An Investigation into the Use of Collaborative Concepts for Planning in Disaster Response Coalitions

Clauirton Siebra and Austin Tate
Artificial Intelligence Applications Institute
School of Informatics, The University of Edinburgh
Appleton Tower, Crichton Street, EH8 9LE, Edinburgh, UK
{c.siebra,a.tate}@ed.ac.uk

Abstract

This paper investigates the implications of using concepts of collaboration as part of a planning architecture, which intends to support hierarchical coalition operations. Such concepts are mostly based on Teamwork approaches and they were integrated into the planning architecture via the same constraint-based framework, already in use by the architecture. The approach intends to maintain the planning and collaboration mechanisms independent of each other, providing a general rather than specific environment for the development of coalition support applications. Advantages, limitations and open issues of this approach are discussed through a practical demonstration in a disaster relief domain based on the RoboCup Rescue simulator.

1. Introduction

Coalition, from Latin *coalescere* (*co-*, together + *alescere*, to grow) is a type of organisation where joint members work together to solve mutual goals. One of the principal features of a coalition is the existence of a global goal, which motivates the activities of all coalition members. However, normally such members are not directly involved in the resolution of this goal, but in subtasks associated with it.

The use of intelligent planning as a resource to support coalition operations brings several advantages to these organisations, such as prediction of failures, resource allocation, conflict identification and so on. The planning process in coalitions is naturally distributed because each coalition member is a decision-maker. In this context, the use of hierarchies is a natural way to arrange coalition members in decision-making levels, where such members deal with different details and knowledge associated with a plan in development.

The *I-X project* [Tate, 2004] has created a planning architecture that can be applied to the configuration and support of hierarchical coalitions. I-X plans are specified according to <I-N-C-A> (Issues - Nodes - Constraints - Annotations) [Tate, 2003], a general-purpose constraint-based ontology that can be used to represent plans in the form of a set of constraints on the space of all possible plans in the application domain. The planning development is based on constraint manipulation and carried out as a two-cycle process (constraint addition and propagation), which aims to build a plan as a set of nodes (activities) with their associated detailed constraints.

This work investigates the integration of collaboration concepts into this architecture, discussing its implications, advantages and limitations. An important aim is to avoid additional requisites to the development of plans, so that existent I-X plans, for example, can take advantages of the new collaboration features without additional changes in their structures.

The remainder of this paper is structured as follows: Section 2 employs the I-X approach in a search and rescue domain called I-Kobe, which is based on the *RoboCup Rescue* simulator [Kitano and Tadokoro, 2001], highlighting the limitations of this application. Section 3 presents the *Teamwork Theory*, a formal framework for collaboration that can be used to cope with such limitations. Section 4 details how we are integrating the collaboration concepts with the planning architecture. Section 5 discusses the results of applying this collaborative version in the previous search and rescue domain, while Section 6 concludes with final remarks and directions.

2 I-Kobe

The I-Kobe application uses a disaster relief domain, based on the RoboCup Rescue simulator. The experiment discussed here focuses on the performance of a sub-

coalition Θ_p composed of one police office (operational level) and ten police forces (tactical level) during a period of 150 cycles, which corresponds to 150 minutes in the real world. The objective of Θ_p is to clear the roads that are blocked by collapsed buildings. A good performance of Θ_p is very important to the fire brigades, for example, because they need clear paths to quickly reach the fire points and water refill places.

The tactical agents use a simple plan. Each police force has a list of blocked roads, indicated by the police office, that is ordered by the closest distance from the blockage to the current agent position. Then, if an agent is clearing a road, it remains doing that until one of the passable lines becomes clear. Otherwise, it accesses its list to know the next blocked road. If the list is empty, the agent tries to find (search action) other blockages around the scenario.

Using the I-X architecture, agents are provided with a coordination structure where they can report execution, completion or failure of activities. In addition it is possible to implement handlers to deal with specific activities. For this experiment we have implemented a handler called "SimpleAllocation" that uses reports and information about the environment to generate an efficient delegation of activities to police forces. The results for this experiment are shown in follow (Figure 1).

