KSCO 2006

Knowledge Systems for Coalition Operation

Jeffrey M. Bradshaw
Michal Pěchouček
Niranjan Suri
Austin Tate (eds.)

Jeffrey M. Bradshaw
Michal Pěchouček
Niranjan Suri
Austin Tate (eds.)

Knowledge Systems for Coalition Operations

Prague, June 2006

Cognitive Agent-based Approach to Varying Behaviours in Computer Generated Forces Systems to Model Scenarios like Coalitions

Martyn Fletcher
Agent Oriented Software Limited
martyn.fletcher@agent-software.co.uk

Abstract

In this paper we describe a general approach for dynamic variation in behaviour of entities in military-based Computer Generated Forces (CGF) system scenarios. A typical scenario might represent the coalition of military forces from two countries to provide suppression of an enemy's air defence capability. The entities in the scenario are cognitively-plausible models of individual people and hierarchical groups of people, and are modelled using intelligent software agents. The cognitive agents' Situational Awareness (SA) of the ongoing battle is affected due to the imposition of various moderating factors such as fatigue, fear and stress. This variation of SA impinges on the agents' selection and execution of tactics in response to the observation of threats and so provides the CGF system with a higher degree of realism. This work is central to the development of the next generation of Computer Generated Forces systems that are to be inherently adaptable, 'intelligent' and dynamically re-configurable in order to cope with the fast changing demands of military Operational Analysis, procurement and training environments. The paper discusses the key features of these cognitive agents' variability through a military demonstration.

1. Introduction

The technology advanced via the Human Variability in Computer Generated Forces (HVCGF) project – funded by the UK Ministry of Defence – is creating a technological environment for modelling the cognitive and military complexity of scenarios such as the coalition of combat forces from multiple countries [10, 11] for a particular mission or the linkage between military and non-military organisations for certain purposes such as Operations Other Than War [6, 7, 8]. This technology is needed because there is relatively little work on realistically representing how military people make decisions when they are exposed to various moderating influences. The technology is intended to be both plausible [1], from a UK military perspective, and be sufficiently sophisticated to illustrative the effects of variation in the selection and execution of behaviour. Behaviour is defined as the spectrum of doctrinal (prescribed) tactics and non-doctrinal tactics that can be undertaken by an individual combatant. The choice of behaviour is based on the quality of their Situational Awareness (SA) of the battle. SA is the understanding military personnel have of their current status, the goals of the specific mission and overall political context, the positions of their comrades, and the tasks assigned to them, the location and suspected intentions of enemy units, and operational parameters such as the Rules of Engagement and Emission Control (radio use) policies to apply. Such variation of SA in a soldier, sailor or aircrew can be changed due to the excursion of various types of moderating influences:

- Factors external to the human body to reflect challenging environmental conditions, e.g. heat.
- Internal factors such as the presence of fatigue, fear, sleep depravation and the ingestion of stimulants like caffeine.
- Task-dependent factors like stress generated by expectation of the likely outcome of the forthcoming mission (pre-task appraisal) or the lack of suitable training for the mission.

Exerting influences on a person over time change the management of their SA, their decision-making and their interactions with other people. As an example, a well-rested unstressed soldier operating in a tolerable environment and performing some military task that he has suitable experience of (e.g. manning a road-block) can recognise a vehicle, correctly determine whether it is a potential threat, decide to take appropriate doctrinal actions according to his

training, and appropriately inform his superiors with situation reports (that are brief, accurate and clear), and give orders to his subordinates. However after being on duty for a long night in cold, wet and dangerous conditions, his fatigue has increased and so he might mistake the vehicle or even not notice it until it is much closer and take non-doctrinal action (e.g. immediately fire on vehicle without prior recourse to Rules of Engagement) and instigate sub-optimal interactions.

Today's military-based CGF systems [14] cannot adequately model this behaviour because: (i) they only use doctrinal tactics when modelling entities; and (ii) the CGF entity's behaviour cannot be easy modified at runtime. Having a CGF system augmented with cognitive agent [12] and variation technology [2] is a major benefit to military Operational Analysts and designers of training programmes due to the power it provides them with to realistically model the behaviour of the enemy forces and so provide a richer and more relevant simulation or training experience. The technology is becoming a key resource for the next generation of CGF platforms as they strive to be adaptable, decentralised, 'smart' and re-configurable.

The 'proof of concept' for this technology has been constructed through the HVCGF project [3, 4, 5] which is sponsored by the UK MoD's Directorate of Analysis, Experimentation and Simulation. The project team consists of Agent Oriented Software (AOS), QinetiQ and Penn State University. The technology is geared towards experimenting with, and demonstrating the benefits of, cognitive agent-based control of entities (e.g. rotary wing aircraft, artillery units, dismounted infantry and so forth) moving and operating in the CGF system's scenario. The HVCGF project aims to deliver:

- Plausible models of how war-fighters' cognition and behaviour moderation work which can be then verified via psychological experimentation.
- A software system to experiment with variable cognitive modelling techniques, which can be integrated with a wide range of CGF systems.
- A military demonstration, showing realistic effects of moderators via behaviour of teams of entities [9] in the OneSAF Testbed Baseline (OTB) CGF.
- A scientific demonstration, showing plausible variability of cognitive models using the serial subtraction task [16].

Cognitive agent-based control of a CGF system [13] is a methodology based on the application of autonomous cooperative building blocks (i.e. cognitive agents and situation mapping functions between CGF and agent worlds) that can be put together into an organization to manage a battle simulation scenario running in the CGF. The organisation consists of an infrastructure and a framework (both application-independent) and application-specific models of the simulated battle.

The CoJACK agents were encoded using the JACK™ platform through the inclusion of cognitive overlays in order to supplement the agent's Belief/Desire/Intention model of rational behaviour with cognitive and psychological features that mimic how people think. The inclusion of such overlays provides the framework that empowers the cognitive/tactical modeller to define and use a set of memory values, called *cognitive attributes*, inside the CoJACK agent's tactical plans to modify the process of selecting and executing these plans. By using cognitive attributes the modeller can reflect how people have adjustable reaction times to observations, and have limitations on the capacity of their working memory so only a few 'important' goals/threats are focussed on at any one time. Also when trying to recall some fact, a delay is incurred and a mistake can be made in remembering the fact's value.

In the paper, we begin with some background on the role of cognitive computing and CGF platforms in the UK military. In Section 3, we describe an architecture that is currently under development to support dynamic, intelligent reconfiguration of distributed cognitive-agent empowered CGF systems as a coherent technology. In section 4, we provide an illustrative example of our approach in a military setting.

2. Background

Nowadays, organizations that develop and run military simulations within the UK are facing a rapidly increasing demand from training managers and operational analysts for the ability to model people within their simulations to realistically reflect how military personnel behave and make decisions as well how they are affected by moderating factors such as heat, fear, stress, cultural/gender factors and the affect of suitable training. In other words, they are being asked to provide cognitively plausible computational models of people that have been configured to uniquely reflect how a particular person builds up and manages their Situational Awareness, and then select and execute tactics/behaviours according to their SA

and their individual cognition. Also the proliferation of Synthetic Environments into the process of acquiring, training, using and de-commissioning military equipment means that military procurement and training organisations have to more closely integrate their CGF platforms and models with new tools to reflect how people are at the centre of such processes.

As part of this drive to reflect how people are at the focal point of CGF system development, military organisations are drawing on research results from both (i) psychology and cognitive science to provide insights into how people make decisions, and (ii) from computer science (in particular agent technology) for encapsulating how these cognitive processes can be implemented within a IT system. From cognitive science, a classic experiment to explore the impact of moderating influences on a person's cognitive processing is the serial subtraction experiment. Through performing this task with people, experimental results have been collected (and widely published in the psychology literature) and computer-based models to reflect this cognitive processing – using cognitive modelling architectures like Act-R and Soar – have been constructed and the experimental results validated. Similarly CoJACK is a framework to build cognitive models and so validation of CoJACK's operation in comparison with the serial subtraction human results has been done as a scientific demonstration of cognitive agent plausibility [1].

The cognitive mechanisms of serial subtraction are relatively well understood and data is available for validation purposes. Starting from some large number (e.g. 7492), repeatedly subtract a small number (e.g. 8) for some period of time (e.g. 6 minutes). If the subject makes a mistake, they are corrected, and continue the task from the correct value. Competing strategies are: (i) Subtract from the units column, borrowing from each column to the left as needed; or (ii) subtract ten, then add the appropriate 10's compliment value (e.g. 2 is the 10's compliment of 8) to give an answer. This serial subtraction cognitive operation has been implemented in CoJACK with a cognitive capability being introduced to inject timing and errors into the reasoning as a consequence of accessing the cognitive attributes associated with each possible subtraction. The modeller specifies (explicitly inside the CoJACK agent's plans that implement the subtraction strategies) when the timing and errors are introduced as a result of accesses to the available cognitive attributes.

Cognitive parameters within the CoJACK agent (e.g. latency) impact the delay and accuracy of reads/writes to the cognitive attributes. Moderators impact the base levels of these cognitive parameters, often in a temporal manner. The impact a moderator has on the agent's cognitive parameters is specified via a *sensitivity function* which has an uptake and an excretion cycle into reservoir(s) and a mapping of reservoir values to the cognitive parameters. Also an optional *exposure* gives opportunity to apply moderators not explicit in the environment, and negative time means exposure prior to commencement of the simulation run.

Moderators directly affect cognition, perception and action of individuals. Variations in the behaviour of individuals results in variations in the behaviour and performance of the teams these people are involved with. When modelling at team level, the modeller must keep track of the following factors: (i) moderators affect entities differently (because of different base levels); and (ii) different team compositions can compensate for variations in the behaviour for individuals. Figure 1 shows a Graphical User Interface that could enable the modeller to define moderator models; in this example the moderator is caffeine and two moderator models can be constructed (one to reflect how caffeine affects someone who is relaxed but alert, the other reflects influences on someone who is already sleepy) in terms of their impact on the 'latency factor' reservoir over a time period (i.e. up to 15000 seconds from scenario start-up).

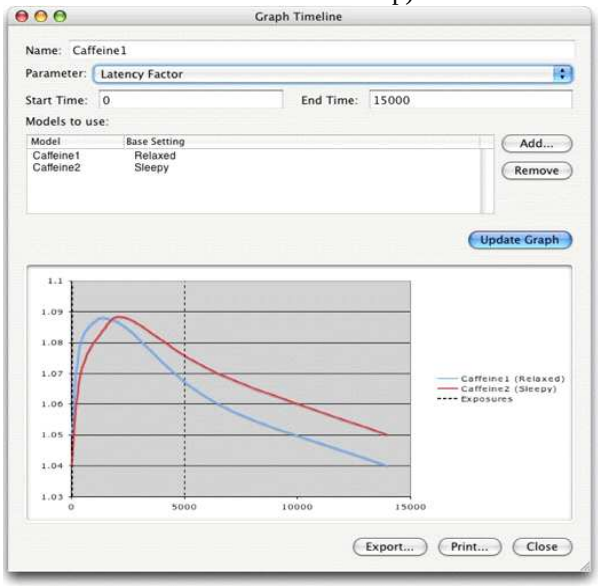


Figure 1: Screenshot of GUI used to set-up CoJACK moderator parameters

The modeller can then observe the changing behaviour of the CoJACK agent during the task as a

consequence of applying the moderator. For example by exposure to fatigue, access to the cognitive attributes associated with each step of the subtraction process is delayed leading to more effort on each calculation (observed as more time taken to perform the subtraction) and "confusion" when attributes are incorrectly retrieved (observed as generating more wrong answers).

3. System architecture

The following software components are being implemented to form the System Architecture, as illustrated in Figure 2.

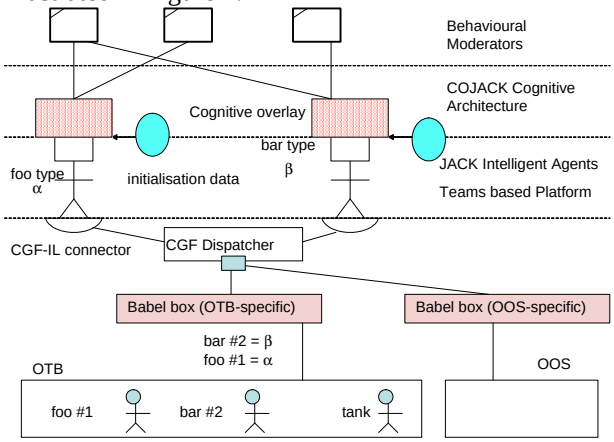


Figure 2: System Architecture

First, a set of *Behaviour Moderators*, each representing the influence of moderators on practical reasoning that can impact the agents' psychological attributes.

Second, a CoJACK *Cognitive Architecture* for enhancing the functionality of the agents with psychological attributes that can be varied in order to implement practical reasoning. These extensions provide an 'easy to use' framework for developing cognitively-plausible agents that simulate human psychological parameters such as the capacity of working memory to handle multiple concurrent military tasks, reaction speeds to events and perceptions coming from the battle-space, and the number/complexity of tactics for problem solving within the battle-space, e.g. the soldier's ability to choose from several ways to execute a Tactical Advance to Battle (TAB).

Third, a *JACK™ Intelligent Agents and Team-based Platform* to support the design, configuration and execution of a collection of rational agents and teams of agents organised into realistic UK military structures that, within the scope of the cognitive values set by CoJACK, can act intelligently. These actions relate to executing (non)doctrinal behaviour tactics that are encoded as JACK™ graphical plans and interface to entities inside the CGF system, e.g. vehicles managed.

Finally, a lightweight generic interface layer, known as the CGF Interconnection Layer (CGF-IL) facilitating integration of CoJACK with the variety of CGF systems used by MoD. Via the CGF-IL, each agent issues commands and receives events to/from CGF entities using private views of the CGF world. CGF-specific BabelBoxes convert generic commands and data used by CoJACK agents into CGF formats.

4. Demonstration in a military setting

The Military Demonstration has three key goals:

1. To show the use of more realistic human behaviours through a high-level cognitive model of military personnel. This includes demonstrating agents' situation awareness and showing that the selection and execution of doctrinal and non-doctrinal behaviour can vary based on the quality of SA maintained by the CoJACK agent.
2. To represent teamed human behaviour and variance of decision-making at the Command and Control (C2) level.
3. To show the effects of moderators, e.g. fatigue, on behaviour and thus on military effectiveness.

All of these aspects are to be demonstrated within the limits and fidelity of the nominated CGF system, i.e. the OneSAF Testbed Baseline (OTB) environment. Many scenarios were investigated to explore these goals. The scenario selected is a multiple Attack Helicopter (AH) mission – with AHs possibly from multiple countries in a coalition – that is seen to have both high NATO military relevance and operational feasibility, and can avail itself from considerable input on use of doctrinal tactics from UK Subject Matter Experts (SMEs).

The scenario is based upon a team of six Attack Helicopters attacking a ground target within enemy territory. The AHs are to: take-off from a Forward Arming and Refuelling Point (FARP); fly to a target area; locate, engage and destroy the target (a motorised rocket artillery regiment); and return home. On the ingress and egress, the AHs may encounter air-based and ground-based threats (e.g. a SA-8 surface-to-air missile installation or a MiG-29 fixed wing aircraft) that they have to deal with, and may be re-tasked by a Mission Commander onboard an AWACS aircraft monitoring the Theatre of Operations to engage

different targets while in the air (e.g. destroy a SCUD). The schematic of this mission is – as scenarios in a CGF are shown as icons on a map layout – in Figure 3.

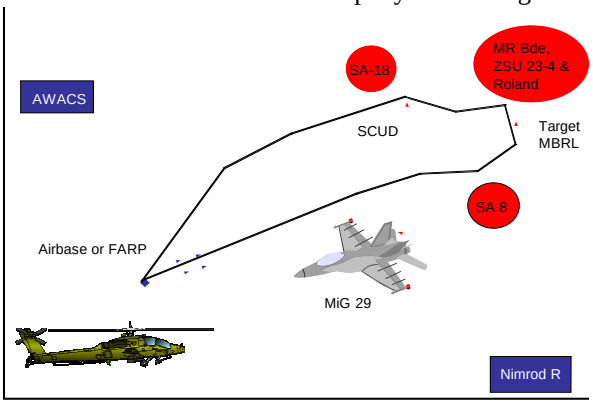


Figure 3: Layout of the AH Mission

The demonstration uses teams of CoJACK agents to control the AH entities that are being played out inside OTB in order to capture the very wide scope for modelling human variability throughout this mission.

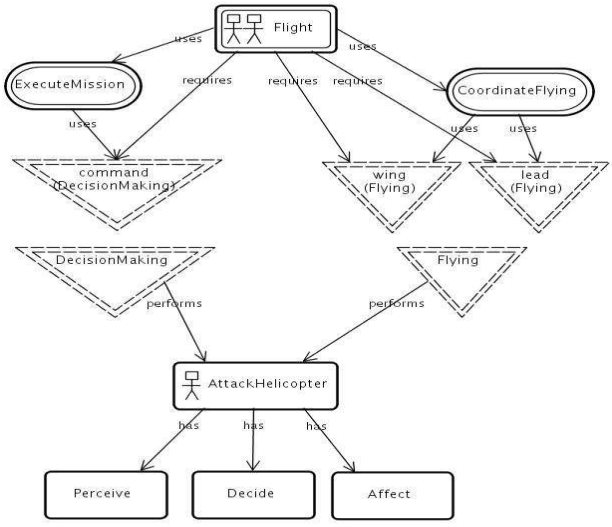


Figure 4: Designing teams of agents using roles and capabilities in JACK™

This is a complex military task with a rich spectrum of possible courses of action (i.e. doctrinal and non-doctrinal tactics) that could realistically be executed by the CoJACK agents in the Theatre of Operations. The demonstration shows how and why tactics are selected – in a similar fashion to how this is performed in the scientific serial subtraction demonstration – based on the situation awareness held by: (i) an individual AH modelled as a CoJACK agent, (ii) the Flight, i.e. a pair of AHs, flying as a lead and wing modelled as a JACK™ team, or (iii) the Patrol in charge of three AH Flights. This teaming design is expressed graphically in JACK™ as in Figure 4. The behaviours presented in the demonstration are based on tactics suggested by the military SME, but have been extended and extrapolated to best illustrate the goals of the HVCGF project, namely to show variability and changes in doctrinal to non-doctrinal decision-making at both the individual AH level and the C2 level in Flights or Patrols. Therefore, although the scenario has been designed to exhibit military credibility, it occasionally demonstrates exaggerated unlikely behaviour specifically to make the effects of human variability clear. Figure 5 illustrates an example screenshot of the Military Demonstration running.

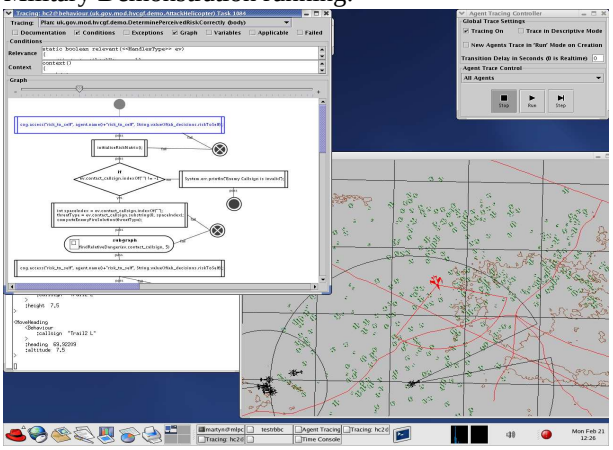


Figure 5: Screenshot of the Military Demonstration in execution.

The lower-right window is the display from OTB with a map of the battle area with the vehicles imposed on top; the upper-right window is a control panel for the simulation; and the upper-left window showing a trace of a CoJACK tactic – expressed in graphical form – for an AH. Two vignettes were chosen from the full mission scenario: a mid-mission re-tasking of the AH Patrol team to destroy a SCUD site; and an AH's chance encounter with a MiG-29 fixed wing aircraft as these vignettes illustrate: (i) use of SA, (ii) variation in tactic selection based on both SA and access to cognitive attributes, and (iii) fatigue as a critical moderator. For economy we focus here on the play-out of the SCUD vignette; the MiG-29 vignette is described in [15].

The start-up status for this SCUD vignette is that the Patrol's six Attack Helicopters are all capable of flying, but some may be damaged, have low fuel levels or might have depleted ammunition. A request is received from the Mission Commander to attack a newly identified target: a SCUD missile site. The

request specifies the locations where to form-up, observe and that of the target. It also specifies minimum resource requirements for the mission:

- The minimum number of fully operational Attack Helicopters required for the mission.
- The distance to target and estimated fuel load to complete the mission and return to the FARP.
- The minimum number of air-to-ground rockets needed per functional AH to destroy the target.

The Patrol team (reflecting the reasoning of a human patrol commander flying onboard one of the AHs) requests status information from each AH in the Patrol. Variability is not applied to the provision of this information, though in a coalition environment misinterpretation of data received from cohorts is a very real problem as for instance the fuel load might be expressed in Kgs by one country and in Lbs by another. The Patrol team then determines whether to undertake the mission based on the damage state, fuel load and missile complement for each member of the Patrol. If at any stage in this process it is determined that the minimum number of fully functional AHs is not available, then the mission request is rejected and the patrol returns to the FARP immediately.

Variability is applied in this stage of the process – the Patrol team's (mirroring that of the human patrol commander's) memory accesses are moderated to the cognitive attributes representing the status information received. Depending on the impact of moderation upon the Patrol's cognitive parameters, he may do one of three things. First, he might forget to include information relating to one or more Attack Helicopters in his calculation. Second, he might access the data incorrectly, e.g. AH4 radios in that his fuel load is 540Kg and three minutes later he performs his computation using the value of 450Kg. Third, he might take some long period of time to complete the computation, leading to the AHs being too far from the target to safely be re-routed.

Either of these circumstances may impact his overall assessment of whether to undertake the mission. The information on the status of each AH in the Patrol is structured as a set of cognitive attributes which he accesses whenever he requires this knowledge; the more frequently he accesses the cognitive attribute for a particular Attack Helicopter, the more credence is associated with the attribute and so he is better able to recall it. Conversely infrequent accesses to data about a AH status mean he is likely to forget it or make mistakes in remembering this knowledge. If the decision is made to proceed with the mission, no further variability is introduced into the deliberative behaviour – the effect of variability on reactive behaviour is the focus of the MiG-29 vignette. However the Patrol team may need to be dynamically restructured if there are only 5 fully functional Attack Helicopters. In this case the Patrol is restructured to consist of two Flight teams, one with 3 AHs and the other with two AHs. One Flight team will be designated with the role of 'attack group' and the other the 'observation group'. Any partially functional Attack Helicopters (namely those with limited fuel or rockets) are assigned to the observation group.

Hence AHs from different countries within the coalition might get re-formed into groups dynamically as the mission progresses, and so coordination among the AHs is a significant challenge as different forces use diverse Standard Operating Procedures (for close flying, manoeuvring, route following) and Concepts of Operations (for radio communication and firing on targets). If there are six fully functional AHs, then the original structure is retained: one Flight team will be the observation group and the remaining two Flights will be primary and secondary attack groups.

Variability is introduced each run by modifying the fatigue model applied to the Patrol team. This cognitive parameter affects the Patrol's ability to recall data: values between 0.0 to 1.0 results in respectively high, moderate and low levels of forgetfulness.

Runs for 1.0 and 0.5 were performed (i.e. relatively better and poorer recall capability). In both cases five helicopters were fully functional, with the wing AH for Flight team 1 having sustained damage. In the first run, the status information for all helicopters was remembered correctly and the SCUD site is successfully attacked.

In the second the commander "forgot" to include the fuel load for the lead helicopter for Flight team 2, which resulted in an incorrect assessment and the commander (incorrectly) assesses that his team of AHs have insufficient resources and declines to pursue the tasking, carrying on flying an egress route to the FARP. The vignette clearly demonstrates that fatigue moderation can have a significant impact on the behaviour of the patrol commander (reflected in the Patrol team). However the assessment process used by the commander was simplistic and needs further extension from both a cognitive and a tactics perspective. This was not pursued because of project time constraints. That said, the cognitive model can accommodate a variety of access models and JACK™ provides an ideal mechanism to represent tactics.

5. Conclusions

Inspired by the capability gap that currently exists for having cognitively-plausible and realistic control of CGF systems, we have discussed the background to this technology and the concepts of intelligent software agents, cognitive psychology and Computer Generated Forces systems to create a new coherent framework able to perform moderated decision-making by agents involved in scenarios like coalitions. In doing so, we illustrate the functionality of this system architecture through a Military Demonstration using teams of Attack Helicopter agents that have the quality of their Situational Awareness altered and so the selection and execution of tactics by CoJACK agents is moderated through the influence of fatigue.

Acknowledgement

The investigation which is the subject of this Report was initiated by the Director of Technology Development, Ministry of Defence, Metropole Building, Northumberland Ave. London WC2N 5BP and was carried out under the terms of contract RT/COM/3/006. Note that the views expressed here are those of the author and do not necessarily reflect those of MoD or other members of the project team.

References

[1] E. Norling, and F.E. Ritter, "A Parameter Set to Support Psychologically Plausible Variability in Agent-based Human Modelling", *Proc. of the 3rd International Joint Conference on Autonomous Agents and Multi Agent Systems*, New York, USA, published by ACM, 2004.

[2] F.E. Ritter, F. E. and E. Norling, "Extending a BDI architecture to make a better and more interesting team member: The case of JACK to CoJACK", in R. Sun (ed.) *Cognition and multi-agent interaction: From cognitive modeling to social simulation*, published by Cambridge University Press, 2005.

[3] C.A. Lucas, "Modelling Human Variability in Computer Generated Forces". *Proc. of International Conference on Behaviour Representation in Modelling and Simulation*, Scottsdale, Arizona, USA, 2003.

[4] M. Fletcher et al., "Enigma Variations – Simulating Changes in Behaviour of British Military Personnel", *Proc. of International Conference of Simulation Technology*. Canberra, Australia, 2004.

[5] M. Fletcher, C.A. Lucas, and S.P. Russell, "Moderating Behaviour in Teams of Attack Helicopters within a Synthetic Environment", *Proc. of the DTI/MoD Simulation and Synthetic Environments Symposium*, London, UK, 2004.

[6] M. Pechoucek and A. Tate (eds.), "*Proceedings of the 3rd International Conference on Knowledge Systems for Coalition Operations*", Prague, Czech Republic, 2004.

[7] A. Tate. (ed.), "*Proceedings of the 2nd International Conference on Knowledge Systems for Coalition Operations*", Toulouse, France, 2002.

[8] Multinational Forces Standing Operating Procedures Collaboration Site "Increasing Multinational Cooperation/Coordination for Military Operations Other Than War", http://www2.apan-info.net/mnfsop

[9] A.I. Karrasch, "Lessons Learnt on Collective Efficacy in Multinational Teams", US Army Research Institute for Behavioural and Social Sciences, http://www.au.af.mil/au/awc/awcgate/army/tr-1137.pdf,2003.

[10] Lessons for Future Coalition Operations, RAND Research Brief, http://www.rand.org/publications/RB/RB72/, 2001.

[11] J.E. Peters, S. Johnson, N. Bensahel, T. Liston, and T. Williams, "European Contributions to Operation Allied Force: Implications for Transatlantic Cooperation", ISBN 0-8330-3038-8, 2001.

[12] F.J. Lee, and S.J. Gamard, "Hide and Seek: Using Computational Cognitive Models to Develop and Test Autonomous Cognitive Agents for Complex and Dynamic Tasks", *Proc. of the 25th Annual Meeting of the Cognitive Science Society*. USA, 2003.

[13] F.E. Ritter, M. Avraamides, and I.G. Councill, (2002). "An approach for Accurately Modeling the effects of Behavior Moderators", *Proc. of the 11th Computer Generated Forces Conference*, Orlando, USA, 2002.

[14] J.L. Weeks, L.B. McDonald and J. Hughes, "Development of Computer-Generated Forces for Air Force Security Forces Distributed Mission Training", http://www.stormingmedia.us/27/2768/A276804.html, 2002.

[15] M. Fletcher, "The HVCGF MiG-29 vignette", TR of Agent Oriented Software, 2006.

[16] J. Tomaka, J. Blascovich, R.M. Kelsey, and C.L. Leitten, C. L. "Subjective, Physiological and behavioral effects of threat and challenge appraisal", *Journal of Personality and Social Psychology*, 56(2), 1993.

Coalition Formation with Unreliable Agents

Viktor Mashkov, Jaroslav Pokorny
Department of Software Engineering Faculty of Mathematics and Physics, Charles University
Malostrannske nam. 25, 11800 Prague 1,Czech Republic
jaroslav.pokorny@mff.cuni.cz

Abstract
The coalition formation problem has received a considerable amount of attention in recent years. This paper deals with the problem of coalition formation with the agents of specific cooperative multi-agent system, namely restricted alliance (RA). The presented approach to agent coalition formation takes into account the fact that agents may fail during task execution. We imply under coalition formation the following: (i) investigation of all possible coalitions with the agents of RA; (ii) determining the "best" one for the particular task execution; and (iii) formation of the chosen best coalition itself via agents communications. In this paper we concentrate on reducing the number of coalitions that have to be investigated and on determining the coalition which is the best in both fault-free and faulty situations.

