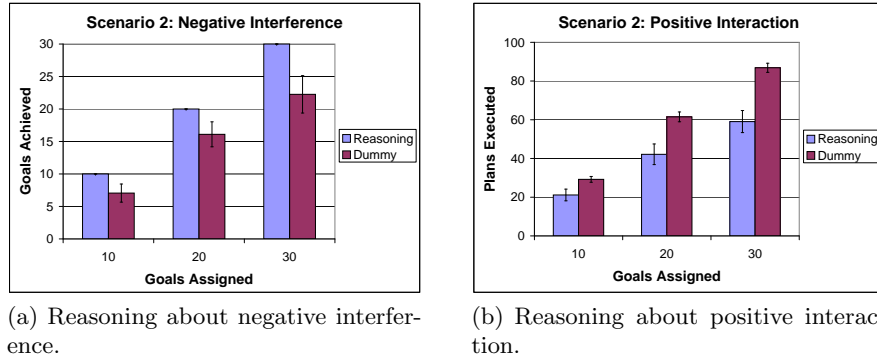


Fig. 7. Results for Negative Interference and Positive Interaction in the Mars Rover Example.

The results again show a definite improvement obtained by adding the reasoning about negative interference, whereby all goals were achieved, while the dummy agent is still only able to achieve on average 75% of its goals, across all the levels of goal concurrency, even at the lowest levels.

Positive Interaction. In the Mars Rover example, there are two main places for positive interaction. The first is when multiple goals all require the rover to perform tests/tasks at the same location, while the second is when the goals require transmitting their results back to the mission control team, after having performed the tests. When the goals have all obtained their test results, these can either be transmitted back to the base individually, or one goal can assume the responsibility of transmitting all the results back at the same time. This means only one plan has to be executed whereas without the reasoning an agent ends up executing one plan per goal.

The negative interference was removed from this setup to ensure all goals could be achieved without any reasoning. This meant the number of plans executed could be compared more fairly. The results are shown in Figure 7(b).

A clear reduction in the average number of plans executed can again be observed in these results, with higher levels of concurrency giving a 32% reduction in the number of plans executed to achieve the same goals. Even the lowest level of concurrency offers a 28% reduction that could be highly beneficial when there are many constraints imposed on an agent, such as time and resource availability.

Combined Negative and Positive Interaction. While both types of reasoning can be effectively used on their own, the combined effects of both types of reasoning give the best results, particularly in highly constrained conditions. In the final set of results reported here, we show the results of the combined reasoning about negative interference and positive interaction in the Mars rover scenario.

The results are shown in Figure 8. These are broken down into three groups: 8(a) goals achieved, 8(b) plans executed, and 8(c) the ratio between plans executed and goals achieved. The standard deviations are also included in each of these graphs.

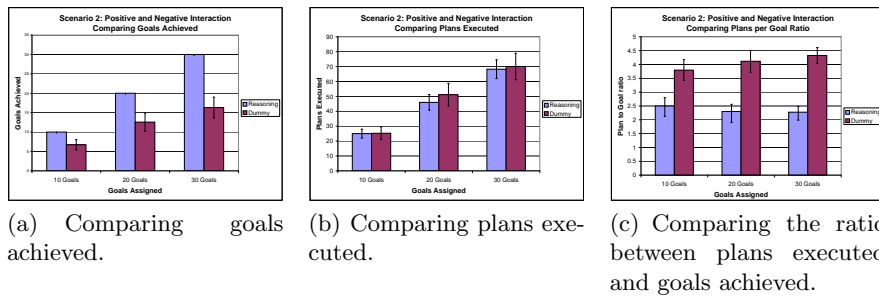


Fig. 8. Experimental Results for Reasoning about Negative and Positive Interaction in the Mars Rover Example.

While the results all show that there is only a slight improvement in the number of plans executed, the number of goals achieved by the reasoning agent is significantly more, and the *plan to goal ratio* is almost half that of the agent without any reasoning, increasing from a 34% reduction in the number of plans per goal to a 47% reduction as the level of goal concurrency increases. The reasoning agent is again consistently achieving all the goals it has been given, while the proportion the dummy agent was able to achieve dropped from 67% to 54% as the amount of concurrency increased. The standard deviation also shows that the reasoning agent is more consistent in its results in this scenario, with a lower range of variation.

5 Conclusions and Future Work

In this paper we have presented an alternative approach to reasoning about negative and positive interactions between goals. The results clearly show a significant improvement in the number of goals being achieved, and the number of plans required to achieve them. To the best of our knowledge, this is the first time the two types of reasoning have been combined together to show the joint effects of reasoning about both positive and negative goal interactions working

in tandem for an individual agent. As only a small extra computing cost is expected to result from the added reasoning, the benefits are very likely to outweigh any costs. However, in future work, we aim to analyse in detail the costs associated with the reasoning and compare this cost with alternative approaches such as a CSP representation and existing approaches such as the approach by Thangarajah *et al.* using a GPT [21–23]. In all experiments reported in this paper, such costs appeared to be negligible.

Preliminary work has been done in representing the same type of reasoning approached in this paper as a CSP, in order to provide further sources of comparison. A further type of reasoning that can be used to improve an agent’s performance is reasoning about resources, particularly when there is a limited supply of consumable resources available. We are currently in the process of including that type of reasoning in both our Petri-net and CSP-based techniques for reasoning about goals.