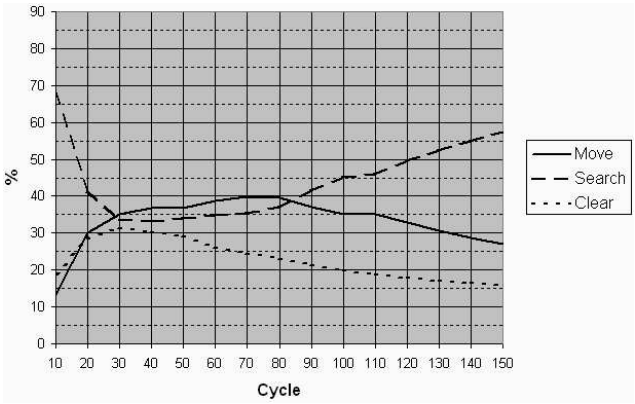

Figure 1. I-Kobe simulation results.

The curves in the graphic represent the average behaviour of the police forces. The *Move curve*, for example, has a peak around the cycle 70 and after that starts to decrease. This means that the police forces are mostly dealing with the delegated activities until the cycle 70 (they are going to blocked positions specified by the police office). The *Search curve* has the opposite behaviour, showing that the police forces are going back to search actions as soon as they complete the delegated activities. The experiment also highlights some limitations of this approach. The principal examples are:

- Police forces only report completion or failure of activities. Reports associated with activity commitments and progress are also important because they provide, for example, useful information to be used by handlers;
- In situations where the police office allocates a clear activity to *n* agents, *n* sub-nodes are created to represent such allocations. These nodes are typically examples of or-activities where only one of them needs to be completed for the overall clear activity be finished. However this does not happen in this experiment.

The next sections show how the design of the I-X planning framework can cope with these and other problems if such a framework is developed considering fundamental concepts of collaboration.

3 Teamwork as Basis for Collaboration

The teamwork research [Cohen and Levesque, 1991] involves a set of ideas that support the implementation of collaborative systems. *Joint Intentions* [Levesque et al., 1990] was the first teamwork proposal to formally define such ideas. A joint intention of a coalition Θ is based on its joint commitment, which is defined as a *Joint Persistent Goal* (JPG). A JPG(Θ,*p*,*e*) to carry out a proposition *p* while *e* is relevant, requires all coalition members to mutually believe that *p* is currently false and want *p* to be eventually true. In addition, a JPG ensures that coalition members cannot decommit until *p* is mutually known to be achieved, unachievable or irrelevant.

The Joint Intentions approach forms the basis to define when agents must communicate some important information (commitments and reports), collaborating in this way with each other. However we are considering two extensions of this work, which are proposed in the *Joint Responsibilities* [Jennings, 1992] and *SharedPlans* [Grosz et al., 1999] theories.

The Joint Responsibilities Theory extends the Joint Intentions ideas to include the notion of plan states. According to this theory, an important reason for explicitly distinguishing between the goals of activities and plan states becomes evident by examining what happens after the two types of commitment failure. In the former case, the team's activity with respect to the particular goal is over. However if the group becomes uncommitted to the common solution (a plan) there may still be useful processing to be carried out. For example, if the plan is deemed invalid, the agents may try a different sequence of actions which produce the same result. Thus dropping commitment to the common solution plays a different functional role than dropping a goal.

The SharedPlans Theory considers also important for collaboration, in addition to commitments and reports, the idea of mutual support. In this way, this theory defines an

intentional attitude (INT.TH, intention-that) which enables an agent to say to others which propositions need to hold so that its activities can be performed. Thus, such attitudes of an agent directly restrict the intentions that other agents adopt, affecting their planning reasoning.

4 Integration Analysis

We can analyse the integration of teamwork ideas with planning via an algorithm that considers the plan creation PLAN(μ,*p*) as one of its functions. For that end, consider that such an algorithm is carried out by an agent μ, member of a coalition Θ_x, that receive an activity p_i from its superior agent *sender*. This algorithm is codified via the "CollaborativePlanning" function in follow:

01. **function** CollaborativePlanning(*sender*,p_i)
02. *subplan* ← PLAN(μ,p_i)
03. **if**(∃*subplan*)
04. **if** (hasNodesToBeDelegated(*subplan*)) **then**
05. Delegate(*subplan*,*subordinates*) ∧ WaitCommits()
06. **if** ∃ *s* (*s* ∈ *subordinates*) ∧ (¬ commits(*s*)) **then**
07. **go to step 02**
08. **endif**
09. **endif**
10. Report(*sender*,p_i,**committed**)
11. Broadcast(Θ_z,*subplan.conditions*)
12. **while** (¬ Complete(*subplan*))
13. **if** (JustReady(*subplan*) ∨ Changed(*subplan*)) **then**
14. Report(sender,n_i,**executing**)
15. **else if** (Violated(*subplan*) ∨ Receive(**failure**)) **then**
16. **go to step 02**
17. **endif**
18. **end while**
19. Report(sender,p_i,**completion**)
20. **else**
21. Report(sender,p_i,**failure**)
22. ∀ s (*s* ∈ *subordinates*) ∧ HasCommitted(*s*,$subplan_s$)
23. Report(*s*,$subplan_s$,**failure**)
24. **endif**
25. **end function**

This function entails some implications. First the function tries to generate a *subplan* to perform p_i (step 02). If a subplan is possible (step 03) and it does not depend of anyone else (step 04) then the agent can commit to p_i (step 10). However, if *subplan* depends on the commitment of subordinates, then μ must delegate the necessary nodes to its subordinates and wait for their commitments (step 05). This means that commitments are done between a superior agent and their subordinates and, starting from the bottom, an "upper-commitment" can only be done if all the "down-commitments" are already stabilised.

Second, if some subordinate agent is not able to commit (step 06), μ returns (step 07) to generate other subplan rather than sending a failure report to its superior. Such a situation is similar to the cases where *subplan* is violated or μ receives a failure message of its subordinates (step 15). This approach implements the idea of enclosing problems inside the subteam where they were generated.

Third, if μ is not able to generate a subplan for p_i, it reports a failure to its superior (step 21). In addition, it must also alert their subordinates that p_i has failed and consequently its subnodes can be abandoned (steps 22 and 23).

After reporting a commitment (step 10), μ must monitor and report execution status until the completion/failure of p_i. Progress reports are associated with changes in the plan, which are monitored and sent to superior as an ongoing execution report (step 13). Constraint violations and failure messages are also monitored (step 15) so that μ firstly tries to repair the problem by itself (step 16) before sending a failure report. Note that, using this function, any activity *p* will have one of the following status: *no-ready*,*possible*,*impossible*,*complete* and *executing*. The <I-N-C-A> definition for activities contains a status attribute that can be filled with one of these options.

We must note that, according to the Joint Intentions theory, if μ finds out a problem in *subplan*, all the commitments previously associated with *subplan* should be cancelled. Differently, the Joint Responsibilities Theory states that if Θ_x becomes uncommitted to *subplan*, there may still be useful processing to be carried out. We are using this idea when μ tries a new subplan (steps 07 and 16).

A last step that must be explained is associated with the idea of mutual support. The principal idea behind mutual support is to enable that one agent has knowledge about the needs of other agents. For example, μ knows that a specific road is clear so that it uses this constraint in its plan. However, as the world is dynamic, the road becomes blocked. If any other agent finds out that such road is no longer clear, it must inform this fact to μ. Thus, this informer agent is supporting the performance of μ.

An easy option to implement this feature is to force that agents broadcast any new fact to all coalition. Consequently all agents will have their knowledge base updated and problems like that can be avoided. However, this is not a good approach in terms of communication and agents will also receive much useless information.

Consider now that *subplan* of μ ($\mu \in \Theta_x$) has a set of conditional constraints *C*, which μ desires to hold so that *subplan* is still valid. In this case, each $c_i \in C$ is a constraint that μ believes to be true and hopes that it is still true. Then μ broadcasts *C* (step 11) for every agent $\mu_j \in \Theta_x$ so that other agents of its subteam know what it needs. A function based on this idea, and applied by agents that receive *C* from μ, is defined as:

01. **function** MutualSupport(μ,C)
02. **while** ($\exists c_j \ \ c_j \in C$)
03. **if** ($\exists c_j c_k \ \ c_j \in C \wedge c_k \in$ BEL(μ_j) $\wedge$ Conflict(c_j,c_k)) **then**
04. *newactivity* $\leftarrow$ CreateActivity(Goal(c_j))
05. **if** ($\neg\exists$ *newactivity*) **then** Inform(μ,c_k) **endif**
06. Retire(c_j,C)
07. **endif**
08. **if** ($\exists c_j \ c_j \in C \wedge \neg$Valid($c_j$)) **then**
09. Retire(c_j,C)
10. **endif**
11. **end while**
12. **end function**

According to the function, each agent μ_j must compare its beliefs BEL(μ_j) with C (step 03). If μ_j finds some *conflict*, it must try to create a new activity whose goal is to turn c_j true (step 04). If this is not possible, μ_j must inform μ that c_j is no longer holding and its new value is c_k (step 05). The idea implemented by this function is simple, however there are two more complex points: the "Conflict" and "Valid" functions.