1. Introduction

Nowadays, multi-agent systems are a subject of research in different fields, like computer science, mathematics, social science, economics and some others. Usually, under agent is understood a computer system capable of flexible autonomous action in a dynamic unpredictable and open environment. However, currently many researchers use the term agent to present different entities ranging from low-level implementation (e.g., transport unit [1]) to high-level (e.g., non-governmental organizations, army troops, etc. [2], [3], [4]) that may need to cooperate and coordinate their activities in pursuing certain goal(s). In this paper we consider an agent in a broader sense (as autonomous agent in MAS and as entity in complex social systems). Agents rarely act in isolation; in contrast they are increasingly required to act as elements of a large and complex system, and cooperate and coordinate with a number of other agents. Agents, referred to as entities, can cooperate and join together (i.e., form a group) in order to execute in a more efficient way the faced tasks (mission), or in order to gain benefits in case of self-interested agents. One of such groups was considered in [2], and was called alliance. Generally, the agent population of alliance can change (i.e., new agents can join the alliance, and some agents can abandon the alliance). We assume that agent population of alliance doesn't change during certain time (during coalition formation and mission execution), and all of the agents know about all the goals (missions) and the structure of alliance. The issues of alliance formation have been considered in [5], [6], [7]. Alliance is regarded as a long-term cooperation agreement among the agents. In many cases (e.g., peace-keeping operations, disaster-relief operations or rescue operations) a group of agents can be referred to cooperative MAS in which the agents are more interested in the overall outcome of the system rather than in their own benefits. Since agents may also be "self-interested" in a certain way (i.e., have their own intentions, goals and reasoning), they may agree to share resources and information only within some well specified community (which is called bellow as coalition). Agents may also utterly refuse to cooperate with some other agents. This fact results in restricted alliance (RA) formation [8], [9]. Being faced with particular mission, agents of RA should decide upon its execution. In the situation where a mission cannot be performed by a single agent or when an agent performs a mission inefficiently, the agents form coalitions. A coalition, unlike an alliance, is usually regarded as a short-term agreement among collaborative agents. The agents of RA may be regarded as non-supper-additive environment [10], [11]. Therefore the grand coalition is not always beneficial, and determining the most preferable one for executing the mission is a problem. Coalition is formed with the agents of RA each time a

call for help is received from an in-need entity. During coalition formation, agent shares its semi-private information only with those agents which it agrees to collaborate and perform the mission. Generally, there can be formed numerous potential coalitions with the agents of RA (this is NP-hard problem). Various potential coalitions may have different degrees of efficiency in mission execution due to differing capabilities of their members. Some of them have the capabilities needed to perform the mission. The task of choosing the single coalition can be solved either by central authority (e.g., independent arbiter) or by the agents themselves. In order to make decision about the most preferable coalition for performing a mission, it is necessary to assess and compare all coalitions which satisfy the mission's requirements. Assessment of coalitions (with group rational agents) is rather complex. There are few researches devoted to this problem (e.g., [12], moreover they don't take into account the possibility of agent failure while performing a task. Each coalition agent tries to adhere to its commitments and perform the assigned operation(s) in the best way, since it is group-rational. Nevertheless, there cannot be excluded situations when one or more agents fail in task execution. Unlike open environment where searching of a new agent to substitute the failed one may be not too costly, in case of RA finding a new agent may be hard to realize or even unrealizable owing to lack of agents. Moreover, for some missions (e.g., rescue operation) time is a crucial factor, and, consequently, there is limited time to recover the coalition. That is why, there should be envisage the appropriate methods and algorithms allowing coalition to tolerate different faulty situations which may occur while performing the mission. The main idea behind our approach to this problem consists in that we choose such coalition for mission performing which will require minimal costs for recovering when one of its agents fails. It is worth noting that in fault-free situation this coalition may not be the most preferable when applying the existing methods of coalition assessment. The trade-off between coalition that requires minimal costs for recovering and coalition which is the most preferable in fault-free situation can be performed on the basis of information about agents' reliabilities and about mission goals(s).

2. Environment description

It is assumed that RA consisting of N agents is faced with the mission which can be presented as one integrated task T. When a coalition is assigned a mission, task T can be further partitioned by the members of coalition into subtasks. We don't discuss either how the members of coalition fulfil that partition or the problem of mission planning. We assume that the agents are mainly interested in performing the mission in the most efficient way and don't pay attention to their own benefits since they are group-rational. Following to [12] we consider that each agent A_i has a vector of capabilities $P_i = < P_1^i,\ldots, P_r^i >$, and for satisfaction of mission M, vector of capabilities $\Omega = < \omega_1,\ldots, \omega_r >$ is necessary. Capability of an agent can be either splitable or non-splitable. When k agents form a coalition C, their capabilities are summed up, and the resulting coalition has a vector of capabilities

$$P_C = < \sum_{i=1}^{k} P_1^i,\ldots, \sum_{i=1}^{k} P_r^i > \text{ or } < P_1^C,\ldots, P_r^C >.$$

Coalition can perform the mission only if the vector of capabilities necessary for its fulfillment Ω satisfies:

$$\forall i,\ 1 \le i \le r : \omega_i \le P_i^C.$$

An excess of agents in the coalition for the case of non-splitable capabilities may lead to overheads in coalition capabilities. It is desirable to have the exact volume of capabilities needed for satisfying the mission. In non-super-additive environment an excess of agents in the coalition also leads to overheads due to communication, coordination and internal organization costs on the formation, operation and maintenance of the coalition. These are the main reasons why the number of agents in coalition should be restricted to optimal value. We assume that agents can communicate, negotiate and make agreements. These enable the formation of coalitions. In case of RA, where each agent has prior knowledge of the structure of alliance, each agent knows to which agents it can send the information about its capabilities. As the key element of RA's structure, we suggest to use the non-expandable group of agents agreeing to communicate and ultimately form coalition. Under the term "non-expandable" we mean that the mentioned group of agents cannot be expanded by adding one more agent to it. Hereafter we call such group of agents of RA as a potential local great coalition (PLGC), since the agents of this group may potentially

form a coalition which will include all the agents of the group. Generally, any agent may be a member of several such $PLGC_s$. The agent which is a member of more than one PLGC, is termed as junction agent (JA). The RA can be presented in the form of graph. The vertices of the graph represent agents, and the edges correspond to contacts among the agents. Any two vertices of the graph have mutual edge if the corresponding agents agree to communicate with each other. In Figure 1 the simple RA consisting of three $PLGC_s$ is shown.

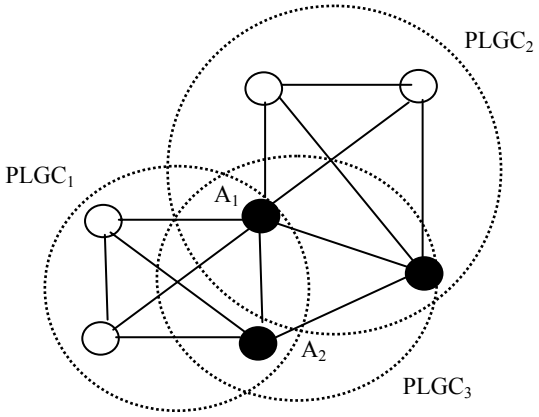


Figure 1. Example of restricted alliance

Junction agents are shown as vertices filled with black. Agents of each PLGC agree on the distribution of coalition capabilities calculations among themselves. Calculation-distribution among agents of RA was considered in [9].

3. Candidate coalitions

Under candidate coalition we understand the coalition that has sufficient capabilities to perform a mission. Generally, with the agents of PLGC of size m there can be formed 2^m-m-1 potential coalitions. In order to check if the potential coalition can be the candidate coalition, it is necessary to compute its capabilities and then compare them with Ω. The number of potential coalitions is exponential. That is why an exhaustive search for the potential coalitions is infeasible for large m. A reduction in this number is possible by limitations on the permitted coalitions. This can be done via the constraints of the specific problem under investigation. As it follows from the environment description, coalitions with a small number of agents are more likely to be regarded as preferable. In view of this, we suggest organizing searching procedure for candidate coalitions so that the potential coalitions with smaller number of agents are examined at the beginning. Searching for the candidate coalitions is assigned to agents of each PLGC. To enable this searching procedure, the agents should be provided with the appropriate algorithm. Agent executing the algorithm should maintain a list of candidate coalitions L_C. The algorithm takes the data $\{P_i\}$, i=1,…,m and Ω as an input and returns the completed L_C.

The algorithm

Step 1. Set i=2.

Step 2. Compute all potential coalitions {C} of size i and add them to list of potential coalitions L_p

- if there is at least one coalition C^* on L_C such that $C \subset C^*$, then don't add C to L_p;
- otherwise, add coalition C to L_p;

Step 3. Check if L_p is empty,

- if yes, then end;
- otherwise, proceed with next Step.

Step 4. Choose the first in order coalition on L_p and compare its capabilities P_C with Ω

- if L_p is empty, then proceed with next Step;
- if $P_C \geq \Omega$, then remove C from L_p and add it to L_C; repeat Step 4;
- if $P_C < \Omega$, then remove C from L_p; repeat Step4.

Step 5. Set i=i+1; and proceed with Step 2.

This algorithm is efficient when the probability of candidate coalition of two agents is quite high. This was confirmed by results of simulation (Figure 2). We intentionally set the values P_i, i=1,…,m and Ω so that the number of candidate coalitions consisting of two agents, f, takes the definite values. The average number, N_0, of potential coalitions which had been examined at Step 4 of algorithm for m=5, 7, 10 is presented in Figure 2.

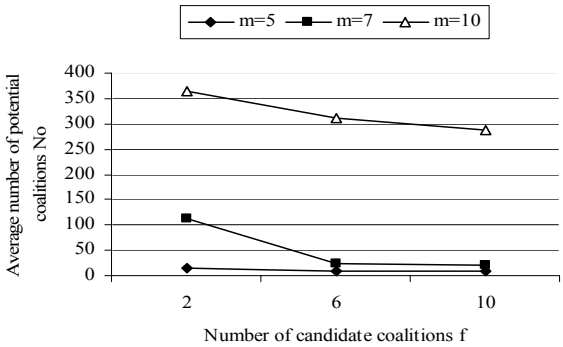


Figure 2. Results of simulation

We can adjust the algorithm so that if there had not been found candidate coalitions consisting of two or three agents, then end the algorithm after any candidate is found. In the worst case, we can adjust the algorithm so that it skips several steps and proceeds with i=i+k-2, where k is integer which denotes the highest coalitional size allowed. In which case, the number of investigated coalition $O(m^k)$ is a polynomial number in m. This approach was exploited, for example, in [12].

Having found all coalitions that would be able to fulfil the mission (i.e., candidate coalitions), we can assess and compare them in order to choose the single one which will be granted the mission execution.

4. Coalition assessment

In order to enable the assessment of coalitions and of mission fulfillment, an evaluation function shall be attached to each type of capability. Such a function shall transform the capability units into monetary units [12]. For some domains (e.g., transportation company case [1]) this function may be the income from performing a task, and the utility gained from performing the task depends on the capabilities that are required for its execution. However, in many cases (e.g., peacekeeping and humanitarian operations, rescue operations, etc.) it is very difficult, or even impossible, to assess the utility gained from mission execution. For these cases we suggest assessing the coalitions via the total expenditure E_M, on the execution of coalition task. In which case, the most preferable coalition is the one for which the value E_M is minimal. In reality, there is always probability that agents fail during mission execution. If the event of agent failure occurs, the subsequent recovery from faulty situation may be difficult and expensive. In view of this, it is reasonable to include the cost of coalition recovering in total expenditure on mission execution. Since the cost of coalition recovery depends to a great extent on agents' reliabilities (i.e., random value), the total expenditure E_M is also random value. Our proposal is to consider the mean value of total expenditure, ${}_mE_M$,

$$ {}_mE_M = \sum_{i=0}^{q} E_M^i p^i , $$

where E_M^i – total expenditure on mission execution when i agents fail; p^i – probability that exactly i agents fail during mission execution.

In this paper we don't tackle the following problems: what may constitute an agent failure; how agent's failure is detected; which failure detection techniques and facilities should be used; how to assess the quality (degree of perfection) of failure detection facilities; how agents become aware of other agent's failures; and some other problems pertaining to traditional dependability [13]. Following to conceptual framework provided by dependability, we state that for the whole coalition an agent failure can be considered as error. If the appropriate error handling means are not envisaged, the error may lead to coalition failure. Increasingly, all comprehensive complex systems include error handling techniques to tolerate the possible errors. The core dependability concepts distinguish three forms of error handling: rollback recovery, rollforward recovery and compensation. In context of coalition error, examples of each form include: (i) rollback recovery: failed agent is substituted by available one; operations performed by the agents are discarded; coalition returns to initial state and mission execution starts from the beginning; (ii) rollforward recovery: failed agent is substituted by available one; coalition is searching for a new state from which it can operate and continue the mission execution; (iii) compensation: coalition exploits redundancy in the agents; failed agent is removed from the coalition; remaining agents proceed with mission execution; some corrections are possibly needed in execution plan. In this paper, the focus is on rollforward coalition recovery. Considering possible faulty situations in their entirety is complex task. For that reason, at the beginning we consider the simplified task, and assume that the probability of two or more agents failing is negligible. With the account of this assumption, the mean value ${}_mE_M$ can be defined as ${}_mE_M = E_M^0 p^0 + E_M^1 p^1$, where

$$ E_M^0 = W_C + I_C , \quad (1) $$

$$ E_M^1 = {}_mW_C^1 + {}_mI_C^1 + {}_mR_C^1 . \quad (2) $$

where W_C - the cost of coalition capabilities which agents of coalition use for mission execution in fault-free situation; I_C – internal coordination cost of coalition in fault-free situation; ${}_mW_C^1$ – mean value of cost of capabilities which agents of coalition use for mission execution when one of the agent fails; ${}_mI_C^1$ - mean value of internal coordination cost of coalition when one of the agents fails; ${}_mR_C^1$ – mean value of the cost of coalition recovery after agent's failure.

We consider mean values ${}_mW_C^1$ and ${}_mI_C^1$, because these costs depend on which particular agent fails, i.e.

$$ {}_mW_C^1 = \frac{1}{q}\sum_{i=1}^{q} W_C^{A_i} \quad (3) $$

where W_C^{Ai} , i=1,…,q, is the cost of capabilities which agents of coalition use for mission execution when agent A_i fails. The same refers to the internal coordination cost, i.e.

$$ {}_mI_C^1 = \frac{1}{q}\sum_{i=1}^{q} I_C^{A_i} \quad (4) $$

The cost of coalition recovery ${}_mR_C^1$ can be determined as

$$ {}_mR_C^1 = P_r Z + (1 - P_r)Y, \quad (5) $$

where P_r – the probability that there is an agent in the alliance capable of substituting the failed agent; Z – the cost of substituting the failed agent; Y- the cost of recovery from faulty situation by way of substituting the whole coalition.

We suppose that value Y is much greater than Z. We also assume that these values have little dependence on

which particular agent fails. Consequently, probability P_r can be given by $P_r = r/q$, where q is the size of candidate coalition; r is the total number of agents in the candidate coalition which can be substituted. Given W_C and I_C , value E_M^0 can be easily computed according to (1). Computed expenditures E_M^0 of all candidate coalitions are put on the list L_E , and are arranged in the order from the lowest to the highest. Whereas, for the value E_M^1, there are needed appropriate algorithms allowing to compute the mean values of corresponding costs.

5. The algorithms

The value E_M^1 is computed for the situation when there is a failed agent in the coalition and it is not known precisely which particular agent has failed. In this case it is necessary to check the possibility of substituting for each of its agents.

Agent A_i may be substituted by agent A_j only if the following two conditions are satisfied:

1) agent A_j belongs to the same PLGC as the agents remaining in the coalition C after removing agent A_i (i.e., $C \setminus \{A_i\}$);

2) coalition (of size q) after substituting agent A_i has the capability P^C such that

$$\forall i,\ 1 \le i \le r : \omega_i \le P_i^C , \qquad (6)$$

where r is the number of different types of resources required for mission execution. Generally, agents of coalition may be members of several $PLGC_s$. It means that agent A_j (substituting agent) has to be searched in all these $PLGC_s$. For example (see Fig. 1), if candidate coalition consists of two agents A_1 and A_2 , then agent which would be able to substitute agent A_1 has to be searched in $PLGC_1$ and $PLGC_3$. Below we present the algorithm 1 for calculating the number r. This algorithm should be executed by all junction agents of the alliance. The number r is determined for each candidate coalition C.

Algorithm 1:

At the preliminary step for the coalition C of size q the list of its agents, L_A, is formed.

Step 1. Set r=0.

Step 2. Form the list of $PLGC_s$, L_g, where each PLGC on the list L_g includes at least (q-1) agents of coalition C (i.e., $\forall PLGC \in L_g :\ \exists Sc \subset C, |S_C|=q-1 :\ S_C \subset PLGC$).

- if list L_g is empty, then return r (end);
- otherwise, proceed with next step.

Step 3. Choose the first in order agent on list L_A and check all $PLGC_s$ on list L_g for substitution of chosen agent

- if list L_A is empty, then return r (end);
- if substitution is possible, then remove agent from list L_A ; set r = r + 1; and repeat step 3;
- if substitution is impossible, then remove the agent from list L_A ; and repeat step 3.

Checking at step 3 if the substitution of agent is feasible is based on (6). After determining r, it becomes possible to compute the cost ${}_mR_C^1$ according to (5). As concerns the costs ${}_mW_C^1$ and ${}_mI_C^1$, we can make some notes. When capabilities of agents can be split, there is no need to compute ${}_mW_C^1$. This cost will be the same for all candidate coalitions and equal to cost W needed to satisfy the mission. If internal coordination cost doesn't change after agent substitution, this cost should be computed for each candidate coalition only once for fault-free situation (i.e. I_C in (1)). In general case when agents' capabilities cannot be split and internal coordination cost changes after agent substitution, the values ${}_mW_C^1$ and ${}_mI_C^1$ can be computed for each candidate coalition by using the algorithm presented below. To execute this algorithm, each junction agent in addition to list L_A should also maintain the list L_E , the list of costs ${}_mW_C^1$, L_W, and the list of costs ${}_mI_C^1$, L_I. At the preliminary step the first in order expenditure E_1 is chosen from the list L_E.

Algorithm 2:

Step 1. Form the list of $PLGC_s$, L_g, where each PLGC on the list L_g includes at least (q-1) agents of coalition C

- if the list L_g is empty, then return E_1; end;
- otherwise, proceed with next step.

Step 2. Choose the first in order agent on list L_A and check all $PLGC_s$ on list L_g for substitution of chosen agent

- if list L_A is empty, then proceed with next step;
- if substitution is possible, then compute W_C^A and I_C^A; add them on the lists L_W and L_I respectively; remove agent from list L_A , and repeat step 2;
- if substitution is impossible, then remove agent from list L_A ; and repeat step 2.

Step 3. Using the lists L_W and L_I , compute ${}_mW_C^1$ and ${}_mI_C^1$ according to (3) and (4) respectively; end.

It is advantageous to provide separate computing of cost ${}_mR_C^1$ (Algorithm 1) and costs ${}_mW_C^1$ and ${}_mI_C^1$ (Algorithm 2), since in such case the above costs can be computed in parallel by different junction agents. The algorithms have low computational complexity, O(N). The corresponding values being computed, the expenditure E_M^1 can be calculated according to (2) for all candidate coalitions. The obtained values $\{E_M^1\}$ are put on the list L_E^1. The list L_E^1, along with the list L_E formed earlier for fault-free situation, present essential data for choosing among candidate coalitions the most preferable one for mission execution. This can be done either in centralized or in distributed manner. The latter was addressed in [9]. We assume that the agents can come to the final decision which coalition is the most preferable also taking into account the respective information about agents' reliability and about the requirements and the goal(s) of the mission. Nevertheless, information about coalitions'

expenditures on mission execution per se can help to predict the consequences of abnormal behavior of agents and envisage the appropriate countermeasures to tolerate failed agents. The algorithms presented in this paper have been designed to enable such prediction.

6. Conclusion

Admittedly, the issues of failure handling and recovery pertaining to MAS are challenging [14]. In this paper we have considered recovery of MAS based on substituting the failed agents by available ones. We imposed two constraints on agent substitution. The first constraint is defined by the structure of MAS (restricted alliance), and the second one – by the agents' capabilities. The problem of recovery is closely linked to the problem of coalition formation. Commonly, these problems are considered separately. The novelty of the research presented in this paper is that the task of choosing the coalition for mission execution and the task of recovery the coalition when one of its agents fails during mission execution, have been considered interdependently, which means that the solution of the first task requires consideration of the second task. This is implemented by including the parameters of recovery in coalition assessment. In this paper we have considered the above tasks for the case when some assumptions are accepted in relation to agents (their resources, reasoning and abilities), tasks (missions) and coalitions. Particularly, we assumed that the tasks are fulfilled by the group-rational agents, the environment of agents is non-super-additive and is limited to the number of agents in the restricted alliance. We also assumed that a coalition can work on a single task at a time, and that the tasks are independent (i.e., without precedence order). We view the presented researches as the initial step towards integrating dependability aspects with coalition formation aspects. In future, we aim to relax some accepted assumptions and to research more general cases.

References:

[1] T.W. Sandholm. An implementation of the contract net protocol based on marginal cost calculation. *Proc.of AAAI-93*, Washington, 1993, 256-262.

[2] M. Pechoucek, V. Maric, J.Barta. A knowledge-based approach to coalition formation. *IEEE Intelligent Systems, 7(3),* 2002, 17-25.

[3] V. Marik, M. Pechoucek. O. Stepankova. Social knowledge in Multi-agent systems. *Lecture Notes in Artificial Intelligence, Vol.2086,* 2001, 211-245.

[4] M. Fasli. From social agents to Multi-agents systems: Preliminary Report. V. Marik et.al. (Eds). *CEEMAS 2003, LNAI 2691*, 2003, 111-121.

[5] V. Mashkov, V. Marik. Alliance formation praocess and communication traffic. *Proc. of IASTED AIA'2002 conference*, Malaga, 2002, 417-422.

[6] V. Mashkov, V. Marik. Diagnosing faulty situations during alliance formation process. *Proc. of IASTED AIA'2003 conference*, Innsbruck, 2003, 72-78.

[7] V. Marik, V. Mashkov. Alliance formation with several coordinators. *Soft Computing Systems.* A. Abraham et.al. (Eds). IOSPress, Amsterdam, 2002, 550-561.

[8] V. Mashkov. Restricted alliance and coalition formation. *Proc. of IEEE/WIC/ACM Internation Conf. on Intelligent Agent Technology*, Beijing, 2004, 329-332.

[9] V. Mashkov. Task allocation among agents of restricted alliance. *Proc. of IASTED ISC'2005 conference*, Cambridge, MA, USA, 2005, 13-18.

[10] R. Conte, M. Miceli, C. Castelfranchi. Limits and levels of cooperation: Disentangling various types of prosocial interaction. *Decentralized Artificial Intelligence. Vol.2,* 1991, 147-157.

[11] J.C. Harsanyi. A simplified bargaining model for N-person cooperative game. *International Economic Review, Vol.4,* 1963, 194-220.

[12] O. Shehory, S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence, 15(3),* 1998, 218-251.

[13] J.C. Laprie (Ed). *Dependability: Basic Concepts and Terminology, Dependable Computing and Fault-Tolerance,* 5, Springer-Verlag, Vienna, Austria, 1992, 265p.

[14] M. Luck, P. McBurney, C. Preist. *Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent-Based Computing.* AgentLink II, 2003.

Modelling a Typical Guerrilla War

Jim Doran

Department of Computer Science
University of Essex
Colchester, UK

doraj@essex.ac.uk

Abstract. An agent-based model of a typical guerrilla war, the Iruba model, has been designed and implemented based upon published descriptions and theories of guerrilla warfare. Experimental results have been obtained with the model and conclusions drawn. A core feedback loop is detected. The possibility of using the Iruba model to predict the outcomes of specific guerrilla wars is discussed and it is suggested that such predictive models are feasible and are potentially useful tools for peacemakers.

Keywords: agent-based modelling, guerrilla wars, asymmetric warfare, Iruba model, predictive model, peacemaking

INTRODUCTION

Both guerrilla warfare itself and the study of it are at least as old as recorded history (e.g. the writings of Sun Tzu in the fourth century BC). Although the concept of guerrilla warfare is perhaps a little unclear, all agree that it involves asymmetric forces with the weaker force (in conventional terms) deploying mobility and surprise "hit and run" tactics, and using difficult terrain or a sympathetic general population as a safe refuge as required. Typically there is an "insurgency" and "regime forces". Complementary political action is entwined. "Terror" may be used at an extreme by either side (e.g. Sederberg, 1989) either as a consequence of a failure of discipline or as a deliberate means to victory.

Relatively modern studies of guerrilla warfare, such as those of Mao Tse-Tung and Che Guevara (1962), Tabor (1970), Gann (1971), Arquilla and Ronfeldt (2001), Beckett (2001), offer general insights coupled with practical guidance, from the perspective of both insurgents and counter-insurgents.

There are more than a dozen ongoing guerrilla conflicts worldwide (notably in Chechnya, Columbia, Iraq, Nepal, Spain, Sri Lanka) in various stages of development. Thus there is a pressing need for maximum scientific understanding to be achieved.

In agent-based social modelling on a computer (Doran and Gilbert, 1994; Doran, 1997; Gilbert and Troitzsch, 1999) the model embodies aspects of individual or collective decision-making. Thus this type of modelling offers a means to achieve new understandings and to enhance those already documented in the technical literature of guerrilla warfare.

Previous and relevant agent-based modelling studies are those of Epstein (2002) and Raczynski (2004). Epstein reports an interesting series of experiments with a model that captures a form of "decentralised rebellion" reflecting initial population grievance levels and degree of perceived regime legitimacy. The model is grounded at the level of the individual and targets "recognisable macroscopic revolutionary dynamics" and effective methods of suppression. Raczynski's study puts the emphasis on the dynamics of terrorist and counter-terrorist organisational structures and on the process of destroying terrorist organization links by the anti-terrorist agents. Again, agents correspond to individuals. Neither study seeks objectively to model a guerrilla war as a whole.

There is, of course, much ongoing work deploying simulations and "intelligent" agents in mainstream defence contexts (see, for example, Mittu, 2004) but such work is typically concerned to enhance existing military capabilities or to support planned military operations rather than to build scientific understanding.

THE IRUBA MODEL

Standard agent-based social modelling procedure envisages the following stages: initial study of target social "system", formulation of model, validation of model, use of model to gain insights into target system. Key issues are the choice of computational structures to represent agents, the nature of agent interactions, the agents' joint environment, and the specific techniques adopted to validate the model, that is, to ensure its reliability as a source of insight about the target.

The Iruba project (Doran, 2005) is following this approach to construct and experiment with a general model of a guerrilla war sufficiently realistic to offer new insights into dynamics or to further develop existing insights. In the model agents correspond to guerrilla bands, regime bases or outposts, and to headquarters on each side. A particular objective is to establish sets of conditions expressed in terms of the model's parameters and structures that guarantee that an insurgency will succeed or, alternatively, will fail.

The Iruba model has been made broadly realistic having regard to the relevant literature. In particular, the model is loosely based on (extensive descriptions of) guerrilla wars that took place in the last century in Ireland (1919-1921) and in Cuba (1956-1959), with some further features drawn from the Arab Revolt against the Turks in the Hejaz (1917-1918) towards the end of the First World War.[1] Reliable sources for these conflicts are Beckett (2001), and Hart (2003) but there are many others.[2] The most important structural and behavioural concepts used in building the Iruba model are drawn from the Irish insurgency: near autonomous regions with only limited central control; mobility; limited weaponry; the importance of terrain; and the importance of ideology and popular support.

Correspondingly, the Iruba model is structured as a network of 32 relatively autonomous regions that vary in terrain and population. The population of a region provides a (finite) recruitment pool for both insurgents and regime forces. Initially the regime forces are relatively numerous, distributed in bases over the island, and relatively static, whilst the insurgents are small in number, localised, mobile and hard to find. As indicated, computational agents represent guerrilla cells/bands and regime bases and insurgent and regime headquarters. Attacks take place within regions following simple rational strategies. For example, a guerrilla band may attack a poorly defended regime base, with the outcome dependent upon terrain, relative numbers and weaponry, and random factors. A successful attack may well lead to capture of weapons. Movement of insurgent or regime forces between neighbouring regions takes place under appropriate conditions. For example, neither the forces that are moved nor those that remain behind are left at significant risk. Recruitment to insurgents and to regime forces (and defection from regime forces) reflects the numbers and attitudes of the so far uncommitted general population of the region in question. This population will partially support the insurgents, and will partially be aware of the insurgency, depending upon the conflict history in that region. These two "population attitude" variables and their use are intended to go some way towards capturing the dynamics of population opinion and its impact upon the course of the insurgency.