Currently, the Petri nets are being produced manually, but their modular design provides scope for automating this process, so that it can be incorporated into an agent architecture for on-the-fly reasoning about new goals to be potentially adopted. This will also be possible for the CSP-based approach, offering the agents a choice of reasoners if one proves to be better suited for particular situations (e.g., the structure/scale of the agent’s GPT, or specific properties of the environment) than the others. Our long-term objective is to incorporate such reasoners into the interpreters of agent-oriented programming languages.

Acknowledgements

We gratefully acknowledge the support of EPSRC’s DTA scheme. Many thanks to Berndt Farwer for recommending the Renew tool and the help in using it.

References

1. O. Bonnet-Torrès and C. Tessier. From team plan to individual plans: a petri net-based approach. In *proceedings of AAMAS’05, 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 797–804, New York, July 2005. ACM Press.
2. R. H. Bordini, A. L. C. Bazzan, R. de Oliveira Jannone, D. M. Basso, R. M. Viccari, and V. R. Lesser. AgentSpeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling. In C. Castelfranchi and W. Johnson, editors, *proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2002)*, pages 1294–1302, New York, USA, July 2002. NY: ACM Press.
3. R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multi-agent Systems, Artificial Societies, and Simulated Organizations. Springer-Verlag, 2005.
4. P. Busetta, R. Rönquist, A. Hodgson, and A. Lucas. JACK intelligent agents - components for intelligent agents in java. Technical report, Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia, 1998.

5. C. Castelfranchi and R. Falcone. Conflicts within and for collaboration. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting Agents: Conflict Management in Multiagent Systems*, Multiagent systems, Artificial societies, and Simulated organizations, chapter 2, pages 33–62. Kluwer Academic Publishers, 2001.
6. B. J. Clement and E. H. Durfee. Identifying and resolving conflicts among agents with hierarchical plans. In *proceedings of AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities, Technical Report WS-99-12*, pages 6–11. AAAI Press, 1999.
7. B. J. Clement and E. H. Durfee. Theory for coordinating concurrent hierarchical planning agents using summary information. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 495–502, Menlo Park, CA, USA, 1999. AAAI Press.
8. B. J. Clement and E. H. Durfee. Performance of coordinating concurrent hierarchical planning agents using summary information. In *proceedings of 4th International Conference on Multi-Agent Systems (ICMAS)*, pages 373–374, Boston, Massachusetts, USA, July 2000. IEEE Computer Society.
9. M. Dastani, F. de Boer, F. Dignum, and J.-J. Meyer. Programming agent deliberation: an approach illustrated using the 3apl language. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 97–104, New York, NY, USA, 2003. ACM Press.
10. M. Hannebauer. Their problems are my problems - the transition between internal and external conflict. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting Agents: Conflict Management in Multiagent Systems*, Multiagent systems, Artificial societies, and Simulated organizations, chapter 3, pages 63–110. Kluwer Academic Publishers, 2001.
11. J. F. Horty and M. E. Pollack. Evaluating new options in the context of existing plans. *Artificial Intelligence*, 127(2):199–220, 2004.
12. O. Kummer, F. Wienberg, and M. Duvigneau. Renew – the Reference Net Workshop. Available at: <http://www.renew.de/>, May 2006. Release 2.1.
13. H. Mazouzi, A. El Fallah Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 517–526, New York, NY, USA, 2002. ACM Press.
14. J. L. Peterson. *Petri Net Theory and the modeling of Systems*. Prentice-Hall, 1981.
15. A. Pokahr, L. Braubach, and W. Lamersdorf. A goal deliberation strategy for bdi agent systems. In T. Eymann, F. Klügl, W. Lamersdorf, and M. H. M. Klusch, editors, *Third German conference on Multi-Agent System TEchnologieS (MATES-2005)*; Springer-Verlag, Berlin Heidelberg New York, pp. 82-94. Springer-Verlag, Berlin Heidelberg New York, 9 2005.
16. A. Raja and V. Lesser. Reasoning about coordination costs in resource-bounded multi-agent systems. *proceedings of AAAI 2004 Spring Symposium on Bridging the multiagent and multi robotic research gap*, pages 25–40, March 2004.
17. P. Surynek. On state management in plan-space planning from CP perspective. In *In proceedings of Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems, International Conference on Automated Planning and Scheduling ICAPS, Cumbria, UK*. AAAI Press, June 2006.

18. P. Surynek and R. Barták. Encoding HTN planning as a dynamic CSP. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, Sitges, Spain*, volume 3709 of *Lecture Notes in Computer Science*, page 868. Springer, October 2005.
19. J. Thangarajah. *Managing the Concurrent Execution of Goals in Intelligent Agents*. PhD thesis, School of Computer Science and Informaiton Technology, RMIT University, Melbourne, Victoria, Australia, December 2004.
20. J. Thangarajah and L. Padgham. An empirical evaluation of reasoning about resource conflicts in intelligent agents. In *proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 04)*, pages 1298–1299, 2004.
21. J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 721–726, Acapulco, Mexico, August 2003. Morgan Kaufmann.
22. J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and exploiting positive goal interaction in intelligent agents. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 401–408, New York, NY, USA, 2003. ACM Press.
23. J. Thangarajah, M. Winikoff, and L. Padgham. Avoiding resource conflicts in intelligent agents. In F. van Harmelen, editor, *proceedings of 15th European Conference on Artificial Intelligence 2002 (ECAI 2002)*, Amsterdam, 2002. IOS Press.
24. M. B. van Riemsdijk, M. Dastani, and J.-J. C. Meyer. Semantics of declarative goals in agent programming. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 133–140, New York, NY, USA, 2005. ACM Press.
25. M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002), 22–25 April, Toulouse, France*, pages 470–481, 2002.