The Conflict function (step 03) is an extension of the Violated function (step 15, CollaborativePlanning function). A violation is a type of conflict between two constraints. It says that two constraints, which are supposed to match, are not matching. However we are also considering as conflict the situation where two constraints have the potential to be identical. For example, ((colour Car),?x) and ((colour Car),blue). In this case the two constraints are in conflict because they have the potential to be identical if the variable ?x assumes the value "blue". This type of conflict is very useful in the following class of situations. Suppose that one of the activities of μ is to rescue injured civilians. For that end, μ firstly needs to find such civilians so that it has the following conditional constraints: ((position ?a),?b), ((role ?a),civilian) and (status ?a),injured). This set of constraints implies that the variable ?b is the location of an injured civilian ?a. Then if other team agents that have or discover a set of constraints that conflict with the set sent by μ, they must inform μ about this new knowledge (note that in this case no make sense to create a new activity).

The Valid function (step 08) accounts for eliminating the constraints that no longer represent conditions to μ. This is important to avoid that μ still receives useless information and also to decrease the number of messages in the coalition. A practical way to do that is to consider that all $c_j \in C$ has a timestamp that indicates the interval where such constraint is valid.

Using the timestamp (t_i,t_f) and considering that t_i and t_f are ground values, the Valid function only needs to compare if the condition ($t_f <$ current-time) is true to eliminate the respective constraint. However this timestamps are not useful if agents do not know when their activities finish because such a temporal value will be a variable. Note that the principal advantage that we are looking for in using timestamps is to avoid that agents (C's senders) need to broadcast the information that they no longer need that a group of constraints holds. Rather, timestamps enables agents (C's receivers) to reason by themselves on the elimination of such constraints. An alternative for timestamps, which we intend to investigate, is to link constraints, not to real world time, but to the partial ordering of the plan elements.

One of the principal advantages of the MutualSupport function is that it improves the information sharing in Θ_x because the sending of information is guided by the constraint-based knowledge that each agent has about the activities of its partners. In addition, it can also be used as a method to avoid conflict between activities because agents know which external constraints must be respected.

5. I-Kobe: a Collaborative Version

In the last experiment, the first report is sent when agents start the execution of their activities. In this new version, the first report is generated as soon as a plan is created (commitment). Note that if there is a long period between the plan generation and the plan execution, the police office will also spend a long period unsure about the status of this activity.

This new version also compels police forces to send progress updates, if some plan information has changed. In this experiment, when a police force *pf* commits to the performance of an activity *ac*, it also sends the cost of *ac* to its police office. The cost here is given by the time, in domain simulation cycles, that *pf* will spend to reach the blockage place, plus the time to clear such blockage. However this cost can change due to, for example, problems in the path and wrong estimations (e.g., *pf* usually does an estimative of the time to clear blockages in the moment of the commitment because it has not seen the blockage yet). As the allocator handler uses the cost values during the process of delegation, progress updates help it in keeping its allocation table in accordance with the real situation of the police forces, improving the process of allocation.

Together with the commitment mechanism, we have also introduced the notion of mutual support into this experiment. The mutual support function plays an useful role during the simulation. When a police force receives an activity to clear a road, it shares the conditions to clear this road. One of these conditions is that the road is actually blocked. If other agent of the coalition has an information that contrasts with this condition, it must inform to the police force.

This process indirectly resolves the problem of or-activities discussed in the last experiment. If two police forces pf_1 and pf_2 receive the same activity to clear a road and pf_1 finishes such activity before pf_2 has started its execution, the new status of the road (status road = clear) will

contrast with the conditional constraint sent by pf_2 to pf_1, so that pf_1 informs this new status to pf_2 (note that both agents have received the conditional constraints of each other). Then, pf_2 automatically reports the completion of its activity to its police office. Using this mechanism, the police forces become available faster and the allocator has more options to perform its allocations.