The core cycle of the model may be expressed in outline pseudo-code as:

Repeat
    Attacks and their impact
    HQ decisions
    Recruitment
    Force movement
Until termination

As indicated above, a degree of central control by "headquarters" agents is possible for both sides. If an insurgency grows, regime force may be concentrated into regions where the insurgency is at its strongest. Furthermore, faced with a dangerous insurgency the regime may take "all out" measures (comparable with, for example, the so called "Salvador option").[3] On the other side, in appropriate circumstances the insurgents may be switched into "hyper-mobile" mode (comparable with the use of "flying columns" by the IRA in Ireland) and/or an "all out" attack across a range of regions or even the entire island, may be triggered

[1] In each of these examples, the insurgents proved (more or less) successful. However, the structure of the Iruba model also allows regime success as will become apparent.

[2] There is often much in the published descriptions of insurgencies that is inaccurate and biased to one side or the other. This is certainly true of aspects of the Irish insurgency as has been demonstrated by Hart (2003).

[3] For the "Salvador option" see Michael Hirsh and John Barry, NEWSWEEK, Jan 10th, 2005.

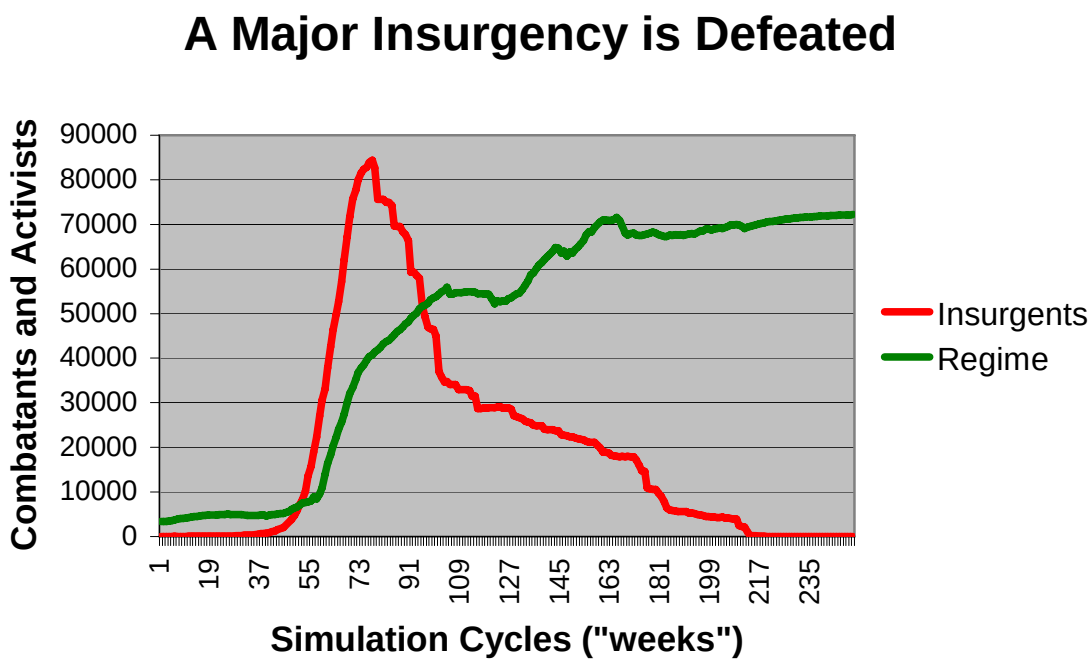


Figure 1. A simulated insurgency "takes off" but is then defeated by regime counter-action. See text for further commentary.

(compare the Tet Offensive in the Vietnam war).

Victory in this model is a matter either of insurgent annihilation, or of the insurgents achieving numerical superiority and hence, by assumption, political power. At several points the model invokes chance factors (using a pseudo-random number generator) so that the success or failure of an insurgency may vary with the pseudo-random number stream seed even if all other model setting are the same.

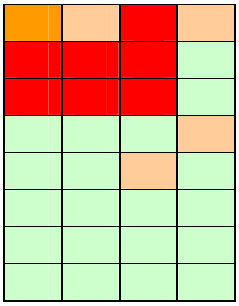


Figure 2. Shows the spatial distribution of the insurgency of Fig 1 on cycle 84 (just after its maximum). The insurgents (red) are still concentrated around their start point in the (mountainous) "north-west" of the Iruba "island". Most of the island remains under regime (green) control and the regime is beginning to regain control of the core region of the insurgency.

The Iruba model has been implemented in the C programming language.[4] Although some model variables (e.g. population support for insurgents) are updated by simple mathematical relationships, many aspects of the model structure are much more complex. For example, agents (guerrilla bands, regime bases and HQs) are essentially expressed as sets of conditional rules. Thus, even if feasible, formal mathematical or logical specification of the model independent of the code would achieve nothing.

EXPERIMENTAL RESULTS AND INTERPRETATIONS

A typical insurgency within the Iruba model is shown in Figure 1. This particular insurgency fails after making good initial progress. The regime goes into a sustained "all-out" mode when the insurgency reaches a certain size (in this case, 15000 personnel in total see Table 3). The immediate effect of this is spatially to contain the insurgency and it is ultimately decisive, in spite of further insurgent successes, when the recruitment pool within the insurgency area is exhausted. It is important to know that the number of weapons available to the insurgents is restricted by their ability to capture weapons from the regime forces. This means that for much of the time the insurgency in Figure 1 is much less powerful than its numbers suggest. Figure 2 is

[4] Guidance on downloading and running the Iruba code is available by email from the author.

a snapshot of the spatial development of this insurgency at its maximum showing how it has spread out from its region of origin in the mountainous "north-west" of the "island".

Experimental trials[5] with the Iruba model show, as expected, that victory for the insurgents or for the regime in the model depends crucially upon parameter settings. Initial experiments have focused on the impact of the initial size of the insurgent group, and of a limited form of central decision making by both insurgents and regime forces. Part of the motivation for these experiments was to test *foco* theory as propounded by Guevara and Debray (Beckett, 2001, p 170-1) following Castro's success. This holds that even a very small dedicated group of insurgents will succeed provided that they have a political as well as military strategy, and provided that there is a significant level of initial support in the population at large.

Initial guerrilla band size	**30**	**35**	**40**	**45**	**50**	**55**
Insurgent success (%)	**5**	**28**	**58**	**79**	**86**	**90**
Insurgent success (%) if regime force concentration	**3**	**23**	**45**	**77**	**83**	**80**

Table 1 Impact of initial guerrilla band size on insurgent success rate. Success is taken to mean that the total insurgent force has grown to more than 100,000. Results were compiled from 100 trials (ie 100 simulated guerrilla wars) for each band size, each with a timespan of 150 cycles (notionally weeks). For other parameters settings see text.

Iruba results (Table 1) suggest that, with this particular calibration of the model, an initial band size of about 40 is needed to give a 50% chance of insurgent success. The insurgent success rate is significantly reduced if there is an element of centralised force concentration on the regime side. In this (and the following) experiments the population in each region was initially set at 10,000 and was initially fully "passive" with only 10% insurgent support. Other parameters in the Iruba model were set at plausible values.

For comparison, at the outset of his Cuban insurgency Castro initially had 81 followers, who were almost immediately reduced to about 20 in an attack by regime forces. The results of Table 1 indicate the unreliability of *foco* theory as propounded by Guevara and Debray. In fact, most insurgencies inspired by *foco* theory do seem to have failed (Beckett, 2001, p. 171).

Table 2 shows what happens when the insurgents are made more effective in attack, and when their efficiency at recruitment (in real life partly a matter of communication) increases. Interestingly, the results suggest that within the Iruba model effective recruitment is more important than military skills.

	1.0	**1.5**	**2.0**
1.0	**58**	**68**	**68**
1.5	**73**	**86**	**90**
2.0	**94**	**95**	**97**

Table 2 Impact of insurgent attack effectiveness and insurgent recruitment efficiency. The former increases with column, the latter with row. Table entries are insurgent success rates (again calculated over 100 trials), with a success criterion of 100,000. Initial insurgent band size is 50.

Taken together these results suggest that sufficient preconditions for likely insurgent success in the Iruba model as calibrated are: a sufficiently large initial band, at least minimal mobility, attack efficiency, some initial population support, and communication processes by which insurgent successes impact the population at large and increase awareness and support for the insurgents.

In all these experiments a potential positive feedback loop is apparent: *increasing insurgent numbers make insurgent success more likely which increases population support for the insurgents and hence recruitment to and the numbers of the insurgents.* All the forgoing trials indicate that if this loop is reliably established, and if spatial spread is achieved, then the insurgents succeed. If not, then they partially or completely fail. However it is possible, within the model, for the loop to be disrupted even when it has been established. In Table 3 is shown the average impact of an "all out" regime counter attack on the insurgents when triggered by the insurgency reaching a threshold total size.

An "all-out" regime counter attack comprises a set of regime changes including better attack efficiency, more effective recruitment, more focussed force concentration, and more effective insurgent group detection techniques, all implemented by appropriate parameter adjustments within the model. Once these changes are triggered in a particular trial, they remain in place until the end of it.

[5] Using Iruba version 5.9. Some of these results were first reported in Doran (2005).

Regime counter-attack threshold	5000	10000	30,000	50,000
Insurgent success rate %	0	4	27	52

Table 3 Insurgent success rates when, in addition to regime force concentration, an "all out" regime counter attack is triggered at the stated insurgency size. Success criterion for insurgents is 100,000, and table entries are again based on 100 trials each here of length 300 cycles. Recall that a total insurgency size of more than 10,000 implies that the insurgency has certainly spread beyond its initial region.

Table 3 indicates that an "all out" response by the regime is highly effective, especially if deployed early. With no "all-out" counter-attack at all the insurgent success rate is 88.

LIMITATIONS OF THE MODEL

Although the Iruba model is already complex, it is apparent that a great deal of relevance is missing from it. The omissions include matters of relative detail, for example, different types of attack including explicit "terror" attacks and assassinations, the distinction between death, injury, and imprisonment, and intelligence gathering and also such major matters as population movement, external third party involvement, and the political and administrative structures that insurgents often create as part of their struggle.

PREDICTING OUTCOMES

It is widely held in the relevant literature that the outcomes of guerrilla wars can sometimes be predicted if the required information is available and taken into account.[6] The same is often asserted for social revolutions (e.g. Foran, 1997). Although Iruba is a general model designed to support exploration of the space of possible guerrilla wars and to discover core properties (see earlier discussion of the core feedback loop), it can be used predictively if we (a) "fit" the model to the specifics of a particular war of interest, and then (b) run it repeatedly from the current military/political situation to find likely outcome(s). Core general properties previously discovered with the aid of the general model may be used to guide and interpret specific prediction.

Of course, this simple prescription ignores major difficulties. There are at least three sources of uncertainty and unreliability. Firstly the general model itself may be inaccurate and incomplete. Some of the Iruba model's limitations in this regard have already been stated. In most real contexts at least some of these limitations would have to be removed.

Secondly, the process of "fitting" the model to the particular guerrilla war instance is likely to be very difficult to perform with precision. The number of parameters to be specified is very large, and many of them would be impossible to collect evidence for under conflict conditions. Even in historical retrospect this task is difficult (Hart, 2003). Perhaps the most than can be hoped for is that a probability distribution be estimated for each such parameter and that prediction is based upon the corresponding joint probability distribution over the parameter space. Just how this is best done remains an open question.

Finally, the model is inherently stochastic so that at best predictions will be in terms of probabilities (see the results presented earlier). At worst there may be regions in a model's parameter space that are "chaotic" in the sense that very small changes in parameter settings may lead to major changes in the probabilities of particular outcomes, and these will need to be carefully mapped.

THE IRISH WAR OF INDEPENDENCE

The Irish War of Independence (1919-1921) is potentially an instructive test case. As it is relatively well documented, it may be possible to use the abundant (but still seriously incomplete) historical evidence to capture within Iruba the state of the conflict as it was in, say, March 1920. Then we may ask what outcome Iruba predicts. Historically, there was a ceasefire and negotiations in late 1921 that ended in partial success for the insurgents and then a brief civil war. For obvious reasons, retrospective prediction of this type falls well short in difficulty of prediction of the outcome of an actually ongoing war.

PEACEMAKING

It seems possible that predictive use of a general model of a guerrilla war has a potential role to play in cease-fire negotiations. To the extent that the reliability of the model has been demonstrated, it has the potential convincingly to elaborate or even correct the assessments that parties to such a negotiation will necessarily already have made about their own

[6] Consider, for example: "Granted mobility, security (in the form of denying targets to the enemy), time and doctrine (the idea to convert every subject to friendliness) victory will rest with the insurgents, for the algebraical factors are in the end decisive, and against them perfections of means and spirit struggle quite in vain" (Lawrence, 1929, page 953).

military prospects. Thus the model might play a role akin to that of an adjudicator in a chess game, and help cut short the conflict by furthering a negotiated agreement reflecting the actual relative strengths of the combatants. Stakeholder participation in agent-based model building, and in collective discussion and utilisation of the results of a modelling study, are ongoing research themes (see, for example, Barreteau et al., 2003).

The cost of creating a general and reliable model of a guerrilla war and standardising its use for prediction is potentially very high, but this cost is surely easily recouped if reductions in death and destruction are achieved on even a single occasion.

CONCLUSIONS

The Iruba model is an agent-based model of a typical guerrilla war that has a degree of realism. Experimental results have been obtained that offer suggestive insights, notably concerning the unreliability of *foco* theory, and the impact of such counter-insurgency strategies as the "Salvador option".

Furthermore, the use of the Iruba model, or a development of it, reliably to predict the outcome of an ongoing guerrilla war seems both possible, if challenging, and potentially of real importance for peacemaking.

REFERENCES

Arquilla John and Ronfeldt David, Eds. (2001) *Networks and Netwars: the Future of Terror, Crime and Militancy*. Santa Monica, CA: Rand

Barreteau, O. et al. (2003) "Our Companion Modelling Approach". *Journal of Artificial Societies and Social Simulation*, 6(1) <http://jasss.soc.surrey.ac.uk/6/2/1.html>

Beckett Ian F. W. (2001) *Modern Insurgencies and Counter-Insurgencies: guerrillas and their opponents since 1750*. Routledge: London and New York.

Doran J E (1997), "From Computer Simulation to Artificial Societies" *Transactions SCS*, 14(2), 69-77, June 1997 [Special Issue: Multi-Agent Systems and Simulation]

Doran J E (2001) "Can Agent-Based Modelling REALLY be Useful?" In *Cooperative Agents: Applications in the Social Sciences* (Eds. Nicole J Saam and Bernd Schmidt) Kluwer Academic Pub, Pp 57-81

Doran J E (2005), "Iruba: An Agent-Based Model of the Guerrilla War Process", ESSA 2005 Workshop, Koblenz.

Doran J E and Gilbert N, (1994). "Simulating Societies: an Introduction." In *Simulating Societies* (Eds. N. Gilbert and J.E. Doran) UCL Press

Epstein J M (2002), "Modeling civil violence: an agent-based computational approach." *PNAS, Vol. 99*, suppl. 3, 7243-7250.

Foran J (1977) "The Comparative-Historical Sociology of Third World Social Revolutions: Why a Few Succeed, Why Most Fail" In *Theorizing Revolutions* (ed. J Foran), London: Routledge pp. 227-267

Gann L H (1971) *Guerrillas in History* Hoover Institution Press: Stanford, California.

Gilbert N and Troitzsch K, (1999) *Simulation for the Social Scientist* UCL Press: London

Hart P. (2003) *The I.R.A at War 1916-1923* Oxford

Lawrence, T. E. (1929) "Guerrilla Warfare", *Encyclopaedia Britannica*, 14th ed., 1929, Vol. 10, p 953

Mao Tse-Tung and Che Guevara (1962) *Guerrilla Warfare* Cassell: London (with a foreword by B. H. Liddell Hart).

Mittu R. (2004) "Towards the Interoperability of Coalition C41 Systems and Simulations in the Global Information Grid via Web-Services and the use of C2IEDM", In *Knowledge Systems for Coalition Operations (KSCO 2004)* (Eds. M. Pechoucek and A. Tate), pp. 125-135, Czech Technical University: Prague.

Raczynski S. (2004), "Simulation of The Dynamic Interactions Between Terror and Anti-Terror Organizational Structures" *Journal of Artificial Societies and Social Simulation*, vol. 7, no. 2, <http://jasss.soc.surrey.ac.uk/7/2/8.html>

Sederberg, Peter. C. (1989) *Terrorist Myths: Illusion, Rhetoric and Reality* Prentice-Hall: Englewood Cliffs, N.J.

Sun Tzu (1963) *The Art of War* (tr. S.B Griffith). Oxford University Press

Tabor R (1970) *The War of the Flea*, Paladin: London.

The Author

Jim Doran is an emeritus professor in computer science at the University of Essex in the UK. His research experience is in artificial intelligence, agent-based modelling of social systems, and computational archaeology. He has a particular interest in the Irish War of Independence (1919-1921).

An Ontology-based Integration Mechanism for Web Searching in Distributed Digital Content Repositories

C. Alexakos[1], K. Votis[1], B. Vassiliadis[2] and S. Likothanassis[1]
[1] *Pattern Recognition Lab., Dept. of Computer Engineering and Informatics, University of Patras, Greece*
{alexakos,botis,likothan}@ceid.upatras.gr

[2] *Digital Systems and Media Computing Laboratory, Computer Science, Hellenic Open University, Greece*
bb@eap.gr

Abstract

Distributed digital data integration is significant for the enforcement of novel searching mechanisms in the internet. The great heterogeneity of web based systems storing and providing digital data requires the introduction of interoperability aspects in order to resolve integration problems in a flexible and dynamic way. Our approach introduces an advanced search mechanism which initializes a semantic model representing the digital content stored in distributed servers, through the use of ontologies. Searching tasks are carried out in the metadata level, where information concerning web digital content is published, managed and stored in the form of a scalable description of knowledge domains.

1. Introduction

The wealth of information available in the internet or local corpora has increased while our ability to search and retrieve relevant information is being reduced. Hypermedia and digital media, being complex information objects, are much more difficult to manage especially when they reside in distributed servers. Semantic Web technologies have been proposed in order to enable machine-to-machine interaction that in turn will facilitate truly efficient searching.

While hypertext searching has seen some significant breakthroughs in the past few years, multimedia searching has still a long way to go. The vision of a media-aware semantic web is one of the more exciting challenges faced by researchers of many scientific disciplines including those of hypermedia, information retrieval and distributed systems [17]. In this context, searching in distributed and heterogeneous sources was always difficult but it seems that semantics may offer solutions to such unsolved problems [13].

In this paper we propose a new scheme for searching hypermedia sources using a multi-layer ontology. Searching tasks are carried out in the metadata level, where information concerning hypermedia objects is published, managed and stored.

2. Related Work

The hypermedia community has already recognized the need for good search and query mechanisms in hypermedia systems [18, 20]. Halasz in [5] forecasted the need for both content-based and structure-based retrieval on hypermedia. This issue raises the need of a detailed description of hypermedia that is not only based on contents (represented by keywords) but also on semantic contents and contextual information. Early works have already recognized the need for managing distributed hypermedia but the appropriate technologies were missing [4]. In the following years, the need of knowledge representation combined with a set of rules and concepts has led to the evolution of ontologies as the main tool to describe data contents and their relations in modern information systems. An early definition presented in [15] describes ontologies as: “a hierarchically structured set of terms to describe a domain that can be used as a skeletal foundation for a knowledge base”. An ontology’s main feature is machine readability and understandability which in turn enable automatic cross-communication of different systems. This leads to increased platform independence as well.

Ontologically principled mechanisms and frameworks for hypermedia have been presented

recently. Proposals such as the one of [11] suggest that semantics should be included in the conceptual modelling stage of hypermedia production. Topia [13] is an architecture for domain-independent processing of semantics and discourse into hypermedia presentations. In [6], a framework and a searching algorithm for locating distributed hypermedia in a P2P network is presented. Approaches such as semantically indexed hypermedia [16] and ontology-based linking [1] are also worth mentioning.

The introduction of the Semantic Web and its, nearly, unanimous approval has driven towards the representation of ontologies in semantics. For this purpose, a variety of semantic markup languages has been developed, based on the dominant XML standard. The most prominent ontology markup languages are DAML+OIL and its successor OWL, which is build on top of RDF Schema. The integration of hypermedia and semantic web technologies has been proposed both for standard [12] and open hypermedia configurations [3], for ontology matching [2] and management [9] while searching in distributed environments. Ontology schemes have been successfully used/designed for centralised management of images [10], XML document schemata [8], MPEG-7 semantically enriched content [19]and query processing in P2P[14].

Hot issues arise when integration is focused on the contents and not on the heterogeneous semantic metadata: lack of adequate knowledge representation methods, time delays in searching distributed hypermedia sources and restrictions due to the specificity of current ontology schemes used for searching.

One of the initial solutions to distributed hypermedia content integration was the use of a single global ontology scheme in order to describe all the contents of hypermedia systems. This centralised approach has proven to be inflexible since sources are exponentially increasing affecting query response times. On the other hand, content-based integration faces three distinct difficulties. First, the utilization of a single description scheme forces the usage of a specific ontology for the semantic annotation of hypermedia contents. Second, when the hypermedia information is provided by distributed servers, the amount of semantic descriptions is increasing according to content volume. Third, in some cases ontologies used are domain-specific. Specificity deters widespread use.

3. Ontological Search Model

Our approach introduces an integrated search model that consists of ontology terms and instances in order to provide a complete conceptual description about the domain of knowledge of the digital data that is provided from the distributed servers. This model is constructed based on the metadata information that is describing the digital content is its server.

The utilization of ontologies in the proposed model is based on the feature that they give a specific description of a domain, where its terms and their relationship is clearly defined. The terms are organized in a hierarchical structure and the relationships consist of HAS-A and IS-A associations. The main feature of ontologies is that the knowledge they describe can be noticeable from different users and be used for platform independent implementation.

The searching model is composed of two functional parts, the first part is the ontology scheme and the second is a set of instances of the ontology scheme containing the metadata of the integrated digital data. The distinction of these two parts is similar with the T-Box and A-Box distinction which is drawn in Description Logics[21].

The ontology scheme, that is called Integrated Data Ontology (IDO), is primarily used as a “catalogue” for the type of contents and the domain of knowledge that are integrated. Knowledge domains are specified in classes and subclasses providing a hierarchical model presenting all the knowledge fields that are included in the hypermedia contents of the distributed servers. There are also a number of properties denoting the relationship between classes. Also provides a semantic detailed description of the data that is corresponding to each knowledge domain. The semantic representation of knowledge domains follows the same level of detail with the semantic metadata schemes used by the hypermedia servers. For example, in a system that provides books the metadata is structured by elements as “Book Title”, “Book Author” and “Book Abstract”. According to IDO scheme there is class “Book” with the corresponding metadata elements as properties, namely there are properties of the class “Book” as “Title”, “Author” and “Abstract”. This ensures that all terms and their relationships utilized by each digital content provider server separately are included in the ontology scheme.

The set of instances, named as IDO Instances, comprises the content of the metadata information provided for the description of the distributed digital data in terms of the IDO ontology scheme. The IDO instances are constructed according to mapping information that depicts the association of the metadata elements to the terms of the IDO scheme. The content of the metadata is transformed by creating instances of the classes of the IDO scheme and fill them with the metadata data according to the mapping information. In

the example of books, there is an IDO instance of each book where the property of "Title" is take the values of the metadata element "Book Title". This transformation simplifies the searching procedure by carrying out the search in ontology instances instead of composing different search queries to the distributed data systems.

The semantics of the Integrated Data Ontology and IDO Instances are composed according to the Ontology Web Language (OWL) [22] standard, a recommendation from W3C as a semantic markup language for representing ontologies based on the dominant XML/RDF standard.

4. Constructing the Search Model

As presented above our approach introduces an ontological search model in order to both indexing the digital data and integrating the metadata description information. In this context, a complete description of different digital contents is possible. What is still missing is the introduction of the metadata information to the semantic information represented in this model.

A methodology is required to address the introduction needs between the different implemented metadata information of a digital content provider system and the proposed ontological model providing the aforementioned semantic information.

The first step in this methodology concerns the composition of the Integrated Data Ontology with additional information provided from the digital catalogue and the metadata structure of each digital content system. This step involves the enrichment of the Integrated Data Ontology with new classes and properties in case that there are not exist in the already developed ontology scheme. This semantic information stems from the metadata description of the digital content and has to be introduced to the proposed model by creating, if it is necessary, corresponding classes and properties of the relevant domain description.

The second step in our methodology regards the transformation of the actual metadata information in the different digital content systems according to the Integrated Data Ontology scheme. This transformation leads to the enhancement of the IDO Instances with the metadata information of the data that is stored in a new imported system. This step includes the mapping of the terms of the systems metadata structure to the already developed ontology terms in the Integrated Data Ontology. This mapping is related to the description, in ontology terms, of the metadata content (e.g. which server provides what). Furthermore, an update of the IDO Instances with data according to the mapping information of the actual metadata content is following.

This methodology leads to the desirable unification of the ontological search model with the metadata described digital contents, and thus contributes to the creation of a flexible and reusable ontology search scheme. It is essential to be mentioned that the same methodology is adopted in case of updating the digital content in one of the distributed systems. The synthesis of the Integrated Data Ontology and the IDO Instances constructs a model that contains all the necessary semantic information for searching and locating the desirable data avoiding the executing of a query to distributed servers.

5. Creating the appropriate Web Services

The presented ontological search model utilizes an integration solution to the searching process in an amount of data located in distributed server. The next step in our proposal is to present the data extraction and transfer from the corresponding system. In this context, our work in this paper proposes the use of web services implementation in order to open up the distributed digital content systems.

In [23] WSs are referred as "software applications identified by a Uniform Resource Identifier (URI), whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and support direct interactions with other software applications using XML-based messages via Internet-based protocols". WSs are reusable software building blocks distributed over the Web easily accessible via widespread protocols like HTTP and SMTP. They are loosely coupled, communicating through XML based documents.

According to the prospects that are supported by the web services, each of the integrated systems is enhanced with a SOAP server in order to provide a flexible and standard based service for transferring the desired digital content. A client script is also installed in the integrated middleware in order to invoke the provided web services from the distributed systems.

The utilization of web services is based on the capability of the easy integration that is achieved through an intermediate adapter layer that relays commands and data to the web from the system and vice versa. A lot of implementation has introduced according the exposition of systems capabilities and services to the internet, such .NET framework and Common Object Model (COM) for Microsoft based applications, Enterprise JavaBeans (EJB) for java based applications and PHP classes for web based applications.

6. Searching Process

The proposed scalable scheme provides the necessary supporting mechanism to a search engine to navigate through the ontology terms and instances faster and efficiently and finally to transfer the resulted data from the associated systems. The basic concept is the separation of the search process in three steps.

In the first step, the engine searches in the Integrated Data Ontology aiming to find the proper general domain (or domains) where the results may be included and the appropriate structural ontology terms that describe the corresponding semantic metadata structure.

In the second step the search engine has extracted from step one the basic structural elements of the semantic metadata that will be used in the hypermedia content searching. In this step a query is composed in according to the RDF Data Query Language (RDQL) querying language for searching in the IDO Instances. RDQL is querying language for RDF structured documents, which is the base language for OWL language. The RDQL queries consists of a graph pattern, expressed as a list of triple patterns and also can have a set of constraints on the values of those variables, and a list of the variables required in the answer set.

In the final step there is the elaboration of the results of the RDQL query in order to locate the systems that contain the resulted data. Furthermore, the appropriate client scripts are executed in order to invoke the corresponding web services and acquire the desirable digital content.

8. Integrated Architecture

The enforcement of the above presented ontology search model requires a specific architecture that makes it possible to integrate the different semantically annotated digital data residing in different servers in a flexible and interoperable way. Our approach

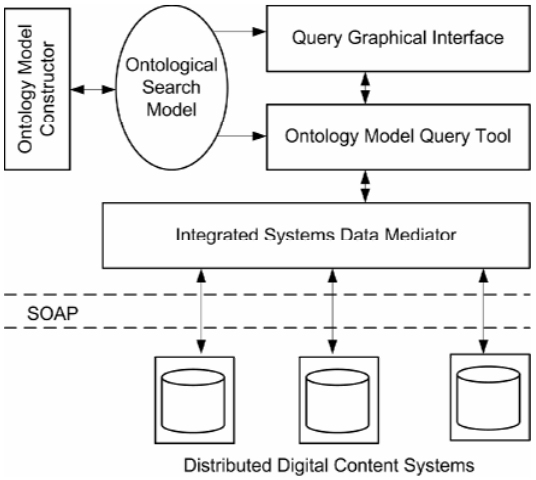


Figure 1. Architecture

introduces a middleware for developing the multilayer ontology scheme and managing the search queries from the users. This middleware, presented in figure 1, involves the following functional elements:

The Ontology Model Constructor that provides the mechanism and the graphical interface for composing the Integrated Data Ontology and creating the mapping information between the ontology terms and the metadata content. Also provides a functional module for transforming the actual metadata information in order to upgrade the IDO Instances according to the corresponding mapping information.

The Query Graphical Interface providing the appropriate search forms to the user of the integrated system.

The Ontology Model Query Tool that is responsible for the execution of the two first steps of the search process to the ontology search model..

The Integrated Systems Data Mediator is the platform which accepts the results of the queries in the ontology search model and invokes the appropriate web services in order to acquire the desirable data.

The integration consists of two discrete phases: a) the Construction Search Model Phase where utilizing the functionalities of the Ontology Model Constructor the Integrated Data Ontology is defined and the IDO Instances are updated, and b) the Search Process Phase where a guest user is using the Query Graphical Interface to compose queries that are processed from the Ontology Model Query Tool and in the next step the Integrated Systems Data Mediator is taking the responsibility to allocate and bring the resulted data to the user. The overall proposed architecture is depicted in Figure 1.

Figure 2. Integrated Data Ontology for books

7. Use case

In this section we show an example use case of the proposed methodology and architecture. More specifically we have chosen two digital libraries with cultural content: the Cultural Heritage of Municipality of Pyrgos (CHMP) and the Religious Heritage of Orthodox Metropolis of Kalavryta and Aigeiala (RHOMKA).

The CHMP is a digital library with cultural content as books, paintings and architectural monuments that are indexed and described with a composite metadata scheme according the guidelines of CIDOC Conceptual Reference Model and adding some custom specified elements. The RHOMKA contains digitalized religious contents as books, icons and other sacred appliances according to the Dublin Core standard.

For the simplicity of the presentation of the use case a demonstration of a searching query about finding books of a specific author will be used for the presentation of the two phases of our proposed work.

7.1 Construction Search Model Phase

In this section we present a part of the composed Integrated Data Ontology that is describing the digital data related to books. The ontology scheme contains three classes named as "Cultural_Heritage", "Regional_Heritage" and "Book". The relationships between these classes are represented by the object property "HasPublications" as depicted in Figure 2. The "Book" class is characterized by a set of data properties: "Title", "Abstract", "Author" and "NumberPages".

In the next step there is the mapping of the metadata of the two e-culture digital libraries to the presented ontology. The mapping includes the association of the elements of Dublin Core and CIDOC CRM Standards to the data properties of the "Book" class. For example, according to Bublin Core standard each book is described by the metadata elements "title", "creator", "description", e.t.c.. Using the Ontology Model Constructor the following mapping information are created: "title" is a associated with the property "Title", "creator" with "author", "description" with "abstract". According to this mapping information, the metadata contents from the two systems are transformed in order to upgrade the IDO Instances set. The executing this action denotes the completeness of the Construction Search Model Phase.

```
SELECT  ?booktitle

WHERE

(?book,
<http://prlab.ceid.upatras.gr/ido#Author>,
"x_author")

 (?book, <http://a.com/ontology#Title>,
?booktitle)
```

Table 1. Search Book RDQL query

7.2 Search Process Phase

The Search Process Phase in this demonstration includes the search process that will query the integrated digital libraries to retrieve the data that is associated with the books of a particular author named as "x_author".

A guest user is set this query using the forms of the user interface and thereafter it is feed to the Ontology Model Query Tool. The user's query is transformed to a RDQL query that is depicted in Table 1.

The result of the query is used as an input to the Integrated Systems Data Mediator that has stored an identifier for each instance in the IDO Instances for recognize the server that is located the corresponding data.

The two digital libraries that are used in this use case have been integrated utilizing web services technology. There are, in both systems, SOAP servers that are exposing retrieving methods as services over the web. Also, the Integrated Systems Data Mediator has stored the corresponding client scripts in order to invoke the provided web services of the systems.

The Integrated Systems Data Mediator will locate the server (or servers) where the data is stored. The last step is to invoke the appropriate web method from the two digital libraries for each data included in the query result. The data is transferred from the two systems and is presented to the user.

8. Conclusion

We have presented an Ontology-based Integration Mechanism for integrating distributes cultural data. The process introduces an ontological search model in order to both indexing the digital data and integrating the metadata description information. A search methodology process has been defined for the desirable unification of the ontology model and a prototype based on searching a set of distributed cultural heritage databases has been implemented to test the feasibility of the mechanism. As we progressed through the implementation some interesting issues emerged and have us looking at an extension of our current model that incorporates more types of edges and domain functions into the ontology model.

9 Acknowledgements

The authors would like to give their special thanks to Mr. Wernher Behrendt for his helpful comments on a draft of this paper.

10. References

[1] Carr, L., Hall, W., Bechhofer, S., Goble, C., (2001). Conceptual linking: ontology-based open hypermedia. Proceedings of the 10th international conference on World Wide Web, pp. 334 – 342.

[2] Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A., (2003). Learning to match ontologies on the Semantic Web. The VLDB Journal - The International Journal on Very Large Data Bases, Vol.12(4), pp. 303 – 319.

[3] Dolog, P., Henze, N., Nejdl, W., (2003). Logic-Based Open Hypermedia for the Semantic Web. In Proc. of International Workshop on Hypermedia and the Semantic Web, Hypertext 2003 Conference, Nottingham, UK.

[4] Goose, S., Dale, J., Hill, G., de Roure, D., Hall, W., (1996). An Open Framework for Integrating Widely Distributed Hypermedia Resources. Proceedings of the 1996 International Conference on Multimedia Computing and Systems (ICMCS '96), pp. 0364.

[5] Halasz, F. (1988). Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. Communications of the ACM, Vol. 31, No. 7, July, pp. 836-852

[6] Larsen, R.D., Bouvin, N.O., (2004). HyperPeer: searching for resemblance in a P2P network. Proceedings of the 15th ACM conference on Hypertext & Hypermedia, pp. 268 - 269

[7] Li, WS., Candan, K.S. (1999). Integrating content search with structure analysis for hypermedia retrieval and management. ACM Computing Surveys (CSUR), Volume 31, Issue 4es, Article No. 13.

[8] Lu, E.J.L., Jung, Y.M., (2003). XDSearch: an efficient search engine for XML document schemata. Expert Systems with Applications, 24, pp. 213–224.

[9] Maedche, A., Motik, B., Stojanovic, L., (2003). Managing multiple and distributed ontologies on the Semantic Web. The VLDB Journal - The International Journal on Very Large Data Bases, Vol. 12(4), pp. 286 – 302.

[10] Mezaris, V., Kompatsiaris, I., Strintzis, M.G., (2004). Region-based Image Retrieval using an Object, Ontology and Relevance Feedback. Eurasip Journal on Applied Signal Processing, Vol. 2004 (6), pp.886-901.

[11] Montero, S., Diaz, P., Aedo, I., Dodero, J.M., (2003). Toward Hypermedia Design Methods for the Semantic Web. Proceedings of the 14th International Workshop on Database and Expert Systems Applications, pp. 762.

[12] Nanard, M., Nanard, J., King, P., (2003). IUHM: a hypermedia-based model for integrating open services, data and metadata, Proceedings of the 14th ACM conference on Hypertext and hypermedia, pp. 128 – 137.

[13] Rutledge, L., Alberink, M., Brussee, R., Pokraev, S., van Dieten, W., Veenstra, M., (2003). Hypermedia semantics: Finding the story: broader applicability of semantics and discourse for hypermedia generation. Proceedings of the 14th ACM conference on Hypertext and hypermedia, pp. 67 – 76.

[14] Stuckenschmidt, H., Giunchiglia, F., van Harmelen, F., (2005). Query Processing in Ontology-Based Peer-to-Peer Systems. Ontologies for Agents: Theory and Experiences, Whitestein Series in Software Agent Technologies.

[15] Swartout, B., Patil R., Knight K., Russ T. (1996). Toward distributed use of large-scale ontologies. In Proceedings of the 10th Knowledge Acquisition for Knowledge- Based Systems Workshop.

[16] Tudhope, D., Cunliffe, D., (1999). Semantically indexed hypermedia: linking information disciplines. ACM Computing Surveys (CSUR), Volume 31, Issue 4es, article No. 4.

[17] Van Ossenbruggen, J., Nack, F., Hardman, L., (2004). That Obscure Object of Desire: Multimedia Metadata on the Web, Part 1. IEEE MultiMedia, Vol. 11(4), pp. 38-48.

[18] Vitali, F., Bieber, M., (1999). Hypermedia on the Web: what will it take?. ACM Computing Surveys (CSUR), Volume 31, Issue 4es, article No. 31.

[19] Westermann, U., Klas, W., (2003). An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions. ACM Computing Surveys (CSUR), Vol. 35(4), pp. 331–373.

[20] Wiil, U.K., Nürnberg, P.J., Leggett, J.J., (1999). Hypermedia research directions: an infrastructure perspective. ACM Computing Surveys (CSUR), Volume 31, Issue 4es, Article No. 2.

[21] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, P. Patel - Schneider, Description Logic Handbook - Theory, Implementation and Applications, Cambridge university press, (2003)

[22] Web Ontology Language (OWL), W3C, http://www.w3.org/2004/OWL/

[23] Web Services Architecture W3C Working Draft 14 May 2003

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).

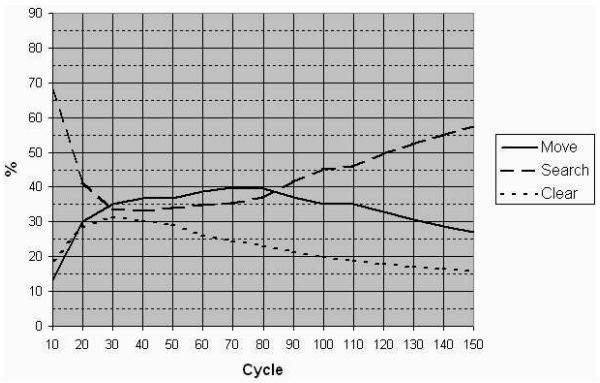


Figure 1. I-Kobe simulation results.

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** 04
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 4**
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.

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.

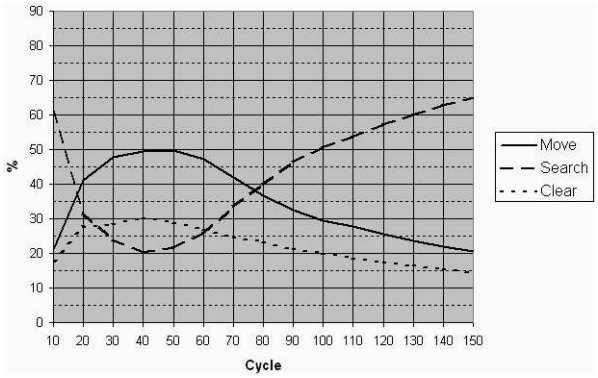


Figure 2. Collaborative version results.

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 a_2, however their possible plans are in conflict with a_2. 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 activi-

ties 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/`

The Role of Roles in Robotics

Matthew T. Long and Robin Murphy
Institute for Safety, Security and Rescue Technology
4202 E Fowler Avenue
Tampa, Florida 33620
{*mtlong,murphy*}*@cse.usf.edu*

Jim Hicinbothom
CHI Systems, Inc.
11838 Bernardo Plaza Court, Suite 102A
San Diego, CA 92128-2413
jhicinbothom@chisystems.com

Abstract

In human social systems, roles are a key mechanism for social interaction and integration. This paper argues that robots can use social roles to enable behavior based on a study of roles from the social sciences, as well as related work in software agents. It poses a novel categorization of the influence of autonomy and reasoning on role mechanisms. This categorization identifies the areas of research needed to build an artificial social system that fully mimics the behavior of natural social systems. The paper also provides a survey of the current state of role-based robotics. Finally, it offers an example of how roles can enable domain integration by extending previous work in defining a robot persona, first presented for allocating resources within distributed, heterogeneous teams of robots, with the addition of roles.

1 Introduction

Multirobot systems are currently used in a wide variety of situations, from military applications to search and rescue to off-world exploration. Unfortunately, there is little interoperability between domains – existing approaches to multirobot system design incorporate domain knowledge directly in the robot software and system designs [20; 13]. Adaptation to a new domain will be important for the rapid deployment of robot systems, particularly for an organization such as Institute for Safety, Security and Rescue Technology (iSSRT) which responds to Urban Search and Rescue events on short notice. Interoperability could allow robot systems designed for use with US search and rescue teams to be integrated into an agent-based multinational rescue operation of the sort described in [25; 22].

In human social systems, roles are a key mechanism for social interaction and integration. Roles define the interactions of individuals within a social context, and are inseparable from the social context. Since roles allow individual human persons to operate in a wide variety of domains on a regular basis, we feel the connection between this concept and artificial social systems should be investigated.

This paper makes four claims about social roles and robotics. First, robots can use social roles to enable behavior. We base this claim on a foundation of roles from the social sciences, as well as related work in software agents. Second, we pose a categorization of the influence of autonomy and reasoning on role mechanisms. This categorization shows that there is a great deal of research needed to build an artificial social system that fully mimics the behavior of natural social systems. Third, we provide a survey of the current state of role-based robotics. Fourth, this paper extends previous work in defining a robot persona, first presented for allocating resources within distributed, heterogeneous teams of robots in [19], with the addition of roles to enable domain integration.

2 Survey of Roles

In order to facilitate a consistent discussion relating work in artificial roles, this section summarizes terminology related to roles from a sociological standpoint. This section then examines the influence of autonomy and reasoning on roles in artificial role-based systems, a major contribution of this work. Finally, we investigate topical related work in agent-based and robotic systems.

2.1 Roles

Roles can be viewed in light of both a *static* description (*role dimensions*) and a *dynamic* interaction (*role mechanisms*). Jahnke, *et al.* [17], define four static dimensions (position, functions, expectations and interactions) and six dynamic mechanisms (role assignment, role change, role making, role taking, inter-role conflict, and role definition). These ten characterizations form the basis for a discussion of roles and are described briefly below:

1. *Position*: Roles reflect a static view of an organizational structure. The position of a role in a social structure defines the functions and tasks that are required for the role.

2. *Functions and tasks*: Each role is associated with a formal set of permissions, obligations and activities that are defined by the social organization. This role dimension associates a role with how it is performed. A person playing a search specialist role is required to search a disaster area according to a predefined protocol. A role my also incorporate restrictions or qualifications; in this example, a search specialist may be unable to search a building until a structural specialist has investigated the safety of the building.

3. *Behavioral expectations*: If the functions and tasks of a role are a formal description of the role, the behavioral expectations are informal conventions bound in the social interactions of a role. Violating conventions can earn an agent negative sanctions from other agents in the community.

4. *Social interactions*: Finally, a role player's ability to play the role can feed back and shape the role definition itself, whether through modifying the role's behavioral expectations, core functions, or even position in the social system itself.

If the four role dimensions are a *structural description* of a role and its relations, the following six dynamic relations, known as role mechanisms, are a *functional description* of how an individual interacts with roles:

1. *Role assignment*: Role assignment is the process of assigning a role to an agent in a social system. Role assignment is a statement of desire, not a guarantee that the agent will accept or be able to play the role.

2. *Role taking*: Role taking occurs when an agent uses the known role played by another agent to build a model of the agent's behavior. Coutu [10] noted that role taking is often erroneously confused with role assignment.

3. *Role change*: An agent may play one or more roles simultaneously or in sequence, and role change is the process of relinquishing a role and assuming another.

4. *Role making*: In a human society, each person that plays a role does so in their own unique manner, and transforms the behavioral expectations into concrete action in different manners.

5. *Inter-role conflict*: An agent may hold multiple roles, and the goals of each may conflict.

6. *Role definition*: Over time the definition of a role may change. This may be due to changing circumstances, where a role may be radically changed or a new role created, or due to a negotiated social change in the behavioral expectations due to the role making of different actors.

2.2 Autonomy, Roles and Reasoning

Natural social systems have a particularly large advantage over artificial social systems – all or almost all the members of a natural system are autonomous entities with a high level of reasoning ability. Artificial social systems do not have this advantage. Members of these systems are typically endowed with much less autonomy and computational power. In systems where none of the artificial agents have the requisite abilities, the role dimensions and mechanisms must be simulated or processed by a human developer or designer and may not operate in real time. Thus the effect of both autonomy and reasoning ability on these role dimensions and mechanisms and artificial system design must be considered to successfully design these social systems and the agents within them. This section discusses the relationship of the four role dimensions and six role mechanisms to autonomy and reasoning and offers a graphical mapping to relate these concepts.

Of the four role dimensions, the *position* and formal *functions and tasks* are of primary importance to artificial social systems and the implementation of roles. In particular the position is a description of how the different roles relate to each other, essentially defining the social structure of the domain. This has a large impact on the norms of the system, the rights, restrictions and permissions one role is granted over another. The functions describe the particular obligations of the role player when the role is assumed. The reasoning ability of a software system is of particular importance in light of these two role dimensions; a reasoning system can use a formal definition to infer information about the interaction of role obligations, right, permissions and other roles.

The *behavioral expectations* and *social interactions* between roles are of a less direct relevance as they are both informal or implicit expectation of behavior. However, as Hicinbothom, *et al* [16] note, experts often rely on this implicit knowledge to boost performance. Autonomy has more of an impact on behavioral expectation and social interactions. Both require the ability to vary from a preplanned behavioral script and the ability to react to other agent's behaviors that do not conform tightly to a formal specification.

The six role mechanisms are particularly dependent on both autonomy are reasoning. Figure 1 presents a novel characterization of the relative system requirements for

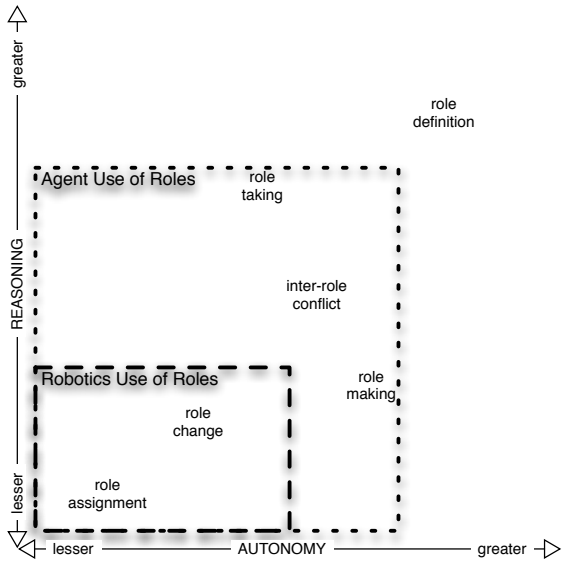


Figure 1. The influence of autonomy and reasoning on role mechanisms

each of the role mechanisms as well as the current utilization in both agent (Section 2.3) and robot systems (Section 2.4). Role assignment can be handled by a system with limited autonomy and reasoning capabilities through a role or task allocation mechanism such as that described in [15] and [18]. However, anticipating the behavior of other entities in the system (role taking) requires a great deal more reasoning ability (to anticipate actions) and autonomy to change behavior based on the predictions. Role definition, the creation of new roles, requires the most reasoning ability and autonomy to handle successfully.

2.3 Agents Using Roles

The bulk of prior work with roles in multi-agent systems has appeared out of the software agent community, and is relevant to multi-robot systems. Work influencing the static role description in agent systems has included social norms and policies [4; 14] as well as social enforcement [3; 6]. Wooldridge, *et al.* [27], in particular, observe that multi-agent system design is more complicated than traditional software engineering, which fails to capture an agent's problem solving behavior and social interactions. They propose the GAIA methodology for agent-oriented design, using roles to model the responsibilities, permissions, activities and protocols of an agent. In a sense, though, this methodology and Zambonelli's [28] extension to incorporate organizational structure, only describe the first two static dimensions of role analysis: position and function.

In open multi-agent systems, where agents can enter or leave the system, it is not enough to simply describe the static role dimensions. These agents need to incorporate role dynamics as well. Dastani, *et al.* [11; 12] have studied dynamic role assignment in such open societies, and Boissier, *et al.* [5] look at how an individual's cognitive framework and behavior can change when playing a role, and describe how the role can influence the individual's goals, desires and beliefs, as well as cause the individual to dynamically gain or lose influence or power. A reasoning agent may very well agree to play a role to gain access to information that will help with private goals even if it does mean accepting new restrictions or obligations.

Colman and Han [9] have investigated autonomy and how it can relate to roles and agency. The propose five-levels of autonomy: no autonomy, process autonomy, system-state autonomy, intentional autonomy, and autonomy from constraints. This view of autonomy maps to that described in 2.2. However, the general level of implementation of the role dimensions and role mechanisms is unclear, and autonomy has not been investigated in this context.

2.4 Robots Using Roles

Multi-robot systems is a less developed field then general multi-agent systems. As a result, research on strong social interactions between robots and well-defined multi-robot domains have been slower to develop. However, there has been some recent work in this area. As field robotics is inherently failure-prone [7] and is thus by definition an open agent environment, it is natural that research will lead in the direction of the dynamic characterization of roles.

Roles appear in limited form in previous robotics literature, but they do so primarily as a synonym for *task*, without many of the stronger social aspects; these uses of roles appear to be strictly limited to role-assignment and role-change. Stone and Veloso [23] used roles in this manner, building formations composed of a set number of specific roles for robot soccer. As the soccer game progressed, formations (and thus roles) would change dynamically. Roles also appeared in Martinson and Arkin [21] as a vehicle to test a Q-learning-based role-assignment mechanism using a foraging task in a hostile environment. Finally, Støy [24] used roles in a more recursive manner for self-reconfigurable robots; each reconfigurable module could assume a role within the robot such as "leg" or "spine". Role selection and assignment provided the required behavior for the module within the social context of the robot, but did not influence how the robot interacted with other agents.

3 The Future of Roles for Robot and Agent Systems

Roles are used in both robot and agent systems to define and regulate behavior. Their use, however, seems to hold particular promise for enabling integration of robots or robot teams into a variety of different domains. Robots are physical, situated agents that are expensive to build and prone to failure in adverse conditions. Thus a particular system is likely to have a limited number available for use and it may be necessary to "share" robots between domains.

As a motivating example, two sample domains are examined. The first is a project to simulate a manned mission to Mars [8] which uses a robot as an integral part of the project to assist human scientists with science missions. The second is a high-level planning and control scenario with the goal of directing a number of coalition groups for a search and rescue operation described in [25].

The Mars analog project [8] is an example of a complex domain where the actors in the system have defined roles. While there are a number of human and software agents in the system there is also an EVA Robotic Assistant (ERA) for science missions. The ERA can perform the following tasks: capture digital panoramic images, carry tools and science samples, move to designated locations, print labels for the rock and soil samples, autonomously follow an astronaut, and it can function as a network relay node. Thus it can perform a number of roles, which may change based on mission parameters and events. If this robot were to fail at a critical time, it might be necessary to recruit a robot from some other mission to fill some of these roles.

The Coalition Search and Rescue - Task Support (CoSAR-TS) project [25] had the goal of creating a system to link the capabilities of a coalition of search and rescue organizations. The system knowledge base contained important data including country information, hospital locations and services, and asset capabilities such as a rescue helicopter with the ability to pick up a downed pilot. In addition, there were agents with predefined roles such as SAR coordinator or hospital information provider. This domain description does not include the use of robots directly, but it is certainly plausible that robot search assets could be used in some manner. In this case, the robots would need to be able to fill roles as needed.

4 Current Work

The concepts expounded here are currently being used to adapt a team of heterogeneous ground vehicles for use in a military operation domain by adding role to the USF Distributed Field Robot Architecture via the addition of a *domain adapter*. This section presents a preliminary view of this work and discusses how the use of roles is built on prior work.

In this military operation domain, a robot is intended to fill three basic roles: *AutonomousNavigator*, *Scout* and *Investigator*. The robot's *position* in this particular domain is of limited autonomy; each role and a goal plan for each role is defined by a cognitive agent residing external to each robot. The following is a high-level description of the *function* of these roles:

AutonomousNavigator: When a robot is assigned this role, it is expected to navigate following a pre-planned high-level path. The cognitive agent does not have a detailed map of the terrain, so the on-board cartographer is expected to modify the path to avoid unmapped obstacles encountered along the way.

Investigator: The investigator role is an information-gathering role, where the robot is expected to sense as much information as possible about a target location or object.

Scout: The Scout role is a combination of AutonomousNavigator and Investigator; in this role the robot acts as a remote sensing platform for the cognitive agent.

Not all of the role mechanisms are necessary for this domain. Dynamic *role-definition* is not an option – the three static roles are pre-defined. *Role-taking* is performed by the cognitive agent in this domain as it plans the mission scenario, and there is little role-based interaction between robot agents during mission execution. However, each robot must be able to handle *role-assignment* , *role-change* and *inter-role conflict*. These three mechanisms are critical for proper operation of the robots in the domain; each robot must be able to accept new roles, change roles dynamically and resolve basic conflicts between role goals. The final mechanism, *role-making*, is intentionally limited. However, as each robot executes a set of reactive behaviors, the interaction between active behaviors and a dynamically changing environment will result in an overall emergent behavior that is difficult to predict. It is thus possible that each robot could, for the same set of assigned roles, produce a slightly different action.

The *domain adapter* is designed to translate information from a robot and present it in new domains. It is the domain adapter that constructs and defines the roles of the robot in each domain, maintaining static role state and implementing the role mechanisms to the extent allowed by the domain and robot characteristics. The domain adapter is also responsible for matching a role's functions and tasks with the appropriate robot implementation. For simple functions, this may be a mapping to or from an internal data type, sensor reading or percept. More complex functions may require complex behavior and social interaction by the robot.

These are typically represented as low-level behaviors or high-level scripts that can sequence groups of behaviors [2]. In the example domain, information in the persona can be used by the cognitive agent to make decisions about role assignment.

The robots are programmed using USF's DFRA [19] and use a key concept from previous work, the *persona*. DFRA is a three-layer architecture, incorporating a reactive layer for behavior-based control, a deliberative layer for high-level reasoning, and a distributed layer for multi-agent communication. The distributed layer of the architecture incorporates concepts from artificial intelligence and software agents and is loosely based on the persona concept from psychology and sociology. The persona provides a metaphor for thinking about the robot and how it interacts with other robots and intelligent agents. DFRA allows the development of distributed algorithms and decision making and allows the robots to work with software and cognitive agents in a larger informational system. While DFRA does not presently have a built-in method for handing social norms, such as permissions and obligations, it can interface with the Knowledgeable Agent-oriented System (KAoS) [26], thereby allowing use of a robust agent system for policy and norm management.

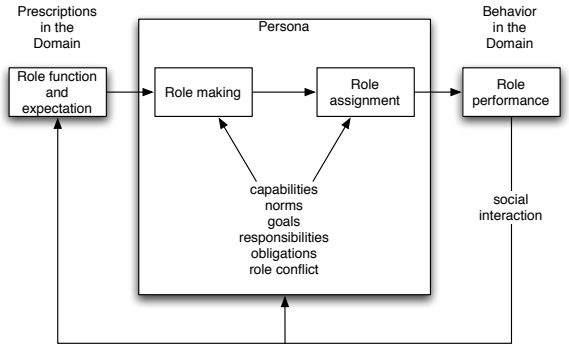


Figure 2. Interaction of persona and domain for role assignment (adapted from [1])

This approach further refines the persona concept within distributed robotics. Allport [1] notes that "role performance is a point of intersection between the personality system [persona] and the social system" (Figure 2). Not only does the persona act as an external reflection of internal state, but it is a facet of internal state that an observer needs or wants to see. This reflection can vary by observer and with circumstance. Within each domain, the robot would appear as an agent of that domain. This is a key difference between our approach and the role-modeling of [27], as the latter focuses on a set of agents to fill roles identified during an analysis phase, while the persona-based approach tries to adapt existing robots to fill new roles in new domains.

5 Conclusions & Discussion

This paper has investigated the use of roles in human social systems and summarized the literature on roles in the software agent and robotics communities. This work has three main implications for additional research.

A formal understanding of relationship between roles, agents and domains can have a practical benefit for robotics. Automated reasoning about roles through the semantic mapping of concepts from domain to domain could be used to simplify and improve adapting teams of robots or other agents to new domains. This can be critical for coalition formation, when new coalition members must be added to fulfill overall goals.

While the use of roles to enable behavior can be relatively straightforward, the impact of autonomy and reasoning ability on the role mechanisms is less well understood and is an open area of research. Most previous work on roles in artificial systems has involved agents that have limited autonomy, reasoning ability, or both. A systematic investigation of the effect of both autonomy and reasoning capability on roles and role mechanisms could uncover guiding principles for system design. Mismatches between the role mechanisms expected or required to fully function in a domain and the role capabilities of autonomous platforms could have severe implications for large-scale, flexible system design.

Artificial systems may need to model or understand roles to successfully interact with humans for extended periods or within complex tasks because roles are inherent in human social interactions. We may also find, for social interaction between humans and artificial agents, an analog to the "uncanny valley" of humanoid robotics if the role mechanisms do not work in a fully human-like manner.

Acknowledgments

The authors gratefully acknowledge the support of the Army Research Institute contracts W74V8H-04-P-0482 and W74V8H-05-C-0052. The view, opinions, and/or findings contained in this paper are those of the authors and should not be construed as official Department of the Army position, policy or decision.

References

[1] G. W. Allport. *Pattern and Growth in Personality*. Holt, Rinehart and Winston, 1961.

[2] L. E. Barnes, R. R. Murphy, and J. D. Craighead. Human-robot coordination using scripts. Technical report, Center for Robot-Assisted Search and Rescue, 2005.

[3] G. Boella and L. Lesmo. Norms and cooperation: Two sides of social rationality. In *Proc. of IJCAI Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, 2001.

[4] G. Boella and L. van der Torre. Enforceable social laws. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 682–689, New York, NY, USA, 2005. ACM Press.

[5] O. Boissier, C. Carabelea, C. Castelfranch, J. Sabater-Mir, and L. Tummolini. The dialectics between an individual and his role. In *Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems: Papers from the AAAI Fall Symposium 2005*, 2005.