On the other hand, this experiment highlights a potential problem. According to the mutual support function, before pf_1 informs the new status of the road, it must try to create an activity that turns the condition true (status road = blocked). This does not happen in this experiment because police forces do not have this capability. But, in a general way, conditions that are negations of goals can generate problems, so that the CreateActivity function (step 04, MutualSupport function) must consider this exception[1].

If we calculate the integral of the Clear curve (Figure 2) for this experiment, the resultant value is almost the same as in the last experiment. However in this case we are sure that the clear actions are associated with the requests of the police office because such a curve follows the behaviour of the Move curve. In other words the police forces are moving to the blockages indicated by the police office. Note that there are two perspectives in which we can analyse the efficiency of Θ_p. From the Θ_p's perspective, such subteam is efficient if they are able to clear a big number of roads. From the perspective of the coalition as a whole, which is the focus of this experiment, Θ_p is efficient if they are able to clear the necessary roads. Thus, rather than a quantitative result, we are interested in a qualitative measure on the performance of clear actions.

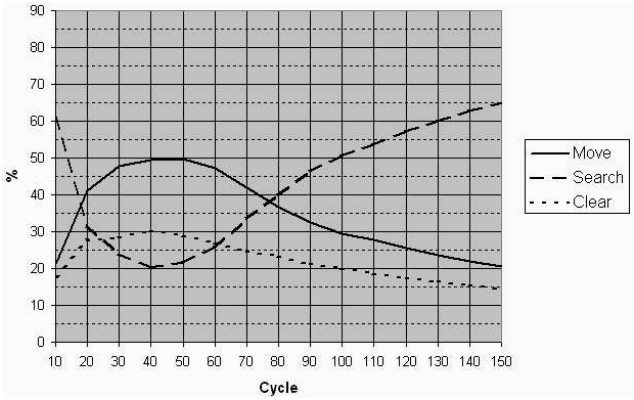

Figure 2. Collaborative version results.

If we compare this graphic with the graphic of the first experiment (Figure 1), we can also notice that the Move and Search curves are more regular and narrower. This indicates that the police forces finish their delegated activities faster than the first experiment, returning to their original action of searching blockages by themselves.

[1]This is a common problem in AI planning and planners often use "constraint types" to indicate which constraints are only intended to be tested and which are intended to be targets to be achieved [Tate, 1995].

6. Final Discussion and Directions

A deficiency in the current planning literature is the lack of discussions about the amount and kind of knowledge that each agent of a coalition must maintain about the coalition's activities so that they mutually support one another. According to our approach to mutual support, we are arguing that the knowledge about the conditions required by each agent is appropriate to enable such support. However, based on our experiments, it is not possible to demonstrate the real efficiency of such an approach. A detailed investigation of this issue could be done via measures of the usefulness and usability of the knowledge, in our case the activities' conditions shared through the coalition.

Other practical matters associated with the mutual support approach are: the elimination of useless information, the number of messages and the post-conflict decision process. As discussed previously, the solution applied to eliminate conditions is to stamp a timeline in each constraint saying the period that it should be considered valid. However such a solution was not very useful in our application because the majority of the activities did not have a defined timeline (start and finish times).

Another pertinent problem appears when an agent abandons an activity. In this case its conditional constraints are no longer valid, but as they were shared into the coalition, they are still generating unnecessary reasoning and performance of activities. Thus, the development of a process like a team *garbage collection*, applied to unnecessary constraints, could be appropriate to avoid collateral effects.

Concerning the number of messages, the experiments have demonstrated that the mutual support function is likely to require considerable communication. The idea of filter algorithms could be applied to this problem, avoiding that an agent sends its conditional constraints for all agents of its (sub)team.

The post-conflict decision process is another possible reason for low efficiency. Consider the following scenario: an agent a_1 generates a plan p_1 with a conditional constraint c_1, which is shared into the coalition. Meanwhile, an agent a_2 is trying to generate a plan p_2, however its possible plans are in conflict with p_1. According to the simple post-conflict decision process that we are applying, all the agents must consider the constraints already shared. Thus, a_2 will not be able to complete its activity. This problem becomes worse if the performance of a_2 is critical to the coalition aim. In this case a_1 should replan its activities, eliminating c_1 and enabling the generation of p_2 by a_2. We can conclude that the

simple use of time is not adequate for the post-conflict decision process and the priority of the activities is an important attribute that must be considered during such process.