[6] J. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, and R. V. Hoof. Representation and reasoning for daml-based policy and domain services in kaos and nomads. In *Proceedings of the Autonomous Agents with Multi-Agent Systems Conference*, 2003.

[7] J. Carlson, R. R. Murphy, and A. Nelson. Follow–up analysis of mobile robot failures. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 5, pages 4987–4994, April 2004.

[8] W. Clancey, M. Sierhuis, R. Alena, D. Berrios, J. Dowding, J. Graham, K. Tyree, R. Hirsh, W. Garry, A. Semple, S. Buckingham Shum, N. Shadbolt, and S. Rupert. Automating CapCom using mobile agents and robotic assistants. *American Institute of Aeronautics and Astronautics 1st Space Exploration Conference*, 2005.

[9] A. W. Colman and J. Han. Organizational roles and players. In *Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems: Papers from the AAAI Fall Symposium 2005*, 2005.

[10] W. Coutu. Role-playing vs. role-taking: An appeal for clarification. *American Sociological Review*, 16(2):180–187, Apr 1951.

[11] M. Dastani, V. Dignum, and F. Dignum. Role-assignment in open agent societies. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 489–496, New York, NY, USA, 2003. ACM Press.

[12] M. Dastani, M. B. van Riemsdijk, J. Hulstijn, F. Dignum, and J.-J. C. Meyer. *Enacting and Deacting Roles in Agent Programming*, volume 3382. Springer-Verlag GmbH, 2005.

[13] J. DiLeo, T. Jacobs, and S. DeLoach. Integrating ontologies into multiagent systems engineering. In *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)*, 2002.

[14] C. Felicissimo, C. Lucena, G. Carvalho, and R. Paes. Normative ontologies to define regulations over roles in open multi-agent systems. In *AAAI Fall Symposium on Roles- an interdisciplinary perspective, 2005*, 2005.

[15] A. Gage. *Multi-Robot Task Allocation Using Affect*. PhD thesis, University of South Florida, August 2004.

[16] J. Hicinbothom, F. Glenn, J. Ryder, W. Zachary, J. Eilbert, and K. Bracken. Cognitive modeling of collaboration in various contexts. In *Proceedings of 2002 ONR Technology for Collaborative Command & Control Workshop,*, pages 66–70, 2002.

[17] I. Jahnke, C. Ritterskamp, and T. Herrmann. Sociotechnical roles for sociotechnical systems – a perspective from social and computer sciences. In *Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems: Papers from the AAAI Fall Symposium 2005*, 2005.

[18] M. Long, A. Gage, R. R. Murphy, and K. Valavanis. Application of the distributed field robot architecture to a simulated demining task. In *IEEE International Conference on Robotics and Automation*, 2005.

[19] M. T. Long. Creating a distributed field robot architecture for multiple robots. Master's thesis, University of South Florida, November 2004.

[20] R. E. Marmelstein. Force templates: A blueprint for coalition interaction within an infoshpere, 2002.

[21] E. Martinson and R. C. Arkin. Learning to role-switch in multi-robot systems. In *ICRA*, pages 2727–2734, 2003.

[22] M. Pěchouček, V. Mařík, and J. Bárta. A knowledge-based approach to coalition formation, 2002.

[23] P. Stone and M. Veloso. Task decomposition and dynamic role assignment for real-time strategic teamwork. In J. P. Muller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 293–308. Springer-Verlag, Heidelberg, 1999.

[24] K. Støy, W.-M. Shen, , and P. Will. Using role based control to produce locomotion in chain-type self-reconfigurable robots. *IEEE Transactions on Mechatronics*, 7(4):410–417, 2002.

[25] A. Tate, J. Dalton, J. M. Bradshaw, and A. Uszok. Coalition search and rescue - task support intelligent task achieving agents on the semantic web. Technical report, AIAI, Edinburgh, UK and IHMC, Pensacola, FL, 2004.

[26] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proceedings of the IEEE Workshop on Policy 2003*, 2003.

[27] M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[28] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.

Protocol for a truly distributed coordination among agents in competitive survival video-games

Julio Cano, Javier Carbó
Applied Artificial Intelligence Group
Carlos III University of Madrid
Spain
{jcano,jcarbo}@inf.uc3m.es

Abstract

Competition games are dynamic and distributed scenarios where agent technology is often applied. In this context, agents form collaborative coalitions that compete for a goal. Inside such type of coalitions of equals, we cannot centralize coordination on a single agent nor allocate tasks on a given agent by default. A protocol is proposed here to work in such conditions. This protocol is similar to a decentralized Contract Net, but initiator agent does not centralize agent task allocation. This problem is solved through a global system rank that selects the most suited agents for the task. This interaction protocol has been implemented and tested under different simulated conditions, showing a high level of success.

1. Introduction

In recent years, there has been an increasing interest in extending the use of Internet. With this intention, automation and delegation is desirable to exploit the possibilities of spanning geographical, cultural and political boundaries. It can be achieved through the use of autonomous programs often called 'agents'.

From the very different goals that agents try to accomplish in electronic environments, this research is interested just in how they coordinate their efforts towards a shared goal. Many agent coordination protocols have already been proposed for general purpose like TAEMS [9] and GPGP [1]. But their coordination is based on specialized agents like *matchmakers*, *brokers*, etc, or like Contract Net in witch contractor agent centralizes the task allocation.

Since we intend to achieve coordination in a coalition of agents that play no special role of coordination (a society of equals) a new interaction protocol is required.

The proposed protocol is mainly a variation of Contract Net, but modified according to restrictions and conditions of a specific environment (detailed in the next section). These restrictions try to describe most video games environments where several human and/or computerized players (agents) interact and play the game.

The agents that participate in the proposed protocol are supposed to be basically situated agents, but with the ability to communicate with others through direct messages. So they can not be considered as a purely Situated MultiAgent System. The terms: *capabilities*, *characteristics* and *situation* are used indifferently along this paper, indicating whole agent situation, when applied to agents. They could reference the agent position in a map, availability to accomplish a given task, quantity of knowledge the agent has, etc. These agent characteristics are supposed to change dynamically along the time.

2. Problem description and restrictions

Most competitive videogames can be modelled as distributed and highly dynamic environments. Some of the characteristics that usually describe a video game environment are as follow:

- Dynamicity:
  Agents situation changes constantly. That makes almost useless to advertise agent services or characteristics when the agent enters into the system, because these will change along the time. The characteristics or situation of the agent should be checked every time a petition is made or a goal is marked to be accomplished, and right in that moment, so that the best agent can be selected to do the task. As agent situation or characteristics change so dynamically and constantly along the time it results useless to publish, store and centralize them like its position in a map.

- Agents falling (dying):
  Although the kind of services provided by an agent could be published and registered or concentrated in specialized agents acting as central entities like *matchmakers*, *brokers*, etc., these services will depend on agent situation at every moment. And therefore the central agent should be asking every agent about its situation before selecting one of them for a task. Moreover we are supposing a completely distributed environment in which only active agents able to interact with other agents or the environment itself are represented. So this kind of agents should be avoided. Another main reason to avoid concentration of information on a single specialized agent is that in competitive multiagent systems (such as video games), agents can enter, exit the system or even fall (being eliminated) at any time, losing all that information.
  All these reasons lead to a distributed system of agents with minimum hierarchical organization. Then, distributed coordination protocols that avoid centralization become really relevant.
- Teams (or coalitions) setup:
  In distributed agent systems such as described here teams setup is an important issue. Although this interaction protocol is not designed to solve this problem, it could be used to select team components individually or by sending a special kind of multiple petition and then form the team. Selecting agents with complementary abilities to match up is not considered in this paper.
- Communications load:
  One of the biggest problems in distributed systems is communications overload due to the quantity of information that is required to be exchanged. This proposal does not come to solve this problem as the general usage of this protocol still requires a large number of messages sent in the system for an agent or a group of agents (the best suited) to be chosen for a task allocation. Given the conditions described above these messages are needed if the agent that centralizes, process all the information and makes the choice is eliminated. This work is repeated so many times as agents pretend to accomplish the task, as actually it is replicated by every one of these agents comparing themselves to the others.
- Availability (Dynamism):
  Sometimes, in competitive distributed systems, accomplish a given goal (petition) is essential to win (or loose). If the petition sender agent or any other central agent falls or is eliminated, then the allocated task will not be completed and the corresponding goal might never be reached. This is a typical situation in video games in which a defender agent can ask for support when the team base is attacked. If a Contract Net were used and the defender was killed in the interaction process, the supporter could never arrive and therefore the base would be left undefended. This is one of the main reasons this kind of coordination protocol is needed.
- Benevolence:
  In a cooperative environment, resolutions are made based on benevolence of the rest of the team agents. In this proposal agents' benevolence is supposed. If there is a common goal then benevolence is needed for team work. If agents lie about their capabilities or situation a wrong selection will be made. Several other architectures and protocol have the same problem, which can be treated with different approaches like trusting techniques, secure entries to the system, etc. These techniques are not discussed in this paper.

3. Some previous coordination protocols

There are several protocols dealing with the coordination problem. But must of them result useless or very hard to make work on some truly dynamic or risky environments. Some of the most relevant protocols are outlined in this section to show the differences with the proposed protocol.

3.1. RETSINA

RETSINA is a distributed multiagent system that uses a hierarchical organization divided in three layers. The first layer is constituted by interface agents, dealing with users. The second one is formed by task agents receiving petitions from interface agents, splitting them in subtasks and allocating them. Finally information agents receive petitions from task agents. Once data are retrieved by information agents it is sent back being processed until interface level.

Task agents use HTN (Hierarchical Task Networks) for their organization, tasks and actions scheduling. Different kind of plans libraries like general or domain specific are used for that.

This means that task agents' hierarchy will depend on plan libraries and it will be mainly static. If a few task agents stop functioning some functions of the information system will not be available. In a different environment like a competitive video game it could represent some basic capabilities of the whole team not being available and so a direct game lost.

Task allocation is made by *matchmaking* in which every agent that provides a service must be registered.

These *matchmaking* agents are supposed to do not fail either, because this would mean that the all the registered agents would not be available. And if agents are continuously going in and out of the system a lot of messages to register and unregister would be generated.
So this architecture still lacks of enough dynamicity to work reliably in a dynamic distributed and coordinated system like a video game.

3.2. GPGP

GPGP is very useful to schedule group tasks and coordinate agents even in a monitored and synchronized way [1]. Even with proposed enhancements to GPGP the quantity of communication messages can not be reduced. Commitment mechanism messages in a group are not reduced, still resulting in N^2 messages. And communications between different groups are made by leaders and a static leadership hierarchy. But this hierarchy may accumulate several timeouts if agents in hierarchy fail until message is sent from a group to another.

3.3. Contract Net

Contract Net is an extensively used interaction protocol. It has been adopted by FIPA [5]. But it can vary depending on implementation details. It consists mainly in a *contractor* initiating the protocol dialog sending a Call For Proposals to the rest of participants in the dialog. These participants send their proposals back to the initiator. The initiator accepts or rejects proposals depending on his interests. As can be seen all the selection process and information is concentrated in the initiator.
The risk here is not completing dialog if initiator is eliminated of the system.

4. BDI model of the proposed agents

In this section a Beliefs, Desires, Intentions (BDI) [8] model is described for agents to participate in the coordination protocol described in this paper. It is a partial model given and does not include specific beliefs, desires and intentions an agent may need to do their work. Only those relevant to the coordination protocol are described next:

Beliefs

- Petitions/Goals: The agent needs to keep a list with goals or petitions received. These goals can be modelled in several ways depending on the way the services offered in the system are modelled. Basic elements to include in the goal specifications can be the next:
  - Sender: The agent that publishes the goal.
  - Ontology/Ambit: A context must be specified in which petition has a determined meaning. It can be represented by ontology or a working area or ambit. Generally this ambit will let us classify agents according to their characteristics.
  - Action: The request itself specifies the concrete task to be accomplished. It will be interpreted depending on the previous ontology.
  - Quantity: It could be included in the previous attribute. Represents the exact number of agents needed or recommended to complete the petition. It is represented here because it is referenced in the protocol description.
  - Petition identification: Useful internally and in communications with other agents. Should allow a unique identification in the system. It could be based on the sender's system identification and a counter, supposing a unique identification for the agent.
  - Delays/Deadlines: Waiting times associated to the task actions, like minimum time waited before responding to the petition or deadline to be accomplished.
  - Restrictions: Generally related to work ambit, needed to discard agents that would not can to accomplish the task.
- Ranks: A global ranking is made in which every agent in the system knows his own position, respectively to every petition sent, indicating its adequateness to the petition requirements. So, every rank position must be associated to every petition in agent's beliefs list
- Characteristics/Capabilities: These represent the rest of beliefs the agent has. These does not need to match capabilities specification o services this agent offers, but a complete or partial vision of these characteristics will be interchanged among agents so that they can compare themselves and determine their position at global ranking.

Desires/Goals:

Agent's goals can be split in two kinds:

- Own goals: Agent will have his own goals so that he can work independently or autonomously. These goals are optional, making the agent act only on demand.
- Foreign o Group goals: Agent can receive petitions from other agents, creating new goals independently from owns to work in group or serve other agents.

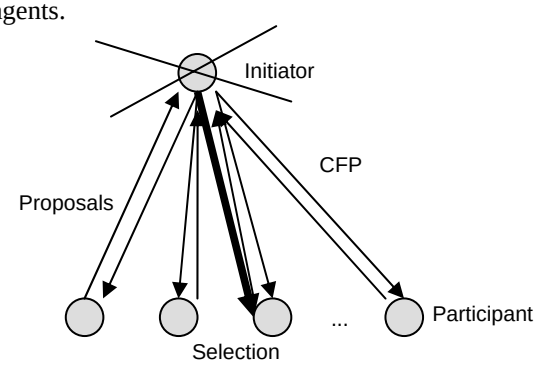
In case of not accepting external petitions, the agent is supposed to be autonomous and will not work in group. So it will not use this kind of coordination protocol and does not need to adopt this formalization, at least as service provider. It still could send petitions to the system and even receive responses in case that they are needed.

Intentions/Plans:
There are two plans directly related to the protocol as defined here:

- Accept Petition: This plan must accept incoming petitions to the agent, determine if these petitions correspond to agent capabilities, assign a global rank value corresponding to such petition (initially 1), check out specified restrictions and act after indicated deadline if agent is in the needed rank.
- Accept Capability/Situation Descriptions: In parallel with the previous plan. After accepting a petition, a message with agent situation is sent to the rest of agents. So messages with characteristics of the rest of agents will be received. This plan compares information included in every message received to own characteristics to determine agent position at global rank.

5. The proposed Coordination Protocol

This protocol can be considered basically as a Contract Net without a contractor role. Contractor role is reduced just to send the petition. Actually contractor role is distributed and replicated at the rest of agents.



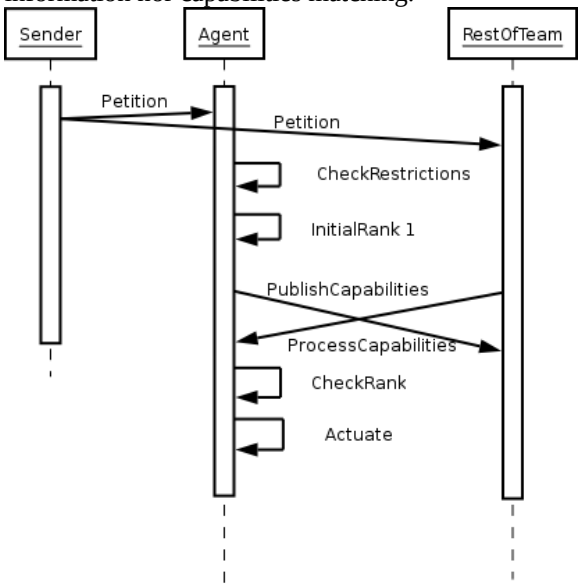
Noticing restrictions imposed to problem, the petition sender (contractor) can not assure to be there to process offers from the rest of agents. In fact a petition could be not sent by an agent itself but any other element in the system.
Once the petition is sent agents get into competition comparing their capabilities and situation among them to determine the optimum agent(s) to accomplish the solicited task in the given moment. In this competition a global ranking is created in which every agent will occupy a position depending on their characteristics or situation in that moment.
A detailed description of the interaction protocol is as follows:

- An agent o element in the system sends a message to the other agents (a specific group of agents, like agent's team or known neighbours) with the description of the task to be accomplished. This description formulation is already given above.
- Every agent that receives the message checks if all the necessary restrictions are fulfilled to respond to petition. Restrictions can be of two kind:
  - Own Restrictions: Those imposed by the agent itself to respond to the petition, like not being already working in another task, tasks incompatibilities, etc.
  - Imposed Restrictions: Those described in the petition, like time to respond, maximum distance to a given target, etc.
- Once the agent checks that all restrictions are fulfilled initializes its global ranking position associated to the petition to 1, as it is going to participate in the selection.
- Every agent publishes its capabilities, characteristics or situation at that moment, sending it to the rest of agents. This message generally has to be sent to the rest of similar agents as the original receivers list is neither known by this agent nor which of them fulfil restrictions and will respond to the petition.
- Every agent waits a given time, generally specified in the petition, as a delay before starting the required action. During this time messages specifying capabilities from other agents are processed. These characteristics are compared to own characteristics. This lets modify the global ranking position of the agent in the following way:
  - In case that message characteristics are higher than own value of global ranking is incrementing. This means that if initially ranking position is 1 it is changed to 2. This actually represents a global ranking position decrement.
  - In case of the other agent's characteristics being lower the message is just ignored and position at global ranking is not changed.
  - In case that characteristics or situations are the same or equivalent a *play-off* is needed. As an initial proposal an identifiers comparison is suggested, so that identifier will determine priority in tie cases. Other *play-off* methods are encouraged given that unique identifiers represent to have a

centralized access method to the system to ensure those identifiers are not repeated.

- Once specified delay time is over all messages with capabilities or situation at that moment from other agents must have been processed. So global rank belief must contain its final value. Then it is matched ranking position to the quantity of agents specified in the petition.
- If agent results to be among those with rank value lesser than or equal to that indicated in the petition, the specified goal at the petition is accepted by the agent as own, making the agent to start the corresponding task. In other case the agent has not been selected to do the task. Even then it could be useful to keep rank value associated to petition in beliefs list in case that other agents would not can to finish their task, so that next agents in ranking can be selected and start their work.

As can be seen, the optimum agents are selected for every petition at every moment. It is done without intermediaries, centralizing message interchange, information nor capabilities matching.



6. Analysis of the proposed protocol

Protocol described above represents a generic method applicable to all situations with restrictions imposed to the problem of finding optimum agent to accomplish a specific task in a given moment in a completely decentralized way.

Total number of interchanged messages is due to these strict restrictions and generalization, and so time spent since petition is sent until a response is given.

Total number of interchanged messages is $O(N^2)$, approximately $N+A*N$, being N the total number of agents and A the number of agents from N ($A \leq N$) that fulfil own and imposed restrictions. First N comes from petition sent from the sender and $A*N$ are characteristics messages sent by *competing* agents for being selected.

Total time spent since petition is sent until a response is given must be adjusted previously depending on the system implementation and specified in petition at least that a default value is used for all petitions. As in a Contract Net this delay time is needed so that all messages with characteristics information are sent and processed by every agent to determine ranking positions. If waiting time is not completed or messages are not completely processed agents will act according to not real rank values.

Agents should not act before having received and processed all incoming messages from other agents or waited enough time. This time does not necessarily has to be much greater than in a Contract Net.

This time is still much sorter than in synchronization mechanism proposed in [1] given that here there is no dependence of a leader and no timeouts are chained.

7. Possible optimizations of the proposed protocol

As stated above, protocol here described represents a general case with certain restrictions imposed. Depending on environment or system characteristics these restrictions can be relaxed and protocol modified or adapted adding enhancements to reduce the number of interchanged messages.

Here are described possible situations and modifications in the implementation of the protocol:

- Caching capabilities or characteristics of the rest of agents: Depending on how fast agent characteristics change along the time, messages sent by agents can include a time to life mark. These characteristics can be included as agent beliefs. In case of receiving a new petition it will not be necessary to resend new messages with agent situation as this is still stored, at least that agent situation has changed.
- Static Ranking: In case that situation varies slowly along the time a static ranking can be maintained. Every agent keeps referenced those preceding and next agents in the global ranking. If a new petition is received every agent only exchanges messages with those agents that are adjacent to the rank position. Once rank positions are adjusted, petition can be answered.

- As a modification to last one, the agent sending the petition can send only the message to one of the agents in the rank. This agent will compare its situation to adjacent ones and will resend the petition to those with better characteristics in that moment. When an agent does not find better agents, it selects itself to accomplish the task. This method actually does not assure to find the optimum agent as previous methods, but a local minimum or maximum.
- Other reduction in the number of messages is through the use of fuzzy logic. In this case agents can use fuzzy terms to label their characteristics or situation as *high*, *medium* or *low* level. Agents that consider having *low* level characteristics inhibit themselves from participating in ranking creation.

Even if agents' situation keeps quite stable along the time this method results useful to initiate a global optimum rank of agents depending on their capabilities. After that initiation phase, one of the simplifications described in this section can be used.

8. Simulation Tests of the proposed protocol

Different simulations have been created to test the protocol.

The first one simulates agents running across a room while petitions are sent to ask for a certain number of agents to run to a given position. Task is accomplished while enough agents are available to complete the specified number. This is especially useful for future implementations of Unreal Tournament© Bots (Agents) team work.

The second test simulates an area watched by surveillance video cameras while objects are circulating. Video cameras use this protocol to control objects in the area as much time as possible. They do it by sending a petition to the rest of cameras to watch an object that has gone out of their vision range. Objects' control is successfully passed from one camera to the next. It is done while the next camera is free or will not lose control over other objects. Simulation works successfully independently of mobile cameras or even failing cameras in the area with available resources at every moment [6].

9. Conclusions

Video games are a very common testbed of agent systems, but current coordination protocols allocate tasks assuming a relatively stable environment. But survival competitive games include conditions such as the sudden fall/die of agents involved in the game. So classic coordination protocols seem to be not directly applicable to this kind of video games.

Interaction protocol proposed here comes to fulfil this gap, letting petitions to be responded under difficult conditions by all available agents in that moment.

This proposal is made with the most difficult conditions and generalization in mind, resulting as a consequence of such strict requirements. If explicit implementations environments are not so strict, the protocol can be adapted as described in the previous section, reducing the communication load and even the delay to respond to petitions.

This interaction protocol has been implemented and test under different simulated ad-hoc experiments, working successfully. Next, we intend to test it on more real-time scenarios of video-games such us Unreal Tournament and other real-world coordination problem such as surveillance with camera.

References

[1] S. Abdallah, N. Darwish, O. Hegazy, *Monitoring and Synchronization for Teamwork in GPGP*. 2002

[2] W. Chen, K. Decker. *Managing Multi-Agent Coordination, Planning, and Schedling.* AAMAS'04

[3] K. Decker. "TAEMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms." *Funtadions of Distributed Artificial Intelligente, chapter 16.*1996. pp. 429-448.

[4] K. Decker, K. Sycara, M. Williamson, "Modeling Information Agents: Advertisements, Organizational Roles, and Dynamic Behavior," Working Notes of the (AAAI)-96 Workshop on Agent Modeling. 1996

[5] FIPA Contract Net Interaction Protocol Specification. http://www.fipa.org/. 2001

[6] J. García, J. Carbó, J. Molina. "Agent-based Coordination of Cameras." International Journal of Computer Science & Applications. Special Issue on Applications of Software Agents in Engineering. Vol. 2, No. 1 (January 2005).

[7] H. Muñoz-Avila, T. Ficher. *Strategic Planning for Unreal Tournament© Bots*. Proceedings of AAAI-04 Workshop on Challenges on Game AI. 2004

[8] A. Rao, M. Georgeff, *BDI Agents: from theory to practice*. In V. Lesser, editor, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95), pages 312–319. The MIT Press: Cambridge, MA, USA, 1995.

[9] K. Sycara, J. Giampapa, B. Langley, M. Paolucci. "The RETSINA MAS, a Case Study." 2003.

[10] K. Sycara, K. Decker, A. Pannu, M. Williamson, "Distributed Intelligent Agents," 1996

Context-Based Disaster Management Support

Alexander Smirnov, Michael Pashkin, Nikolai Chilov, Tatiana Levashova
St.Petersburg Institute for Informatics and Automation
of the Russian Academy of Sciences, St.Petersburg, Russia
{smir, michael, nick, oleg}@iias.spb.su

Abstract

The paper describes an approach to decision making support for disaster management. The approach is based on the methodology that assumes three levels of information integration. The application domain is described via an application ontology using the formalism of object-oriented constraint networks. The problem is described via an abstract context that is obtained as a result of the slicing operation on the application ontology. Finally, filling the abstract context with up-to-date information about the current situation produces in an operational context. Contexts of both types share the same knowledge representation formalism that is used by the application ontology. As a result the operational context can be considered as a constraint satisfaction problems. Solving this task produces feasible decisions in the current situation.

1. Introduction

The number of annual natural and human-made disasters continually increases. For the first five years of the decade (1994 to 1998), an average of 213 million people were affected. The second half of the decade (1999 to 2003) saw this figure rise by over 40 per cent to an average of 303 million per year [1]. The practice shows that one of the most difficult steps is getting the right relief supplies to the people in need at the right time. At the same time delivering of too much supplies or wrong supplies means loosing time and money. Therefore, humanitarian logistics standing for *processes and systems involved in mobilizing people, resources, skills and knowledge to help vulnerable people affected by natural disasters and complex emergencies*, is central for disaster relief [2]. This fact motivated the choice of the case study for implementation of the presented here approach.

Very often, local organizations involved in emergency response do not have the resources to respond effectively to a disaster. It is therefore important to determine what resources an organization has (or is lacking), and what is required for relief operations to be carried out effectively. Given actualized information available for logistical planning and preparations, this will make it easier to determine which resources are available – and which are lacking and must be produced elsewhere.

Such operations take place in rapidly changing content of network-centric environment. Increasing complexity of decision making and wide acceptance of information technologies, computational intelligence is currently highly demanded in the area of coalition operations.

Coalition operations include but not limited to: emergency preparedness and response (to terrorism attacks / incidents, catastrophic events, natural disasters, emergency situations, etc.); global war on terrorism and Multinational operations other than war, etc. To manage any coalition operation an efficient knowledge sharing between multiple participating parties is required [3]. This knowledge must be pertinent, clear, and correct, and it must be timely processed and delivered to appropriate locations, so that it could provide for situation awareness. This is even more important when coalition operation involves coalitions uniting resources of both government (military, security service, community service, etc) and non-government organizations.

Operations exploit information and network technologies to integrate widely dispersed human *decision-makers*, networking *sensors*, and *resources* into a highly adaptive, comprehensive *network-centric environment* to achieve shared *situation* awareness and *unprecedented mission* effectiveness by efficient linking *knowledgeable components* in the environment

The rest of the paper is organized as follows. The methodology proposed is described in Section 2. It is based on usage of ontologies and contexts of two types: abstract and operations. These constituents of the methodology are described in Sections 3-5. Section

6 describes a case study to be used for future experimenting. Some findings and results are summarized in the conclusion.

2. Proposed Methodology

The methodology presented proposes integration of environmental information and domain knowledge in context through linkage of representation for this knowledge with semantic models for environmental information sources providing information about the environment.

The methodology (Fig. 1) considers context as a problem model based on the knowledge extracted from the application domain and formalized within an application ontology by a set of constraints. The set of constraints additionally to the constraints describing domain knowledge includes information about the environment and various restrictions of the user on problem solving. Within a coalition the restrictions of the user include different user roles. The methodology takes into account the different user roles as different levels of user responsibility.

The problem is suggested being modeled by two types of contexts: abstract and operational. *Abstract context* (Fig. 2, left) is an ontology-based model integrating information and knowledge relevant to the problem. *Operational context* (Fig. 2, right) is an instantiation of the abstract context with data provided by the information sources. In Fig. 2 it can be seen that attributes "x-coordinate", "y-coordinate" and "cost" are assigned values 246, 310 and 1000 respectively.

3. Application ontology

Ontology library is internal knowledge storage. It stores ontologies imported from distributed heterogeneous knowledge sources. The ontologies are formalized in a uniform way. They are described by means of the internal ontology formalism and the vocabulary supported by the ontology library.

References to the knowledge sources the ontology have been imported from are organized in a knowledge map. Besides the references, the knowledge map contains knowledge sources metadata, and information about their accessibility, location, native format, and other properties.

Domain knowledge is modeled by ontologies of three types: domain ontology, tasks & methods ontology, and application ontology. Domain ontology represents conceptual knowledge about the domain, tasks & methods ontology formalizes tasks identified for the domain and hierarchies of problem solving methods (taking into account alternative ones). The tasks and methods are represented by classes; the sets of methods' arguments and argument's types are represented by sets of attributes and domains, respectively. Domain and tasks & methods ontologies are interrelated by relationships specifying values of which class attributes of a domain ontology serve as input arguments for the methods of a task & methods ontology. Application ontology is a specialization of domain and tasks & methods ontologies. Knowledge from domain and tasks & methods ontologies is integrated into application ontology that describes a real-world application domain depending on particular domain and problem [4]. Ontologies of these types are stored in an ontology library.

Decision making deals with complex problems expecting deep knowledge in the domain. The users do not necessarily have satisfactory knowledge. This fact is the most important at the operational level when the user has to make decisions under time pressure. Because of this, the approach relies on an availability of sufficient domain knowledge and support of subject experts, if required. The domain knowledge is collected before it can be used in decision making.

The phase of domain knowledge accumulation consists in importing knowledge relating to the domain in question from Internet resources, representation of

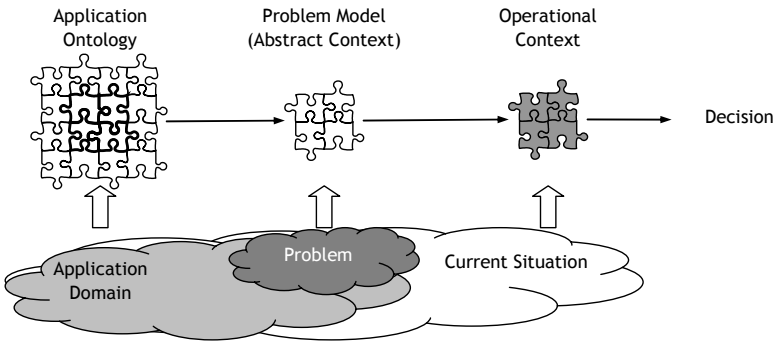


Figure 1. Context-based decision support

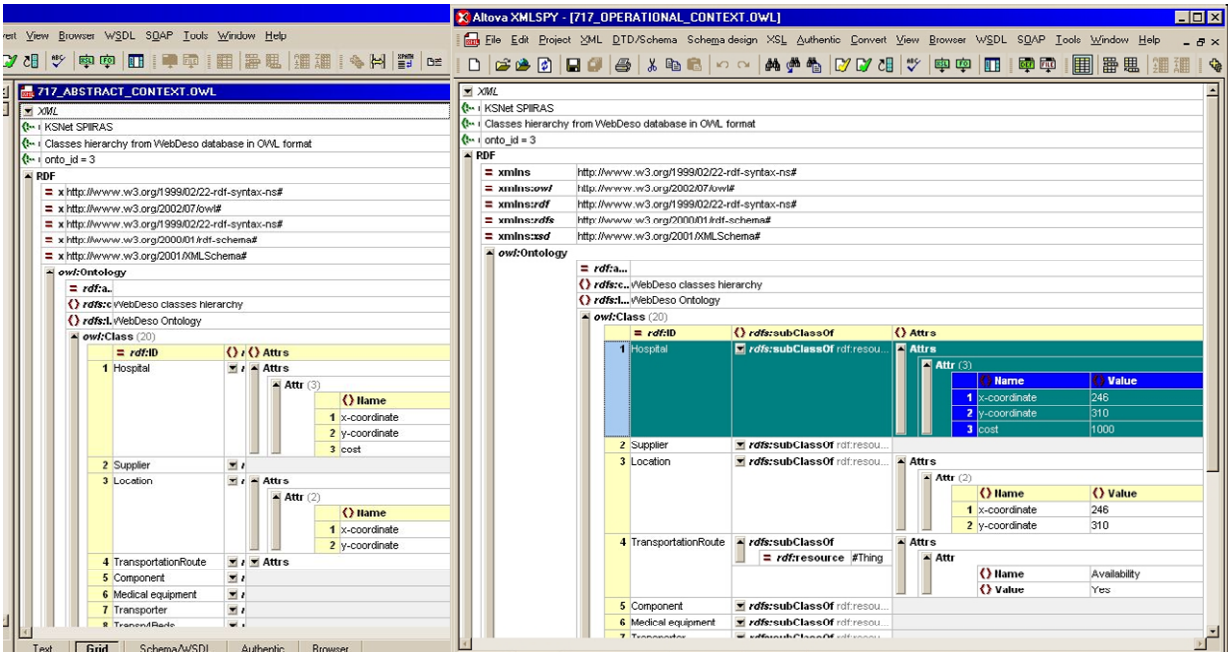


Figure 2. Abstract context (left), and operational context (right) stored as XML files

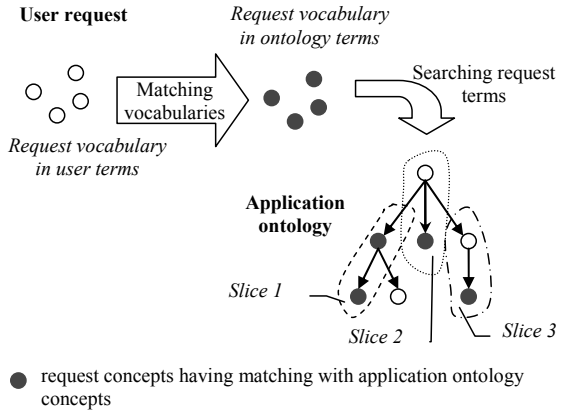


Figure 3. Application ontology slicing

the imported knowledge by the formalism of object oriented constraint networks described in detail in [5], and saving this knowledge in the ontology library.

Since the user vocabulary (the request vocabulary) and the ontology library vocabulary can be different, these vocabularies are matched. Then concepts of the request having matches in the vocabulary of the ontology library are searched for in the application ontologies. The terms found serve as "seeds" for the slicing operation [6], [7], [8]. The purpose of this operation is to extract pieces of knowledge from the application ontologies, that is believed to be relevant to the request, and consequently to the problem to be solved (Fig. 3). The operation assembles knowledge related to the "seeds" based on attributes and constraints inheritance rules. The result of the

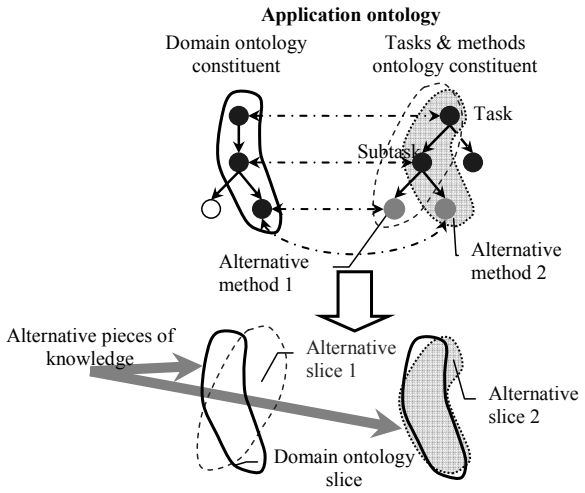


Figure 4. Alternative slices

operation is a set of ontology slices containing pieces of knowledge that surround "seeds". Different slices that combine knowledge representing alternative methods are considered as alternative (Fig. 4).

The slices are merged so that alternative slices would become parts of different pieces of knowledge (Fig. 4). The resulting pieces of knowledge will make up alternative problem models. The result of the integration is a single resulting slice if slicing algorithm has not revealed any alternative slices, or a set of resulting slices where each resulting slice is purposed to describe an alternative problem model. The resulting slice (a set of slices) checked for

consistency is considered ontology-based problem model. Alternative slices constitute alternative problem models.

4. Abstract context

The starting point for the decision making level is the user request containing the formulation of the problem to be solved. Based on the result of the request recognition, knowledge relevant to it is searched for within and extracted from the application ontologies of the ontology library. This knowledge is integrated into *abstract context*. The abstract context is an ontology-based problem model supplied with links to representations of the information sources that will provide values for the class attributes included in the abstract context. The attributes represent both attributes of domain ontology classes and arguments of methods that come from the tasks & methods ontologies. Referring to the constraint satisfaction problem (CSP) the attributes correspond to variables of this model. An example of the abstract context can be seen in Fig. 5. Rectangles denote classes with attributes, solid lines denote associative relationships. Part (a) illustrates abstract context for "Resource Allocation" subproblem (b) illustrates abstract context for "Hospital Allocation" subproblem, and part (c) illustrates abstract context for "Routing" subproblem.

Due to links between ontologies and information sources, the integrated knowledge is connected to those information sources and users that are supposed to provide data values for problem variables. Information source representations that represent these data values are sliced. For this, a slice of an information source of a complex data model is formed limited to the model elements representing information needed for the problem. If an information source is of a simple data model the slice is the representation of this information source. This issue is described in detail in [9].

5. Operational context

The information sources providing data values needed for the given problem instantiate the abstract context. The instantiated abstract context is *operational context* that is the problem model along with problem data and object-oriented constraint network to be processed as a CSP. Changes in the environment result in changes in the operational context.

The operational context is presented to the user. The user makes decisions based on this context if it is a current situation description or based on a set of feasible solutions generated by the constraint solver if the context is a problem definition.

In order to enable capturing, monitoring, and analysis of decisions and their effects the contexts representing problem models and respective decisions made are retained in an archive. As a result the user is provided with reusable problem models and knowledge of similar situations and decisions made in those situations.

The information sources instantiate the abstract context through resizing of variable domains. The abstract context with fully or partially resized domains is operational context. An example of the operational context is given in Fig. 6.

A constraint solver based on the operational context generates a set of feasible solutions for the problem modeled. This set is presented to the user. The user estimates these solutions and chooses desirable one that is the decision. In order to support evolution of knowledge included in the contexts, allow the user to

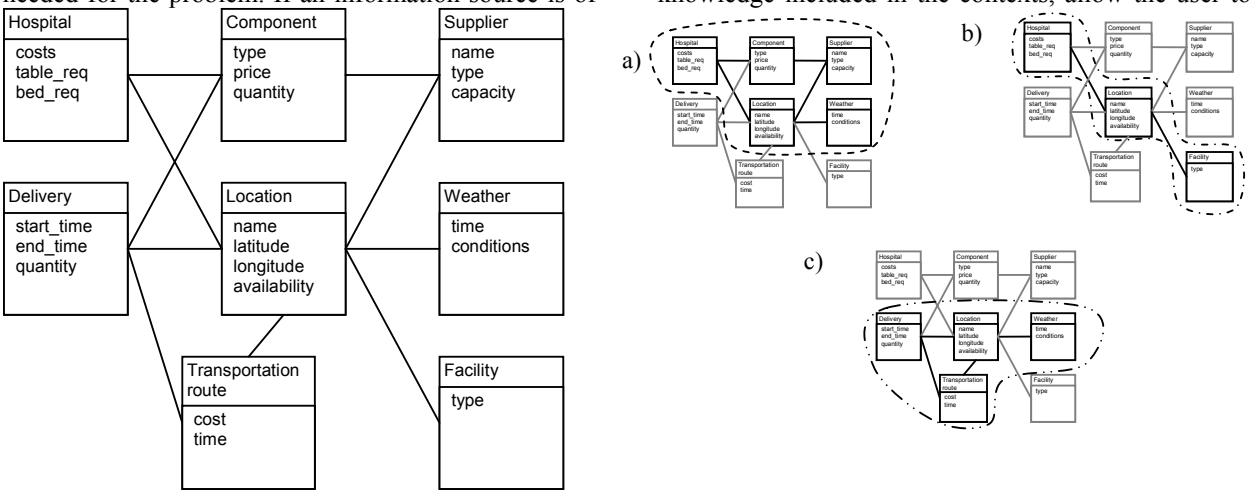


Figure 5. Examples of abstract contexts

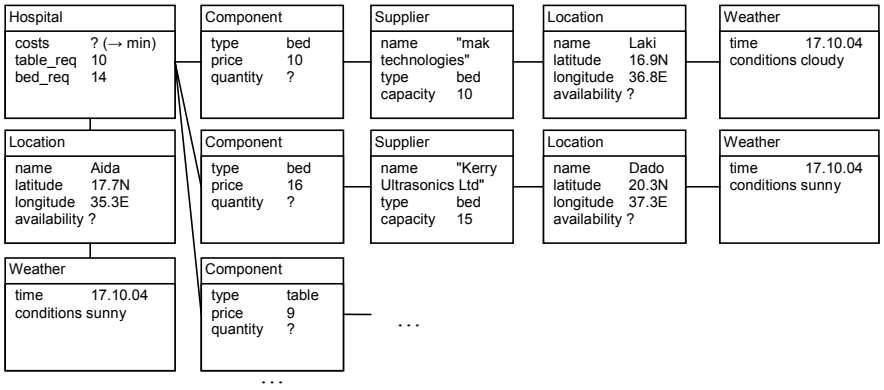


Figure 6. An example of operational context for the "Resource allocation" subproblem

access reusable problem models, and provide the user with knowledge of similar situations and decisions made within the contexts the abstract context, operational context, a set of the generated solutions, and the decision are saved in the archive. The support purposes are achieved applying techniques of context versioning and profiling.

The operational context in its form of object-oriented constraint network is supposed to be processed by a constraint solver as a CSP. The user makes decision based on alternatives generated by the solver.

6. Extended case study

The described approach has been implemented in a case study of portable hospital configuration. Its detailed description can be found in [5]. Currently, it is planned to extend the case study. This section describes the extended case study to be used for further research.

6.1. Disaster

The problem considered is based on a simulated natural large-scale disaster. For experimenting different types of disasters will be considered. They include earthquake, flooding, fire, etc. Usually, disaster type defines common injuries of people affected and main relief measures to be undertaken. Different types of disaster will make it possible to simulate similar scenarios with different parameters.

For example, burns will be most common for fires. Burn facilities and firefighter teams will be required. However, in case of earthquake facility profiles will be different, and there will be needs for humanitarian aid, rescue teams and construction workers. The parameters are to be defined based on available information sources.

It is planned to use different locations with different features (cities, transportation routes, landscape types) for the case study. It will make it possible to compare solutions for different territories. The territory information is to be obtained from public sources. Processing of this information is planned to be done via a GIS (geographical information system).

6.2. Tasks to solve

The problem is divided into two main subproblems:

- Relief – defining and getting right supplies and workforce to the place of disaster
- Evacuation – evacuating people affected from the location of the disaster

The first subproblem is further subdivided into the three tasks.

The first task is defining right supplies and their quantity in accordance with the context of disaster type, scale and location. The supplies may include: medical supplies, humanitarian aid, hospital assemblages for estimated injury types and patient quantity, etc. The task can be defined as a table function or a set of rules. The inputs are disaster type and estimated number of victims; the output is a set of supplies / supplies types and their required quantities.

The second task is defining suppliers who can provide for the required supplies. Solving this task should take into account their capacities and capabilities as well as locations. Initial information about the suppliers can be obtained from public sources or fictitious suppliers will be introduced. This task is a configuration or resource allocation task. The goal is to find a feasible suboptimal solution. The inputs are required supplies and available suppliers and their parameters; the output is a set of rules defining amounts of supplies to be acquired from each supplier.

The third task is solved jointly for the both subproblems: defining routing plan for delivering

supplies and evacuating people. This is a logistics task. It can be treated as an extended routing or transportation task. Its solving should take into account current conditions in the region (e.g., flooded roads, etc.), available transportation means (ground, air, etc.) and existing infrastructure (airports, roads, etc.). The inputs are the results of the second task and the above parameters; the output is a routing plan. Information about the current conditions will be acquired from sensors and other similar information sources. Information about transportation means will be obtained from public information sources. Existing transportation infrastructure will be taken from the GIS.

Conclusion

The paper presents an approach to decision making support for disaster management. The presented approach has a number of potential advantages for the operational decision making: (1) contexts contain information relevant to a particular task or situation, that allows selecting source types responsible for observation constraints relevant to the area of interests; (2) ontologies make possible to transform information provided by sources into knowledge at the level of description of the area of interests therefore an ontology-driven context at the decision making level provides the decision maker with the knowledge; (3) context management technique enables generation of alternative contexts representing alternative situations or alternative ways of problem solving; (4) knowledge representation via object-oriented constraint networks allows working with the operational context as if it is a CSP and generate feasible solutions using a constraint solver.

Acknowledgements

The presented research was partially supported through the CRDF partner project # RUM2-1554-ST-05 with US ONR and US AFRL, project # 16.2.35 of the research program "Mathematical Modelling and Intelligent Systems", the project # 1.9 of the research program “Fundamental Basics of Information Technologies and Computer Systems” of the Russian Academy of Sciences, and the project funded by grant # 05-01-00151-a of the Russian Foundation for Basic Research.

References

[1] *Humanitarian Logistics: Getting the Right Relief to the Right People at the Right Time*, Fact Sheets, Fritz Institute, 2005 URL: http://www.fritzinstitute.org/fact_sheets/f_s-hls.html.

[2] Scott P, Rogova G: Crisis management in a Data Fusion Synthetic Task Environment. *Proceedings of the 7th Conference on Multisource Information Fusion* (*Fusion 2004*), 2004: 330-337.

[3] [Pechoucek] Pechoucek m., Rehak M., Rollo M., Sislak D., Tozicka J. Solving Coordination Inaccessibility in Coalition Operations. In *Knowledge Systems for Coalition Operations* (Pechoucek M., Tate A., eds.), ISBN 80-01-03065-2, 99 – 114, 2004.

[4] N. Guarino, “Formal Ontology and Information Systems”, *Proceedings of FOIS'98*. Trento, Italy. Amsterdam, IOS Press, 1998, pp. 3—15.

[5] Smirnov A., Pashkin M., Chilov N., Levashova T., Krizhanovsky A. Ontology-Driven KSNet-Approach to Coalition Health Service Logistics Support. *Knowledge Systems for Coalition Operations* / Eds. by M. Pechoucek, A. Tate. – Prague: Czech Technical University, 2004, pp. 99—114.

[6] V.K. Chaudhri, J.D. Lowrance, M.E. Stickel, J.F. Thomere, R.J. Wadlinger, *Ontology* Construction *Toolkit*. Technical Note Ontology, AI Center. Report, 2000. SRI Project No. 1633.

[7] T.V. Levashova, M.P. Pashkin, N.G. Shilov, A.V. Smirnov, “Ontology Management”, in *Journal of Computer and System Sciences International*, part II, vol. 42, no. 5, 2003, pp. 744--756.

[8] B. Swartout, R. Patil, K. Knight, T. Russ, “Toward Distributed Use of Large-Scale Ontologies”, *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96)*, Banff, Canada, 1996, URL: http://www.isi.edu/isd/banff_paper/Banff_final_web/-Banff_96_final_2.html.

[9] Smirnov, A., Pashkin, M., Chilov, N., Levashova, T., Krizhanovsky, A. Ontology-Driven Information Integration to Operational Decision Support. *Proceedings of the 8th International Conference on Information Fusion (IF 2005)*, Philadelphia, USA, July 25—29, 2005. IEEE Catalog Number 05EX1120C, ISBN: 0-7803-9287-6, IEEE, 2005.

Improving Agent Coalitions by Behavioral Patterns Clustering and Conservative Reconfiguration

Krzysztof Ciesielski
Institute of Computer Science, Polish Academy of Sciences
Ordona 21, 01-237, Warszawa, Poland
K.Ciesielski@ipipan.waw.pl

Abstract

In this paper we present a novel application of incremental data mining algorithms to the problem of formation and reconfiguration of coalitions of agents cooperating in dynamically evolving environment. Our experimental generator of coalitional structures takes into account both the stability of resulting coalitions and efficiency of computations. It focuses on providing average-case optimal solution and generates coherent stable groups with respect to agents beliefs, intentions, capabilities as well as the current environmental state. Incremental clustering leads to a robust adaptation of existing structure in response to rapidly changing environmental conditions, even in case of complex, high-dimensional models.

1. Introduction

A coalition formation process, in which a number of independent, autonomous agents come together to act as a collective, is an important form of interaction in multiagent systems. Cooperation in an effective coalitional structure can improve the performance of the individual agents and the system as a whole can reach goals which are beyond individual capabilities. There are numerous real-life applications in which efficient formation and re-formation of coalitional structures in dynamically changing environment is needed (to mention only e-business or grid computing). However, the main problem is the computational complexity of coalitional structure generation. The problem of partitioning a group of agents in order to maximize the social payoff has been shown to be NP-hard [14] and even finding a suboptimal solution (that can establish a worst-case bound from the optimal) requires searching the exponential number of solutions [13, 15, 10]. Our approach focuses rather on providing efficient solution for coalitional structure generation which is average-case optimal and generates coherent stable groups with respect to agents beliefs, intentions, capabilities and the current environmental state [3]. Another important issue here is to provide methods for a fast and conservative reconfiguration of existing structure in case of collective task execution failure [4]. Conservativeness means to change an existing structure as little as possible with respect to context-dependent similarity measure between teams and agents.

2. Adaptive management of coalitions

The main goal of this project is to apply clustering methods to discover relationships and patterns of agents beliefs, intentions and capabilities and to find coalitional structure which is close to optimal (at the present state). This structure is computed by adapted BIRCH algorithm [17], supplemented by heuristics establishing optimal algorithm parameters values. From the algorithmic point of view, coalition formation is construction of CF-tree of agents characteristics (represented as vectors of attributes). Next, a global grouping of CF-tree leaves is performed and coalitions are assigned to active tasks in a way that minimizes distance measure between coalition capabilities (as a whole) and given task specification.

The latter problem (so called *optimal assignment problem*) is solved by randomized greedy algorithm, but it is planned to adapt different methods, especially required when we permit agents to participate in more then one coalition at the same time *(fuzzy coalitions* [11]). Informational (*beliefs*) as well as motivational (*intentions*) factors are taken into account during coalition formation to assure cohesion of coalition behavior as a whole. During task assignment phase only capabilities of individual coalitions (which are product of capabilities of agents constituting given coalition) are considered.

BIRCH is an incremental algorithm, which means that it is possible to add or remove agents to existing coalitional structure and dynamically modify it, if needed. Having the coalitional structure built, we take advantage of some specific features of PAM and Clarans clustering algorithms [12] to conservatively rebuild it in response to dynamically changing internal and external conditions.

2.1. Global clustering

Global methods require access to the whole information about clustered objects during processing, which means they are not incremental and not directly applicable in multi-agent context. We would exploit some of their features, as discussed later, but themselves they are not suitable for grouping (and re-grouping) dynamic sets of agents.

2.1.1 K-MEANS. K-MEANS is a *direct* technique: the desired number of clusters K has to be given as an input parameter. Cluster is represented by geometrical center of gravity of object in a given subset (so called centroid). In the sequel we will treat terms *coalition* and *cluster* as synonyms and use it interchangeably.

First, K random agents $\overrightarrow{X_j}, j = 1, 2, ..., K$ operating in the MAS are selected and set as initial *centroids*. Next, every other agent is assigned to the cluster represented by the nearest of K centroids. Finally, new centroids are computed as $\overrightarrow{X_j} = \sum_{X \in C_j} \frac{X}{|C_j|}$ and error measure of current partition is calculated: $E = \sum_{j=1}^{K} \sum_{X \in C_j} \|X - X_j\|^2$. All steps are repeated until the change of error value E is insignificant or the cluster structure is not changing anymore.

The main drawback of K-MEANS is the complexity of a single iteration, which is in order of $O(N^2)$.

2.1.2 PAM. K-MEANS ignores information calculated in single iteration step and repeats clustering process almost from the beginning, ignoring fact that the cluster membership in a single iteration changes insignificantly. Next three algorithms are free of this drawback. Assuming that one has initial (random) partition of N agents into K coalitions (represented by cluster centroids), it is modified by moving single agents between clusters to improve coalitions quality.

If we treat search space as a graph where each node is a K-element subset of the agent set (possible configurations of coalition groups partitions) and edges connects configurations which differ exactly one element (i.e. one agent changes group allocation) then PAM is *hill-climbing search* algorithm in such graph.

PAM algorithm [12] works as follows: K random agents are selected and assigned to one of the initial *medoids*. Next, the cost function of the current configuration is calculated: $TC_{ih} = \sum_j C_{jih}$ for every pair of agents such that A_i is one of medoids and A_h is not. The pair $\langle A_i, A_h \rangle$ that minimizes value of the cost function TC_{ih} is selected. If the minimal value is negative, replace medoid A_i with A_h and return to the cost function recalculation step. If no cost function lowering is possible then stop and assign agents to one of the previously found medoids.

Main drawbacks of PAM algorithm are: (a) the complexity of search space (PAM works well for moderate size systems, ca. 100 agents divided into 5 groups), (b) very high graph branching factor (second step of algorithm have to dispose of $K \cdot (N - K)$ pairs of objects, so the complexity of single iteration is $O(K \cdot (N - K)^2)$) and (c) convergence only to local minimum of cost function, which in general case can be arbitrary distant from optimal partition.

2.1.3 CLARA. CLARA [12] is a modification of PAM, which tries to overcome first of PAM drawbacks. It randomly selects subgroup of agents (sub-graph) and runs PAM on such sample. Repeating this procedure several times we expect to find minimum. Resulting configuration is the one which minimizes the cost over the original agent set (i.e. in original space of agents attribute vectors).

CLARA is a trade-off: it significantly reduces the complexity [to $O(n)$], but it is also unlikely that it will find real optimum in original search space. Another difficulty is a choice of proper sample size. Authors suggest values proportional to the number of expected clusters (i.e. K).

2.1.4 CLARANS. Basic idea of CLARANS is to combine the best features of PAM (finding the true minimum at the expense of searching in the full space) and CLARA (being fast while searching only randomly sampled subspaces). Idea of CLARANS is to search only *dynamically* chosen subset of neighbors starting from a given configuration (a set of potential medoids). It has two input parameters. $maxneighbor$ is the number of randomly chosen neighbors in a single iteration. The bigger this value is the more similar CLARANS is to original PAM approach. Second parameter is the number of algorithm iterations $numlocal$. The lower $maxneighbor$ value is, the higher should be $numlocal$ value, in order to increase the probability of finding the global minimum.

In each step it considers $maxneighbor$ random neighbors of *current* node and evaluate the reduction of cost function value that would be obtained by moving to configuration represented by these nodes. If moving to any of them would reduce the cost then the best node found becomes the current node. This process is repeated until no further coalitional structure improvement can be made. In such case it is restarted ($numlocal$ times) from randomly chosen node.

We claim that CLARANS algorithm can be reinforced by introducing some kind of system's memory (in a manner similar to evolutionary strategies approach) and switching between heuristically chosen node from a memory in case of encountering poorly evaluated path. It can also be computed in parallel, making use of distributed environment.

2.1.5 Conclusions. Although all the above described algorithms stand significant progress comparing with original K-MEANS approach, they still have many disadvantages. Among most important we should mention: high sensibility to initial partition (configuration) and order in which agents data is processed, convergence to a local minimum (at best, vide Clarans), exponential complexity of the solutions space and the lack of incrementality (appearance or disappearance of an agent requires recalculation of the whole clustering structure). Moreover, all agents are treated equally (in particular, above mentioned algorithms don't cope with outliers). Last but not least, all four methods are are global, which means that every decision requires complete information and makes such approach impractical in MAS context.

In problem of searching for optimal agents coalitional structure and reconfiguring this structure conservatively (with respect to the appropriate distance or similarity mea-

sure, guiding local changes), most troubling drawbacks are the last three. Hierarchical approach presented below and based on BIRCH algorithm is free of these drawbacks.

2.2. Hierarchical clustering

BIRCH [17] (Balanced Iterative Reducing & Clustering using Hierarchies) algorithm has many features, which we consider encouraging from the MAS perspective. It has the ability to operate on huge agents characteristics data (numerous and high-dimensional attribute vectors) and also to operate on summary of agents data, not only on single object. It is optimized in terms of I/O operations cost [$O(n)$] since it exploits data structures similar to widely-known B-trees. It has low sensibility to *unfavorable (skewed)* order of agents characteristics processing. Contrary to previously described global methods, BIRCH is incremental and allows to include or exclude agent and then modify existing partition when it is required.

In our research, hierarchical approach has an additional advantage, since we plan to integrate it with hierarchical contract net, which explicitly represents hierarchy of task bidders and their subcontractor groups.

2.2.1 Clustering-Feature trees. The conception of CF-tree is based on widely used B-trees, which is balanced search-tree minimizing the cost of I/O operations [search, insertion and deletion of an object (agent) requires $O(h(T)) = O(log_t(N))$ disk access operations and $O(T \cdot h(T)) = O(T \cdot log_t(N))$ processor time; T - node capacity, N - tree size (the number of agents)]. A single inner CF-node consists of B-tuple in the form of B $[CFentry(i), childnode(i)]$ entries. The whole CF-tree defines hierarchy of clusters, which, in turn, defines hierarchy of agents coalitions.

CF-entry is a triplet $\left\langle N, \overrightarrow{LS}, \overrightarrow{SS} \right\rangle$, where N is the number of object in cluster, LS - vector sum of these objects (represented as points in Euclidean space), SS - square sum of these points. It is straightforward conclusion, that all basic clustering measures can be effectively computed solely on the basis of CF-entries representing individual clusters. Moreover, it can be shown that if CF1 and CF2 are CF-entries representing two disjoint clusters, then CF1+CF2 represents merged cluster.

CF-tree is parameterized by two parameters: branching factor B (defining maximum size of CF-entry) and threshold value T, which defines maximum cluster diameter. If cluster diameter exceeds T, then it has to be splitted. It can be easily shown that if $T1 \leq T2$ are threshold values of trees C1 and C2 respectively (having the same branching factor B) then $height(T1) \geq height(T2)$. The higher threshold value is, the smaller input data have to be clustered and the less computation is required. On the other hand, lower threshold value leads to higher clustering accuracy.