The definition of possible extensions for this work is directly indicated by such limitations. These extensions are listed below:

- Development of experiments that measure the usefulness and usability of conditional constraints, considering the process of mutual support. The idea is to investigate, from the set of all constraints received by an agent, which of such constraints are useful for the different processes provided by the mutual support approach (conflict resolution, information sharing and activity generation). Based on the results of this experiment, we could also be able to know which information, other than conditional constraints, is important to agents. If we apply such experiments to all the hierarchical levels, we can produce the basis for supplying the lack in the current planning literature previously cited;
- Study and implementation of mechanisms that enable the elimination of knowledge which is no longer valid from the coalition. Rather than agents exchanging messages saying which information must be eliminated, agents should be able to reason about such elimination by themselves. An interesting metaphor, used in the previous section, is to think about this process as a garbage collection used for some object-oriented languages. In Java, for example, each virtual machine uses a specific rule (there are no longer any references to an object) to eliminate unnecessary objects. In the same way, we could implement some rule in each agent so that they eliminate unnecessary knowledge;
- Specification and test of a post-conflict decision process so that it considers the idea of priority. In fact the <I-N-C-A> ontology already provides a representational attribute for priority in the activity definition. Thus we could use this attribute to decide which agent must replan in case of conflict.

The extension of our search and rescue domain to three levels of decision-making (strategic, operational and tactical), together with the implementation of a new different domain associated with space applications [Siebra et al., 2004] are also themes for future works.

7. Acknowledgement

Scholarship for Clauirton Siebra is sponsored by CAPES Foundation under Processes No. BEX2092/00-0. This material is based on research within the I-X project, sponsored by the Defense Advanced Research Projects Agency (DARPA), US Air Force Research Laboratory under agreement number F30602-03-2-0014, the EPSRC-Sponsored UK Advanced Knowledge Technologies Project, and others sources.

The University of Edinburgh and research sponsors are authorised to reproduce and distribute reprints and on-line copies for their purposes not withstanding any copyright annotation here on. The views and conclusions contained here in are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of other parties.

References

[Cohen and Levesque, 1991] Cohen, P. and Levesque, H. (1991). Teamwork. *Special Issue on Cognitive Science and Artificial Intelligence*, 25:487-512.

[Grosz et al., 1999] Grosz, B., Hunsberger, L. and Kraus, S. (1999). Planning and Acting Together. *AI Magazine*, 5(2):23-34.

[Jennings, 1992] Jennings, N. (1992). Towards a Cooperation Knowledge Level for Collaborative Problem Solving. *Proceedings of the Tenth European Conference on Artificial Intelligence*, Vienna, Austria, 224-228.

[Kitano and Tadokoro, 2001] Kitano, H. and Tadokoro, S. (2001). RoboCup Rescue: A Grand Challenge for Multiagent and Intelligent Systems. *AI Magazine*, 22(1):39-52.

[Levesque et al., 1990] Levesque, J., Cohen, P. and Nunes, J. (1990). On Acting Together. *Proceedings of the Eighth National Conference on Artificial Intelligence*, Los Altos, California, USA, pp.94-99.

[Siebra et al., 2004] Siebra, C., Tate, A. and Lino, N. (2004). Planning and Representation of Joint Human-Agent Space Missions via Constraint-Based Models. *Fourth International Workshop on Planning and Scheduling for Space*, Darmstadt, Germany, pp.180-188.

[Tate, 1995] Tate, A. 1995. Integrating Constraint Management into an AI Planner. *Journal of Artificial Intelligence in Engineering*, 9(3):221-228.

[Tate, 2003] Tate, A. (2003). <I-N-C-A>: an Ontology for Mixed-Initiative Synthesis Tasks. *Proceedings of the IJCAI Workshop on Mixed-Initiative Intelligent Systems*, Acapulco, Mexico.

[Tate, 2004] Tate, A. (2004). I-X: Technology for Intelligent Systems. `http://www.aiai.ed.ac.uk/project/ix/`