2.2.2 BIRCH. BIRCH algorithm starts with CF-tree build phase: objects (representing agents) are inserted one by one into the tree, descending on path from root to leaf. On each level choose entry most similar (with respect to a certain similarity or distance measure) to a given object, in manner similar to B-tree insertion algorithm [17]. One should note that further computations operate solely on summary of agents data instead of original information, so we can expect smaller impact of agents insertion order on clusters structure. It also facilitates identification of possible outliers during coalition formation.

Second phase is the optional CF-tree condensation: shrink CF-tree (via CF-tree rebuild algorithm [17], which increases threshold value T and then successively, one by one, in dictionary order, copies all paths to a newly created tree) to a size, which is adequate for global clustering algorithm, applied in third stage (e.g. Clarans - 5000 objects).

In the final phase, global clustering algorithm is applied to all leaf entries. Complexity of this step is in the order of $O(m^2)$, where m is the number of CF-entries.

2.3. Coalitional structure reconfiguration

Our application is designed to work as a part of multi-agent system (in particular real-time MAS), incorporating all above described methods of efficient coalitional structure reconfiguration into a mechanism coordinating agents cooperation. A suitable solution is contract net [16, 7].

2.3.1 Cooperation via contract nets. Contract net is a mechanism utilizing principles of real-world auctions and tenders market. Communication between the client (manager) and the suppliers (bidders) consists of interleaved phases of requests for bids, proposals submissions and evaluations of proposals. Cooperation between manager and agent (or a coalition) is based on bilateral agreement, which makes possible to take account of a large number of parameters, such as agents capabilities, its current workload, the type of task to be carried out, the description of operations, the type of data and resources to be supplied, expected task execution time and cost etc.

Initiating agents, working inside contract net module, exploit their beliefs about the current state of environment as well as up-to-date models of agents' beliefs, intentions and capabilities, and collects alternative solutions of suboptimal coalitional structure. Next, optimal coalition is created. From this moment on a coalition is called a team, glued together by a *collective intention* towards a common task ([5]). Then, agents collectively commit themselves to solve presented tasks. The *collective commitment*, resulting from a rather complex process (see [3, 6] for details) is the strongest motivational attitude within a team. This notion pragmatically reflects the way in which a social plan leading to the overall task is to be executed.

Contact net module copes with multiple, concurrent reconfiguration demands and allows to run many parallel grouping processes with different parameters and on different subsets (or subspaces) of agents characteristics. It is naturally designed for a distributed architecture so there is no need to create any additional, separate control mechanism. Taking it into consideration, contract net seem to be adequate control mechanism to complement it with our own approach to problem of creation and reconfiguration of

optimal (with respect to agents cooperation) coalitions of agents. Since we consider generalization of the presented approach to tasks decomposition, allocation and reallocation issues, all the most important cooperation coordination tasks could be included in one MAS control module.

2.3.2 Coalitional protocol. State of environment, agent models and the dynamics of environmental changes in our experimental application are all described by scripts defining so-called coalitional protocol. Complete protocol specification can be found in [1]. Here we describe only the most important RECONFIGURE command, triggering actual reconfiguration. in the current state.

Current state is defined by the execution of a sequence of commands and their combined influence on the initial environment state. From the algorithmic point of view reconfiguration is a construction (and further modification) of clustering model representing subgroups of agents. Next, coalitions are assigned to active (pending) tasks in a way that minimizes distance measure (commonly, weighted cosine angle) between coalition capabilities (as a whole) and given task parameters. The latter problem (so called *optimal assignment problem*) is solved by randomized greedy algorithm, which we won't discuss here.

The whole procedure starts, when coordination module (contract net) passes on coalitional script to the agents that are capable of recognizing of potential for cooperation [4] and will initiate coalition formation procedure. Each of the initiators verifies script syntactic and semantic correctness and prepare (or update, in case of reconfiguration) data structures required by clustering process (maps of tasks descriptions and agents models); these structures are prepared on the basis of initialization section of the script. Next, clustering parameters are determined (by user or by execution of heuristic) on the basis of current reconfiguration requirements (e.g. resource and time limitations) and the executional section of the script (modelling dynamic changes in environmental state) is parsed and corresponding grouping or regrouping processes are triggered. If the coalition formation procedure succeeded, initiator returns optimal coalitional structure to the coordination module (contract-net); at this stage all necessary collective commitments [3] should already be established.

As the final remark, it should be stressed that not every agent in a given situation is interested (with respect to his desires or intentions) or should be taken into consideration (with respect to his capabilities) to be a member of any coalition [3]. These agents should be identified as outliers and excluded from final structure. Not taking care of outliers and forcing agents which are not capable or not willing to cooperate (under present environmental conditions) within any coalition can negatively impact stability of resulting coalitional structure. Identification and management of such cases is important and not a trivial problem.

3. Experimental results

Experiments have been divided into separate groups to examine quality and efficiency of coalitional structure reconfiguration as well as the conservative adaptation properties. In the following sections we focus primarily on the adaptive aspects of reconfiguration. Results of other qualitative and performance experiments can be found in [1].

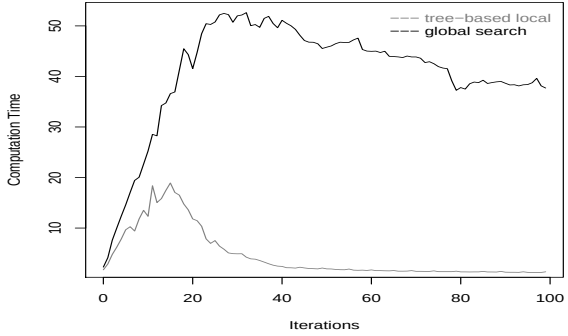


Figure 1. The complexity of the global search versus CF-tree based reconfiguration.

3.1. Qualitative tests

The aim of qualitative tests was to determine whether system behaves rationally (i.e. consistently with theory-based intuitions). As a test data we created small groups of agents (10-20) and several tasks specification, such that capabilities of each agent in unambiguous (but different for individual agents) way corresponded to the specification of one of the task. The purpose was to assert whether at the beginning of coalitional protocol execution agents will form expected initial coalitional structure.

Subsequently, multiple variants of coalitional protocol were executed, varying in dynamic change of environmental conditions (i.e. tasks settings) and agents parameters (i.e. their motivational attitudes, as well as capabilities). As we assumed, reconfigurations of coalitional structure were executed in conservative manner and at the same time it tended to find solutions near optimal under present environmental conditions.

Important observation was the ability to exploit specific feature of PAM and Clarans algorithms, i.e. transition of configuration graph edges connecting *similar* coalitional structures, to model situations in which neither environmental conditions nor agents parameters have not changed, but nevertheless collective task realization has failed. This means a conservative reconfiguration in face of objective failure of collective task execution [4].

Test were also diversified with respect to mutual importance levels (priorities) of informational factors (agent beliefs), motivational factors (agent goals) and agent capabilities. The aim was to verify our hypothesis that increasing (during coalition formation step) role of agent beliefs and desires at the cost of decreasing importance of their capabilities can be treated as an analogy and model for different possible mutual commitment strategies (*blindly-committed*, *single-minded* and *open-minded* [3]).

As expected, decreasing importance of the fact that agents capabilities unambiguously correspond to demands reported by environment (i.e. appearance, disappearance or modification of tasks to be executed) and, at the same

time, increasing role of their individual beliefs and desires caused reduction of system behavior conservativeness and more dramatic reconfigurations of coalitional structure during coalitional script execution.

3.2. Conservative adaptation

To evaluate the adequacy of the overall incremental coalition formation and reconfiguration process, we compared it to the global approach. In the referential ["global"] case, each new coalitional structure has been formed from scratch, without any form of reconfiguration of the existing structure. The system consisted of 200 agents, the average number of parallel bids for tasks was 20 (thus, it required formation of 20 coalitions). System was running for 100 reconfigurations.

The quantization error was calculated as the average cosine distance between each agent's capabilities and the requirements of the task assigned to the agent's coalition:

$$AvgQ = \frac{1}{|C|}\sum_{c\in C}\left(\frac{1}{|c|}\sum_{agent\in c} dist(agent, task(c))\right)$$

where $t(c)$ is the task assigned to the coalition c from the coalitional structure C. Quantization error measure has values in the [0,1] interval, the lower values corresponds to the more suitable coalitions.

Figure 1 presents comparison of the referential, global case (no reconfiguration) with our own CF-tree based reconfiguration. Obviously, tree-based local method [grey line] is invincible in terms of computation time, since it rebuilds existing coalitional structure. Another drawback of the global method is that it is not scalable and depends on the total number of agents and coalitions operating in the environment.

In the next experiment, in addition to the global referential case [dashed black line], we had another two cases of reconfiguration scenarios. In one of the scenarios [solid grey line] two new bunches of tasks and agents, with characteristics and behavioral patterns significantly different from those already existing in the system, were introduced: first one after the 33^{rd} reconfiguration and the second one after the 66^{th} reconfiguration.

In the last case [dotted line] agents representing all major behavioral patterns were present in the system from the very beginning. New tasks and new agents were introduced gradually to the system (starting from those quite similar to already working agents up to the distinct ones), after each few reconfigurations. The purpose was to introduce new entities in such a way that the patterns in the environment and CF-tree structure were slowly evolving over time.

As expected, in all the cases system adapts quite efficiently to the changing task requirements and available agent resources. In the global [dashed line] and the evolving patterns [dotted line] case, the quality of the models were comparable in terms of the Average Quantization measure (see Figure 2(a)). Moreover, after a few dozens of reconfigurations, system was capable to exploit the knowledge of existing behavioral patterns and produce coalitional structures which were better than ones constructed globally from scratch.

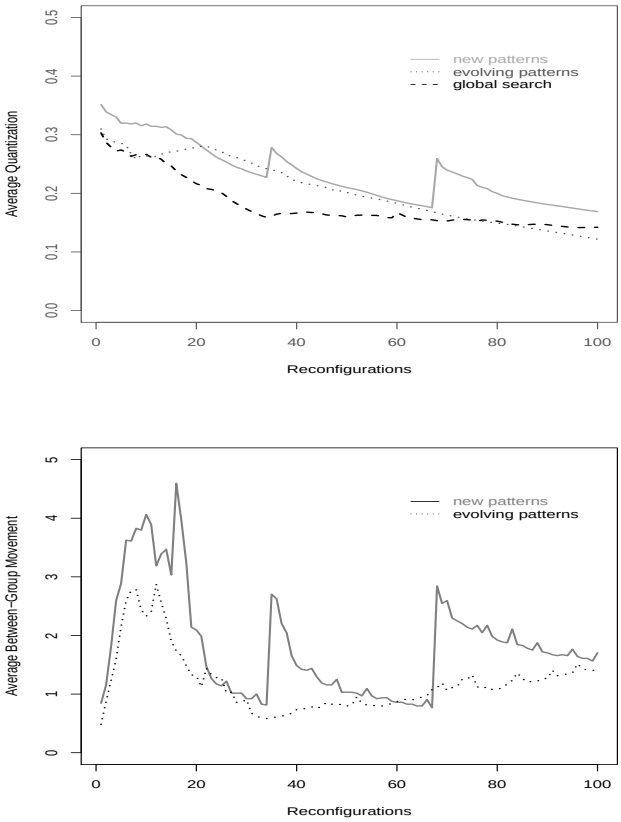


Figure 2. Adaptive reconfiguration: (a) the average quality of coalitional structure (b) the average number of between-group shifts.

At the beginning, results were noticeably worse for the massive appearance of new patterns [solid grey line]. However, after some time system was able to incorporate them into the existing structure and produce structures which were comparable to both other scenarios. It should be noted here that the "massive appearance of new patterns" scenario is rather artificial one and will happen rarely in the real MAS; its main purpose was to present adaptive properties of the conservative reconfiguration.

Figure 2(b) presents the average number of agents changing coalition membership during each reconfiguration. It can be seen that a massive appearance of new patterns causes temporal instability of the model [solid grey line]. Nevertheless, stability and conservative behavior of the coalitional structure is recaptured quite effectively. In case of evolving patterns [dotted line], after a short period of learning, system remain stable during subsequent reconfigurations.

4. Conclusions and future research

Our experimental application is implemented as a open module, enabling future development and comparison of various aspects of cooperation in MASes on one hand, and alternative algorithmic solutions to the problem on the other. We put emphasis on system flexibility, which enables

(by algorithm parameterizations) to control time and precision of computation, depending on particular application, needs, available resources etc.

Requirement of encoding attributes (describing agents and tasks) into real-valued vector space is a deficiency of presented method; and the impact of various encoding strategies on application performance haven't been fully examined. Nevertheless, any coalition building approach needs to utilize some variant of evaluation function, which should enable comparison of *similarities* and *dissimilarities* among different groups of agents. Presented heuristic approach, based on hierarchical clustering, allows to do even quite *wasteful* encoding (i.e. high-dimensional attribute spaces, taking into account diverse aspects of internal [agent] and external [environment] features), while retaining operation effectiveness. However, research on encoding techniques, analogous to some evolutionary algorithms approaches (e.g. [8]) is one of our future research directions.

Other important challenges include realization of *continuity* postulate [4] by taking advantage of specific features of PAM and Clarans algorithm, i.e. transition of configuration graph edges connecting *similar* coalitional structures. We also plan to research into possibility of generalization of the presented solution to issues of *conservative adaptation* of existing collective plans, coping with task decomposition, allocation and reallocation (Partial Global Planning) as well as means-end analysis [9]. Integration with hierarchical contract nets (i.e. explicitly representing hierarchy of task bidders and subcontractor coalitions) would allow to study an influence of various possible agent commitment strategies *early commitment, late commitment, commitment by hiring* [7]) as well as dynamically changing commitment strategies. Last, particularly promising, issue is the development of context-dependent clustering models, enabling local, subjective (different for different agents in the system) vector representations and similarity/dissimilarity measures.

4.1. Approach based on immune system

Typical problem in MAS related applications is that processed data change continually. Moreover, it is common in multi-agent context that the information can be incomplete or uncertain. Individual agents can belong to a different social groups and their behavioral patterns has to be modelled separately. All this requires models which are able to adapt its structure quickly in response to non-stationary distribution changes and can efficiently represent multiple patterns ("evolutionary niches") at the same time. Thus, we decided to implement our own version of artificial immune system (e.g. [2]), to provide adaptive reconfiguration which would be able to cope with the above-mentioned problems.

AIS-based model has numerous advantages: (a) the straightforward adaptation to fuzzy (overlapping) coalitions reconfiguration and to model agents ability to participate in more than one coalition simultaneously, (b) possibility of visual introspection of coalitional structure reconfiguration dynamics, (c) implicit representation of a social memory, enabling detection of agents behavioral and cooperational patterns and adaptation of coalitional structure both on the basis of present environmental state and past experience (so-called *immune response*).

The latter feature is of primary importance since it allows to take a priori problem-domain knowledge into account (including it in fit measure definition) and could constitute basis of previously mentioned context-dependent models. Application of attraction-repulsion fuzzy clustering algorithm allows to include the task specification directly in the clustering process, without dividing computations into two phases: group formation and task allocation. Another important (and common in multi-agent context) issue, which could be handled in this model is information incompleteness and uncertainty during coalition formation.

Also, the natural "niching" seems to be robust in face of rapid changes of the coalitional structure as well as environmental conditions, especially in the case of small agent societies and overlapping coalitions.

References

[1] Ciesielski. *Data mining in multi-agent system reconfiguration*. Thesis, Institute of Informatics, Warsaw Univ., 2003.

[2] de Castro and von Zuben. An evolutionary immune network for data clustering. *SBRN'2000, IEEE Computer Society Press*, 2000.

[3] Dunin-Keplicz. Evolution of collective commitments during teamwork. *Fundamenta Informaticae*, 56(4), 2003.

[4] Dunin-Keplicz and Verbrugge. Reconfiguration algorithm for distributed problem solving. *Engineering Simulation*, 18, 2001.

[5] Dunin-Keplicz and Verbrugge. Collective intentions. *Fundamenta Informaticae*, 51(3), 2002.

[6] Dunin-Keplicz and Verbrugge. Tuning machine for cooperative problem solving. *Fundamenta Informaticae*, 63(2-3), 2004.

[7] Ferber. *Multiagent Systems: An Introduction To Distributed Artificial Intelligence*. Addison-Wesley, 1999.

[8] Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading, MA, 1995.

[9] Jennings, Sycara, and Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[10] Klusch and Gerber. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17(3), 2002.

[11] Mares. Fuzzy coalition structures. *Fuzzy Sets and Systems*, (114), 2000.

[12] Ng and Han. Efficient and effective clustering methods for spatial data mining. *Proceeding of the 20th International Conference on Very Large Data Bases, Santiago, Chile*, pages 144–155, 1994.

[13] Sandholm, Larson, Andersson, and Shehory. Coalition structure generation with worst case guarantees. *Artificial Intelligence J.*, 111(1-2), 1999.

[14] Sandholm and Lesser. Coalition formation among bounded rational agents. *Artificial Intelligence J.*, 101(1-2), 1998.

[15] Shehory and Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence Journal*, 101(1-2), 1998.

[16] Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12), 1980.

[17] Zhang, Ramakrishnan, and Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

Organizational Structure and Responsibility

ABSTRACT

We analyze the organizational structure of multi-agent systems and explain the precise added value and the effects of such organizational structure on the involved agents. To pursue this aim, contributions from social and organization theory are considered which provide a solid theoretical foundation to this analysis. We argue that organizational structures should be seen along at least three dimensions, instead of just one: power, coordination, and control. In order to systematize the approach, formal tools are used to describe the organizational structure as well as the effect of such structures on the activities in multi-agent systems, and especially the responsibilities within organizations of agents. The main aim of the research is to provide a formal analysis of the connections between collective obligations to individual responsibilities. Which individual agent in a group should be held responsible if an obligation directed to the whole group is not fulfilled? We will show how the three dimensions of an organizational structure together with a specific task decomposition determine the responsibilities within a (norm-governed) organization.

Keywords: multi-agent systems, organizational structure, responsibility

1. INTRODUCTION

Many methodologies for multi-agent systems (MAS) are based on organizational structures as their cornerstones. The organizational structure of multi-agent systems involves two basic concepts: *agent roles* and their *relations* in terms of which the collective behavior of individual agents is specified and the overall behavior of the multi-agent systems is determined. The specification of the overall behavior of multi-agent systems concerns the optimization of agents' activities, the management and security of the information flow among agents, and the enforcement of certain outcomes. Agent roles and their relations are often described by a variety of social concepts and relations like norm, power, delegation of tasks, responsibilities, permissions, access to resources, and communication.

The concept of responsibility is a central concept to all legal systems and norm-governed organizations. Analyzing this concept is therefore fundamental if we aim at improving the behavior of these systems or organizations. Obtaining a formal representation of responsibility, however, is quite complex because of the very different meanings of this concept can take. Our concept of responsibility is restricted to the analysis of organizational performance. Therefore, we clarify and classify some meanings of responsibility and we relate them to the three relevant dimensions of an organizational structure we isolated in [11] following intuitions developed in foundational studies in organizations and social theory ([19, 16, 10]). These three relevant dimensions are power, coordination and control, with their matching actions 'to delegate', 'to inform' and 'to monitor'. The coordination actions are actually only one type of meta actions that should be considered. Besides the plan to achieve the content of the obligation the group should create that plan, allocate agents to parts of the plan, create a plan for what to do when the original plan fails, etc. These meta actions should also be coordinated again creating in the end an infinite regression of meta actions. In this paper we will not take all these layers into account, but will limit us to the coordination actions that are necessary to indicate the several notions of responsibility.

In this article we will import some of the studies in organizations and social theory to describe a more rigorous foundation of organizational structures in MAS, which will be informally and formally exposed in Section 2. In order to describe organizational structures we have to first describe exactly what the meaning is of the relations that form the structure. E.g. what is the meaning of an "power" relation and, maybe even more importantly, what are the consequences of the existence of such a relation between two agents? We will introduce a modal logic for this characterization. Several notions of responsibility (given a plan) will be discussed formally in Section 3. How the individual responsibilities relate to the underlying structure of an organization will be discussed in Section 4. In the last section, we will draw some conclusions and give directions for future research.

2. ORGANIZATIONAL STRUCTURE AND ITS LOGIC

Organizations "represent rationally ordered instruments for the achievement of stated goals" ([19]), that is, organizations arise in order to achieve specific objectives, and these objectives are pursued defining a number of subgoals contributing to the overall purpose of the organization. These subgoals identify the roles that are played in the organization. The relation between subgoals and overall objectives of the organization, i.e., the primitive decomposition of tasks within the organization, defines the essential form of organizational structure: "viewed in this light, formal organization is the structural expression of rational action" [19]. Roles are the basic units over which this structure ranges determining the source

of the "rational order" holding in the organization. (Social) roles allow to specify the activities delegated by a social institution to individuals to achieve its purpose, while abstracting from the individuals which will eventually play them (cf. [2]).

The above quotes show some of the sources of organizational structures and indicate why work on organization in MAS[1] presents organizational structure as something mono-dimensional, though it often, but only implicitly, considers a multiplicity of structured aspects: "authority", "communication", "delegation", "responsibility", "control", "decision-making", "power", etc. The thesis we hold here, which is inspired by foundational work on social and organization theory like [19, 16, 10], is that organizations do not exhibit one single structural dimension, but that they are instead multi-structured objects. In particular, we view organizational structure as hiding at least three relevant dimensions which we call: power, information and control. We will analyze **power** in relation with the delegation activity, **coordination** in relation with the knowledge and information issues, and **control** in relation with the monitoring and recovery issues. As a result of this analysis, organizations will be represented as explicitly displaying a triple structure constrained on the basis of the interplay between the three notions of power, coordination, and control. It is the structure based on goal or task decomposition and related to power and delegation capabilities between the roles. Although we do not pretend to give full definitions of these relations (see [5, 14] for some more elaborate definitions of the delegation and power relations) we will characterize these relations in terms of some of their consequences for the agents, enacting the roles, between which these relations are defined.

The capabilities of a role are also an issue worth mentioning, though we will not consider it in detail here. It is somehow analogous to the information issue since it concerns what is presupposed by each role in order to achieve the relevant goals and comply with the relevant norms. A basic type of capabilities lies in the amount of resources that agents should have at their disposal. A second kind of capabilities play a central role in organizations, namely those concerning the so-called *institutional power* ([14, 4]). Again the problem is related with the dynamics introduced by the "delegation structure": delegating a task may require a parallel enabling or empowering activity such as making the relevant resources accessible, e.g., electronic money, and providing the required form of institutional empowerment, e.g., a suitable document. In this case the relevant structural question is: who enables or empowers whom? In order not to complicate matters further we assume in this article that all agent have the capabilities needed to enact the role they fulfill, leaving this issue to future work.

To describe an organization and its structure we will use a (typed) multi-modal propositional logic. The organizational structures are denoted through the special propositions $Power(r, s)$ to indicate that 'the agent enacting role r has the agent enacting role s in its power' (i.e. the agent playing role r can delegate goals to the agent playing role s), $Coordination(r, s)$ to indicate that 'the agent enacting role s has access to the information that is accessible to the agent enacting role r', and $Control(r, s)$ to indicate that 'the agent enacting role r controls the agent enacting role s' (i.e. the agent playing role r is responsible for the agent playing role s). Note that these relations are defined on roles. We denote the fact that agent i enacts role r, i.e., is a *role enacting agent* ([9]), by the special proposition $rea(i, r)$. Furthermore we use a modal operator K_i for knowledge accessible to an agent i. For the characterization of the organization structures we build on dynamic logic ([12]). Dynamic formulas such as $[\xi]\phi$, meaning that after each execution of ξ formula ϕ holds, where ξ is a parameterized construct of the type $i : \alpha$ denoting the performance of action α by agent or role i, or a composed construct such as: $i : \alpha_1; j : \alpha_2$ (subsequent performance), $i : \alpha_1 \& j : \alpha_2$ (parallel performance), $\overline{i : \alpha}$ (i refrains from performing α). The formal semantics is given by means of a Kripke structure where there are accessibility relations $R_{i:\alpha}$ associated with each parameterized action $i : \alpha$.

We can now give a full formal definition of the syntax of our description language Org:

DEFINITION 1 (SYNTAX OF Org). *Given a finite set AR of role names, a finite set Ag of agent names, a countable set P_0 of atomic propositions, a finite set of parameterized actions $\mathcal{A}$ (in general the elements of $\mathcal{A}$ are denoted by $i : \alpha$ with i in Ag the performing agent of α) containing at least $i : achieve(\phi)$, $i : delegate(j, \phi)$, $i : inform(j, \phi)$ and $i : monitor(\phi)$, the propositional language L_p that is built up from atoms P_0, the countable set $P = L_p \cup \{Power(r, s), Coordination(r, s), Control(r, s), rea(i, r) \mid r, s \in AR, i \in Ag, \phi \in L_p\}$, the admissible formulas are recursively defined as follows:*

- $P \subseteq Org$
- *If ϕ and $\psi \in Org$, then $\phi \wedge \psi$, $\neg\phi \in Org$*
- *If $\phi \in Org$ and $i, j \in A$, then $K_i(\phi) \in Org$*
- *If $\phi \in Org$ and $i : \alpha \in \mathcal{A}$, then $[i : \alpha]\phi \in Org$*
- *If $i : \alpha \in \mathcal{A}$, then $DONE(i : \alpha), DO(i : \alpha), O(i : \alpha)$ and $Can(i : \alpha) \in Org$*

Binary connectives $\rightarrow$ and $\vee$, and nullary connective $\bot$ can be defined as usual. For the knowledge operators (K_i) we assume the axiomatization characterizing **S5**. The assertions DONE($i : \alpha$) stands for "α has just been performed by agent i", DO($i : \alpha$) stands for "α is going to be the next action performed by agent i", and $CAN_i(\alpha)$) stands for "α lies in the capabilities of agent i". $O(i : \alpha)$ is the deontic assertion to the effect that agent i ought to perform action α.

The semantics of Org will be given in two steps. First we define the semantics of the special relations $Power$, $Coordination$ and $Control$ through a multi-digraph defined on the set of roles. This defines a tuple OS which will be part of the Kripke model given after.

We will only introduce some basic elements which are strictly of use for the development of this article.

DEFINITION 2. **(Organizational structures)**
OS is characterized by the following:

$$\langle Roles \cup Agents, \{R_{Power}, R_{Coordination}, R_{Control}, Rea\}\rangle$$

where $Roles \cup Agents$ is the finite set of roles and agents; and $\{R_{Power}, R_{Coordination}, R_{Control}\}$ are three irreflexive binary relations on $Roles$ characterizing the Power, Coordination and Control structures. Rea indicates which agents play which roles.

The semantics of Org is defined in terms of Kripke models (cf. [11]).

3. A FORMAL ANALYSIS

[1] See [13] for an exhaustive survey.

3.1 Task allocation

In order for organizations to fulfill their objectives, subtasks are isolated via a form of organizational planning and distributed in a way which defines the roles agents can play in contributing to the performance of the organization. We call this designing process of the activity of an organization *task-allocation*. Roles can then be seen as sort of placeholders in a rationally designed activity of an organizations: an agent taking part to the organization will occupy one of these places, that is, will play a role[2]. In this work, agents playing a role in an organization are called, following [9], *role enacting agents* or *rea*'s.

The distribution of the sub-tasks in an organization in order to achieve a certain goal or collective task τ depends on a plan of the organization, i.e., a concrete manner to achieve the goal (collective task). We can define a plan to achieve a certain goal τ as a decomposition of the complex action $achieve(\tau)$ by a sequence of (possible simultaneous) individual actions:

$$Plan(achieve(\tau)) =$$

$$\langle achieve(\tau_1) \bullet achieve(\tau_2) \bullet \ldots \bullet achieve(\tau_n) \rangle$$

such that $[achieve(\tau_1) \bullet achieve(\tau_2) \bullet \ldots \bullet achieve(\tau_n)]\tau$,

where $\bullet$ stands for either the simultaneous operator '&' or the sequential operator ';'. The action $achieve(\tau_1)\&achieve(\tau_2)$ stands for the simultaneous performance of $achieve(\tau_1)$ and $achieve(\tau_2)$, and action $achieve(\tau_1)$; $achieve(\tau_2)$ stands for the sequential composition of $achieve(\tau_1)$ and $achieve(\tau_2)$.

We need the simultaneous operator, since some actions have to be performed at the same time. The sequential operator is needed because some actions might depend on other ones: a certain action can only be performed if an other action is done. So, the plan must at least determine the *order* of sub-actions. For example, the notification of acceptance of a certain paper by an Editorial Board can only be done if it is reviewed by some members of the Editorial Board. The task-based responsibility of the performance of an action α by an agent depends not only on the individual who is committed to perform the action α, but also on agents who have to perform actions which are necessary to perform action α[3].

Besides task division, task allocation is needed, which indicates which role of the organization has to achieve which sub-task of the complex task. We use the following definition for task allocation:

DEFINITION 3. **(Task allocation)**
A task allocation for a task τ within the set of roles RA is defined as follows:

$$\langle r_1 : achieve(\tau_1) \bullet r_2 : achieve(\tau_2) \bullet \ldots \bullet r_n : achieve(\tau_n) \rangle$$

such that

$$[r_1 : achieve(\tau_1) \bullet r_2 : achieve(\tau_2) \bullet \ldots \bullet r_n : achieve(\tau_n)]\tau.$$

We refer to the task allocation of τ within RA as $Plan(RA, \tau)$. To indicate that task $achieve(\tau_j)$ has been allocated to role r_j in $Plan(RA, \tau)$ (for $j = 1, 2, \ldots, n$), we use the following notation: $\langle r_j : \tau_j \rangle \in Plan(RA, \tau)$[4].

We will use the concept of task allocation as a starting point for framing the various notions we are interested in. In particular, as we will see in the coming section, it plays an essential role for the definition of the notion of task-based responsibility. Besides, we will analyze the notion of "failure" in the accomplishment of a task understanding it as an organizational variant of the notion of social harm described in [6]. In our context, we define the *untoward event* $D\tau_r$ as the impossibility, or the reduction of the possibility to achieve the goal τ allocated to role r. The performance of an action α by an agent i enacting role r determining social harm can then be represented as $[i : \alpha]D\tau_r$, that means, after each execution of action α by agent i the social harm represented by $D\tau_r$ is the case.

3.2 Responsibilities in form

Thus far we have dealt with organizations at their role level, where the task-allocation and the organizational structure range. Responsibilities concern agents and arise in relation with task-allocation and structure once there are agents enacting the roles of a given organization.

Given a task-allocation allocating a specific subtask to a role, and given that an agent is enacting that role, the agent is then said to be responsible for that task or *task-based responsible*. In other words, the allocation of subtasks to roles determines a distribution of what we call *task-based responsibilities* over the set of agents enacting the roles of the organization. Being autonomous, agents can independently decide whether to perform the subtasks to which they are allocated or not, and whether to perform them in the expected way. In this case the fulfillment of the organizational objectives is put in jeopardy by the conduct of some agent that is said then to be *causally responsible* for the failure occurred.

In organizations an agent can happen to be causally responsible of some failure without actually being blamed by the organization. This can happen if an agent i which is task-based responsible for performing a task, *delegates* the performance to a subordinate agent j which fails or jeopardizes the execution of the delegated task. This observation reveals an interplay between the notions of responsibility isolated above, and dimensions of social structure such as the possibility to delegate allocated tasks, i.e., what we called *power relation* in the previous section. Social structure in relation with responsibility will be discussed in detail in Section 4. Here it suffices to notice that the presence of a power structure within an organization causes a difference between the two notions of task-based and causal responsibility: 'I may have not performed the task you delegated to me, but you were the one appointed to it'. Therefore, if an organizational task is not performed, the one being *socially responsible* in front of the organization, the one who gets the blame for the failure, is not necessarily the one causally responsible for it, but it is the one to which that task was appointed. The acknowledgment of such a gap calls for the distinction of yet another meaning of the notion of responsibility which we call *failure-based responsibility*: who should control the performance of an agent to check whether a failure occurs and take countermeasures if that is the case?

We can now provide an action logic representation of the notions of responsibility.

Causal responsibility

[2]We presuppose a distinction between two ways of intending the notion of role within an organizations: role as role-type, and role as role-token. Examples of role-types are the university roles of 'professor' or 'PhD Student'. Role-tokens are instead the specific 'professor' and 'PhD student' positions, like 'professor of x at department y' etc. The notion of roles as placeholders in the organizational activity corresponds to the notion of role-token.

[3]These ideas about the notion of plan are quite standard in the literature about planning in Artificial Intelligence (see, e.g., [17, 7]).

[4]Note that the function of the numeric index j consists in denoting the position within the task allocation sequence.

An agent is said to be *causally responsible* when it does something (or fails to do something) that causes the untoward event D_τ. We formalize causal responsibility as follows:

DEFINITION 4. **(Causal responsibility)**
For all $i \in Ag$:

$$R_i^c(D\tau) := [i:\alpha]D\tau \wedge DO(i:\alpha) \wedge \neg D\tau$$

meaning that agent i is causally responsible for the untoward event if and only if agent i performs an action which necessarily determines the occurrence of the untoward event and, finally, the untoward event is not the case before the agent performs the action.

Causal responsibility can also be attributed to nonhuman events, for example, that a house is severely damaged in a storm. In this article, we restrict ourselves to agents in an organizational context. Notice that an agent which is causally responsible, may not be considered *blameworthy*. For example, if the chairman of the Editorial Board has forgotten to inform a member i to review some papers in one week, and agent i did not review the papers in one week, then the achievement of the goal of the Editorial Board to notify of the results of the reviews within the deadline will be reduced. The member i would be considered responsible in the sense of having *caused the situation*, but he would not be responsible in the sense of *blameworthy*. An agent does something blameworthy, if he knows (or could have known) that the action he performs leads to the impossibility or the reduction of the possibility to achieve a goal τ:

DEFINITION 5. **(Causal blameworthiness)**
For all $i \in Ag$:

$$Bl_i^c(D\tau) := [i:\alpha]D\tau \wedge DO(i:\alpha) \wedge \neg D\tau \wedge K_i([i:\alpha]D\tau)$$

The importance of the knowledge component in the dynamics of responsibilities within organizations is analyzed in detail in Section 4.

Task-based responsibility
The notion of *task-based responsibility* is somehow interchangeable with duty and refers to what individuals are expected to do in virtue of their social roles. We assume that task-based responsibility is a consequence of role adoption: an agent who accepts to play a given role in an organization takes a responsibility with regard to the accomplishment of that role, i.e., with the tasks associated to it [6]. In this article, this notion of responsibility completely depends on the position an agent occupies in the performance of the organization.

DEFINITION 6. **(Task-based responsibility)**
For all $i \in Ag$ and a task allocation $Plan(AR,\tau)$:

$$R_i^{tb}(\tau_j) := rea(i,r_j) \wedge \langle r_j:\tau_j\rangle \in Plan(AR,\tau) \wedge$$

$$O(i:achieve(\tau_j)) \wedge [\overline{i:achieve(\tau_j)}]D\tau_j$$

The obligation $O(i:achieve(\tau_j))$ expresses that the organization entrusts agent i with his task τ_j ($rea(i,r_j) \wedge \langle r_j:\tau_j\rangle \in Plan(AR,\tau)$), and $[i:achieve(\neg\tau_j)]D\tau_j$ expresses the empowerment of i to prevent the reduction of the possibility or the impossibility to achieve goal τ_j. So, an agent i fails to fulfill his task-based responsibility $R_i^{tb}(\tau_j)$ if he violates the norm $O(i:achieve(\tau_j))$ which leads to the untoward event $D\tau_j$. However, the agent is considered blameworthy when he actually knows (or could have known) that he has this obligation and that he can perform the action to achieve his task. For example, he has not received the information needed for the performance of his task, or the achievement of his task depends on an earlier task in the task allocation which is not performed. This notion of blameworthy can formally be described as follows:

DEFINITION 7. **task-based blameworthiness**
For all $i \in Ag$ and a task allocation $Plan(AR,\tau)$:

$$Bl_i^{tb}(\tau_j) := rea(i,r_j) \wedge \langle r_j:\tau_j\rangle \in Plan(AR,\tau) \wedge$$

$$O(i:achieve(\tau_j)) \wedge [\overline{i:achieve(\tau_j)}]D\tau_j \wedge$$

$$K_i(O(i:achieve(\tau_j)) \wedge [\overline{i:achieve(\tau_j)}]D\tau_j) \wedge CAN_i(achieve(\tau_j))$$

Social responsibility
The notion of social responsibility builds on the notion of task-based responsibility, and it is somehow analogous to a notion of violation in standard deontic logic.

DEFINITION 8.
For all $i \in Ag$ and a task allocation $Plan(AR,\tau)$:

$$R_i^s(\tau_j) := R_i^{tb}(\tau_j) \wedge D_{\tau_j}$$

that is to say, agent i has the responsibility to achieve τ_j and the achievement of τ_j is impossible or jeopardized. Notice that this notion of responsibility is very simple and is independent from the notion of causal responsibility.

4. RESPONSIBILITIES AND ORGANIZATION STRUCTURE

Although the notion of organization structure does not play a direct role for the definition of the different types of responsibilities, it does play an important role in the dynamics of responsibilities within an organization. First of all, the causal and task-based responsibilities depend on a given task allocation, which splits the joint task (arising from the collective obligation) over the individual agents of the organization. The mechanism of this task allocation depends on the organizational structures. E.g., in an organization without any hierarchy it might be that all agents bid on some sub-task(s) of the tasks and allocation is done through choosing the optimal allocation based on these bids. In a purely hierarchical organization the allocation might be done through a delegation of sub-tasks through the hierarchy. So, these mechanisms depend on the existing power relations between the agents in an organization.

While formally task-based responsibilities can be allocated through the task allocation and the role enactment mechanisms, we already saw that agents should also be made aware of their responsibilities. The mechanism can be implicit, e.g., through the playing of a role in an organization an agent knows it has certain responsibilities that come with the role, but can also be explicit. For the latter an information structure should be present in the organization that facilitates the right dissemination of the information, such that agents also know they have a responsibility and can be held responsible when things go wrong. The latter point ties in with the last dynamic aspect of responsibilities. When an agent is task-based responsible for a task it should also perform the task. The organization should, in some way, monitor the progress of the performance such that both the blame for non-performance can be attributed correctly to an agent and also appropriate measures can be taken to repair the situation. This monitoring and repair (in case of failures) is structured along the control structure of the organization.

We cannot hope to provide a full account of all interactions between responsibilities and organizational structures. However, in the rest of this section we aim to capture some essential traits of

those interconnections. We understand those relations essentially as guaranteeing some effects to the basic actions of *delegate*, *inform* and *monitor*, which play an essential role with respect to responsibilities and their development in organizations.

The following definitions characterize the influence of the organization relations on the actions above. Through these basic properties we can also formally analyze some consequences of them on the notions of responsibilities studied in the previous section.

DEFINITION 9. **(Power)**
For all $i, j \in Ag$ *s.t.* $i \neq j$ *and* $r, s \in AR$*:*

$$(Power(r,s) \wedge rea(i,r) \wedge rea(j,s)) \rightarrow$$

$$[i : delegate(j,\phi)]O(j : achieve(\phi))$$

If a power relation exists between roles that are enacted by two agents then a *delegate* action will have as effect an obligation for the recipient, that is, a form of "your wish is my command" principle. Intuitively, if a power relation holds between roles r and s, all delegation acts performed by an agent i enacting role r on agents enacting role s succeed in creating an obligation for these agents.

Task-based responsibility cannot be delegated. If Agent i has, according the task allocation, to achieve task ϕ and has a power relation with agent j, he can delegate his task to j, but he remains task-based responsible for the achievement of ϕ. Since ϕ is not the original task of agent j according to the given task allocation (see definition 6). Agent j, however, can be causally responsible if he fails to fulfill his delegated obligation.

A difference between an individual task and a collective task is that in an individual task all information is readily available and can be reasoned about. However, when a collective task is divided over the individuals of that collective, they might not know the whole plan, typically do not have information about actions that are performed, etc. Therefore, we need a coordination structure.

DEFINITION 10. **(Coordination)**
For all $i, j \in Ag$ *s.t.* $i \neq j$, $r, s \in AR$*:*

$$(Coordination(r,s) \wedge rea(i,r) \wedge rea(j,s)) \wedge$$

$$DONE(i : monitor(\phi)) \wedge$$

$$(K_i\phi \rightarrow O(i : inform(j,\phi))) \wedge [i : inform(j,\phi)]K_j\phi.$$

If a coordination relation holds between roles r and s, all information acts performed by agents enacting role r to agents enacting role s are successful in the sense that they create, in these last agents, the knowledge they acquired via monitoring the occurrence of a certain fact: the *inform* action will automatically lead to the corresponding epistemic state in the recipient. Further, there is a normative aspect: agent i *should* inform another agent j about ϕ if they are connected through a coordination link and if agent i has monitored (checked) ϕ.

On this basis, a coordination-related type of responsibility can be defined.

DEFINITION 11. **(Coordinational responsibility)**
For all $i, j \in Ag$ *and a task allocation* $Plan(AR, \tau)$*:*

$$R_i^{coor}(inform(j,\phi)) := rea(j,r_l) \wedge \langle r_l : \tau_l \rangle \in Plan(AR,\tau) \wedge$$

$$O(i : achieve(K_j\phi)) \wedge [\overline{i : achieve(K_j\phi)}]D\tau_l$$

This notion of responsibility has some resemblance to the task-based responsibility (see definition 6), with as task for agent i to inform agent j about ϕ. On the basis of the coordination structure, there is a specific allocation of the information actions, which is needed for the achievement of the individual tasks in the task allocation. Given this definition, we can say agent i is responsible to inform agent j, when the knowledge of ϕ is a necessary means to the achievement of τ_l and that agent j does not have that knowledge.

The responsibility of an agent i to inform some agent j about a certain aspect ϕ can follow from the coordination link between these agents if the knowledge of ϕ is necessary for the achievement of the task of agent j according the task allocation and i can monitor or check ϕ. This shows, in particular, how a given task allocation needs to be integrated with a suitable allocation of coordinational responsibilities in order to guarantee the information necessary for the correct functioning of the organization. This property can be formalized as follows:
For all $i, j \in Ag$ s.t. $i \neq j$, $r_k, r_l \in AR$ and task allocation $Plan(AR, \tau)$:

$$\begin{aligned} Coordination(r_k, r_l) \wedge rea(i, r_k) \wedge rea(j, r_l) \wedge \\ \langle r_l : \tau_l \rangle \in Plan(AR, \tau) \wedge \\ (\neg K_j\phi \rightarrow \neg CAN(j : achieve(\tau_l))) \wedge \\ CAN(i : monitor(\phi)) \rightarrow R_i^{coor}(inform(j,\phi)) \end{aligned}$$

So, agent i is responsible to inform agent j about ϕ if there is a coordination link between the roles r_l and r_k they respectively enact, and without the information about ϕ agent j cannot perform his task according to the task allocation. If agent i does not inform agent j, it follows that agent j cannot perform his task, which can lead to $D\tau_l$. So, agent i can be causally responsible if he does not inform agent j about ϕ (see definition 4). Note, that agent j is still task-based responsible with respect to τ_l, but not blameworthy, when he does not get the information necessary for the achievement of τ_l (see definition 7).

Finally, we get to a characterization of the dimension of control in organizational structure:

DEFINITION 12. **(Control)**
For all $i, j \in Ag$ *s.t.* $i \neq j$ *and* $r_k, r_l \in AR$*:*

$$(Control(r_k, r_l) \wedge rea(i, r_l) \wedge rea(j, r_l)) \rightarrow$$

$$[i : monitor(DONE(j : achieve(\phi))](D\tau_l \rightarrow O(i : achieve(\tau_l))$$

If a control relation exists then the *monitor* action will have as further consequence the generation of an obligation for the controller in case the controlled actor did not achieve the relevant state causing the untoward event. On this basis, the notion of *failure-based responsibility* can be defined.

DEFINITION 13. **(Failure-based responsibility)**
For all $i, j \in Ag$ *s.t.* $i \neq j$ *and* $r, s \in AR$*:*

$$R_i^{control}(monitor(j,\phi)) := Control(r,s) \wedge rea(i,r) \wedge rea(j,s)$$

This type of responsibility depends completely on the control relation.

The control responsibility has another normative aspect: if an agent has control over another agent he is obliged to monitor the controlled agent whenever he knows the controlled agent has an obligation. Formally,

$$(R_i^{control}(monitor(j,\phi)) \wedge K_i(O(j : achieve(\phi))) \rightarrow$$

$$O(i : monitor(DONE(j : achieve(\phi))))$$

We can imagine that an agent who has delegated his task to agent j, has the obligation to monitor whether the delegated agent has done the task, since he might be responsible to monitor agent j and he knows that the delegated agent j has the obligation.

5. CONCLUSIONS

We have provided some elementary notions of responsibility in its interconnection with the structure of an organization. We argued that organizations are defined through several structural relations. Although people refer to these structures they still lack a precise formal definition. In this article these relations have been given a solid foundation. This allows us to check desirable properties of the structures and how they (should) interact. Now we have a characterization and can proof properties given some structural properties of these relations. In future work we will look at more elaborate definitions of the power, coordination and control relations.

The notions of responsibilities are useful in the process of designing an organizational structure, and conversely in understanding how a given organization is structured. We have shown that responsibilities have an impact both on what agents should do within an organization, but also on who to turn to when things go wrong.

Responsibilities are closely related to the specific task allocation within an organization. Although the task allocation can be determined dynamically through the process of delegation, some of it is predetermined through the role structure of the organization which assigns typical tasks to certain roles. The organizational structure plays an even greater role in the monitoring and control of execution of the tasks for which the agents are responsible. The logical framework we presented offers a semantics for the notions of responsibility that is necessary for determining at least some interconnections between organizational structure and responsibilities. It gives some insides into when an agent can really be held responsible for when tasks are not (or wrongly) performed. These observations might lead to guidelines for the design of an organizational structure given that one wants some responsibilities to be covered at all times. In this article we just offered a glance of these observations through the example. However, we hope to extend this area in future work, e.g., to combine our work with the work done in [18] about the representation of organized interaction with action concepts.

Another line of future work concerns the logical formalism that was used to describe the notions in this article. Although it is sufficient to denote most of the basic properties and relations between them, we also touched upon some fundamental issues. In general responsibility is closely related to "causal" relations, i.e., who is actually causing some action or state (cf. [15]). This is a famous problem to represent in logic and we can only hope to give a close approximation that is good enough for the present purpose.

A third related point is the use of temporal relations. In further work we will explore the dynamics of the responsibilities, their persistence and evolution over time. In order to do this we need to combine the current formalism with a temporal framework in the same vain as was done for temporal dynamic deontic logic in [8, 3].

6. REFERENCES

[1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, 2001.

[2] G. Boella and L. van der Torre. The ontological properties of social roles: Definitional dependence, powers and roles playing roles. In E. Francesconi J. Lehman, M.A. Biasiotti and M.T. Sagri, editors, *Legal Ontologies and Artificial Intelligence Techniques*, pages 25–36, Tilburg, 2005. Wolf Legal Publishers.

[3] J. Broersen, F. Dignum, and V. Dignum. Designing a deontic logic of deadlines. In A. Lomuscio and D. Nute, editors, *Proceedings of DEON'04*, pages 43–56, 2004.

[4] C. Castelfranchi. The micro-macro constitution of power. *ProtoSociology*, 18:208–268, 2003.

[5] C. Castelfranchi and R. Falcone. From task delegation to role delegation. In M. Lenzerini, editor, *LNAI 1321. AI*IA 97: Advances in Artificial Intelligence*, pages 278–289. Springer Verlag, Berlin, 1997.

[6] R. Conte and M. Paolucci. Responsibility for societies of agents. *Journal of Artificial Societies and Social Simulation*, 7, 2004.

[7] I. Nunes de Barros, A. Valente, and V.R. Benjamins. In *AIPS 1996*, pages 11–18, 1996.

[8] F. Dignum, J. Broersen, V. Dignum, and Meyer J-J. Ch. Meeting the deadline: Why, when and how. To be published in Proceedings of FAABS III Workshop, Washington, April 2004.

[9] V. Dignum. *A Model for Organizational Interaction*. SIKS Dissertation Series, 2003.

[10] A. Giddens. *Social Theory and Modern Sociology*. Polity Press, 1984.

[11] D. Grossi, L. Royakkers, F. Dignum, and M. Dastani. Foundations of organizational structure in multi-agent systems. In F. Dignum, editor, *Proceedings of AAMAS'05, Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 690–697. ACM Press, 2005.

[12] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 497–604. Reidel, Dordrecht, 1984.

[13] B. Horling and V. Lesser. A Survey of Multi-Agent Organizational Paradigms. Computer Science Technical Report 04-45, University of Massachusetts, May 2004.

[14] A. J. I. Jones and M. Sergot. A formal characterization of institutionalised power. *Journal of the IGPL*, 3:429–445, 1996.

[15] J. Lehmann. Towards the formalization of legal causal reasoning. In *DEXA Workshop 1999*, pages 780–784.

[16] O. Morgenstern. Prolegomena to a theory of organizations. Manuscript, 1951.

[17] S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice Hall International, 2001.

[18] F. Santos, A.J.I. Jones, and J. Carmo. Responsibility for action in organisations: a formal model. In G. Hintikka and R. Tuomela, editors, *Contemporary Action Theory, volume Synthese Library 267*, pages 333–350, Deventer, 1997. Kluwer.

[19] P. Selznick. Foundations of the theory of organization. *American Sociological Review*, 13:25–35, 1948.

Agent-based coalition formation in disaster response applications

Ladislau Bölöni, Majid A. Khan and Damla Turgut
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816,
Email: lboloni@eecs.ucf.edu,majid@hector.cs.ucf.edu, turgut@eecs.ucf.edu

***Abstract*— We present an agent-based coalition formation approach for disaster response applications. We assume that agents are operating in a dynamic and dangerous environment, and they need to form convoys to efficiently traverse unsafe areas and randez-vous at task locations. We assume a dynamic model of multilevel coalition formation where agents can dynamically join and leave the convoy, and the goals of the convoy (as reflected in its physical path and schedule) is determined by its current members. We demonstrate our approach in a simulation study located in the environment of New Orleans in the hurricane Katrina aftermath.**

I. Introduction

Efficient disaster response requires participants to form teams and coordinate their actions. This process is complicated by a variety of factors:

Dynamic, unpredictable and dangerous environment. In the immediate aftermath of a disaster (such as the hurricane Katrina in New Orleans or the Asian Tsunami) previously safe areas might turn into unsafe or unaccessible. The environment might contain new sources of danger in the form of natural obstacles (damaged buildings) or even hostile agents (such as looters or stray dogs).

Dynamic tasks. In rescue missions, tasks appear unpredictably. The discovery of a wounded person at a dangerous location creates a new task with specific logistics, protection and medical facets. In severe disasters, the number of tasks can greatly exceed the available resources.

Dynamic teams and collaboration patterns. Although some of the disaster management teams are pre-established, trained together and have a clear pattern of command and control, many teams are assembled on an ad hoc basis, as a response to emerging tasks. Teams are composed from heterogeneous groups of entities: persons, vehicles, service animals, and so on. Team members might not report to the same chain of command, might have communication problems and their interests might not be completely aligned. For instance, the state police and guerilla groups might cooperate in a rescue operation but resume hostilities after the emergency.

Our research group at the Networking and Mobile Computing (NetMoc) laboratory at University of Central Florida is working on a negotiation based coalition formation approach which can be used to assemble ad hoc coalitions in an emergency management scenario. In this paper, we are concentrating on the negotiation regarding convoy formation for mobility in a dangerous environment. The convoy formation approach, which is based on physical destinations needs to be complemented with a decision process based on assigning tasks based on the roles the agents are able to assume in a team. This task-based component of our system is based on extensions to the Machinetta framework [1].

The remainder of this paper is organized as follows. In Section II, we describe the formalism used for the description of the physical world, the agents and coalition formation. The convoy formation and negotiation models are described in Section III. The experimental results are presented in Section IV. Related work is discussed in Section V and we conclude in Section VI.

II. World model

The environment considered in this paper assumes a 2-dimensional geographic area, where we identify: *safe areas* which are traversable by any vehicle and convoy, *danger areas* which are traversable only by convoys and *inaccessible areas*. The model can be extended in a straightforward way to involve more than three area types which affect the movement of the vehicles in a variety of ways (such as slowing down, requiring higher energy consumption, and so on). In this environment, we consider the actions of a set of *embodied agents*, which have a well-defined physical location and movement capabilities. In practice, these agents can be "RAP" (Robots, Agents and/or Persons). The goal of every agent is to execute a certain task at a destination location.

The time to reach the destination can be improved by the formation of convoys. In certain cases, the agent can not reach the destination except through joining convoys. We assume the agents *self-interested* but *honest*; the agents keep their negotiated commitments. The embodied agents are using message based communication, which can be either point to point or broadcasted to all other agents in the transmission range.

A. Negotiation model

Negotiation is the process by which a group of agents come to a mutually acceptable agreement on some matter [2]. In our scenario, the subject of negotiation is the joining and leaving convoys, and the adaption of the path of the convoy to the requirements of the agent. The agents are exchanging a set of

offers, based on their *offer construction strategies*. The other party is using its *offer evaluation strategy* to make a decision, which can be either to accept the offer, send a counteroffer or terminate the negotiation. Two subprotocols (for joining and leaving convoys) describe the message flows for the different negotiation processes.

While this model is common for any kind of negotiation processes, the fact that the negotiation happens in real-time in the physical world, creates a set of new requirements. The negotiation needs to be *time constrained*, i.e. the time allotted to the concrete flow of messages needs to be limited. The agents being in constant movement, they will be in their communication range for a limited amount of time. The negotiation needs to be *fail safe* due to the frequent loss of messages either due to temporary causes, or because a vehicle got out of the communication range during the negotiations. It is especially problematic if a negotiation is interrupted with the parties having a different view of the outcome of the negotiation. The negotiation has to be *deadline oriented* because the offers made during negotiations can become obsolete. For instance, if an agent A makes an offer to join convoy B at location X and time t, there is a limited temporal window of opportunity when this rendez-vous can take place. Thus every offer needs to carry a timestamp and an expiration date. Also, since one vehicle can take part in several negotiations simultaneously, the negotiation process should make sure that it does not commit to contractual binding with more than one vehicles at the same time.

B. Convoy models

We define a *convoy* as a coalition of embodied agents which agreed on a common path and schedule. Normally, the agents of the coalition have a common location and speed; however, from a logical perspective, we consider an agent which has agreed to join a convoy and it is on its way to a rendez-vous point as part of the convoy.

Convoys have a hierarchical structure, and may contain subconvoys. For the sake of uniformity, we will consider that individual vehicles are being part of single-vehicle convoys. Formally, a convoy C is described by a set of convoys $S = \{C_1, \dots C_n\}$, a leader agent $A_L \in S$, and a set of commitments $G = \{g_1, \dots g_n\}$. The set of commitments are usually expressed as constraints on the path of the convoy. The role of the leader is to negotiate on behalf of the convoy and to determine its path, taking into account its previous agreements G. Although the embedded convoys maintain their leader and set of agreements, the path of the convoy is determined exclusively by the leader of the outermost convoy. The negotiation protocols need to ensure that the agreement sets of the subconvoys are compatible with the agreements of the embedding convoy.

The commitment of the convoys are related to visiting locations and can be classified as "before" (B) and "after" (A) commitments. A "before commitment" $B(L,t)$ commits the convoy to arrive to location L *not later than* time t. An "after commitment" $A(L,t)$ commits the convoy to leave location L *not sooner than* time t (if the convoy reaches that location sooner, it can, of course wait at the location).

We will call a commitment C_1 *stronger than* a commitment C_2 and denote it $C_2 \subset C_1$ if every set of actions which satisfies C_1 also satisfies C_2.

Theorem 1:

$$\forall L, t_1 < t_2 \Rightarrow B(L,t_2) \subset B(L,t_1)$$
$$\forall L, t_1 < t_2 \Rightarrow A(L,t_1) \subset A(L,t_2)$$

We leave the proof of this theorem as an exercise to the reader.

III. Convoy formation mechanism

A. Negotiation for an agent joining a convoy

The convoy joining mechanism is inherently asymmetric, even if it takes place between two single-agent convoys. The leader agent of the first convoy will become the leader of the resulting convoys. There is an asymmetry in the negotiation interests of the main convoy and the joining agent or convoy.

Let us now consider the lifecycle of the embodied agents, and the objectives of the negotiation. The simplified state diagram of the lifecycle of the agent is shown in Figure 1. The default state of the agent is to move independently towards its destination (x, y). Whenever an agent detects the presence of a convoy in its vicinity, it starts a negotiation process. If the negotiation is successful, the agent moves to join the convoy at a rendez-vous point. From then on, the agent moves with the convoy, until the pre-agreed leave point is reached. At the leave point, the agent leaves the convoy, and moves independently to its destination. Even while the agent is in the convoy, it might start negotiations with other convoys, or simply consider to leave the convoy on its own. If the agent wants to leave the convoy before the leave point, the agent needs to negotiate this with the convoy leader. If the negotiation is successful, the agent leaves the convoy, and it is free to follow its separate path to the destination, or join a different convoy.

Let us consider the objective of the negotiation. We will denote with $\tau_C(L_1, L_2)$ the time it takes for convoy C to move from location L_1 to location L_2. In the simplest case, at time t an agent A with the destination D_A and current location L_A considers joining a convoy C, which has a current set of commitments G. The agent has its current expected arrival time $t_A = t + \tau_A(L_A, D_A)$. In the first approximation, the agent would join the convoy if it can add to its list of commitments an agreement $B(D_A, t'_A)$ with $t'_A < t_A$, that is, it can reach its final destination faster. However, even if this agreement is not feasible, it might be worth for the agent to join the convoy up to an intermediate location P, called the *leave point*. A sufficient condition for the agent to be worth joining the convoy until leave point L_{leave} is to have a commitment $B(L_{leave}, t_{leave})$ such that $t_{leave} + \tau_A(L_{leave}, L_A) < t_A$. This is however not a necessary condition; the agent might plan ahead for joining a different convoy after leaving the current convoy at P.

A successful negotiation for an agent joining a convoy will add two commitments to the convoys set of commitments: a commitment $A(L_{join}, t_{join})$ for the join location of the agent,

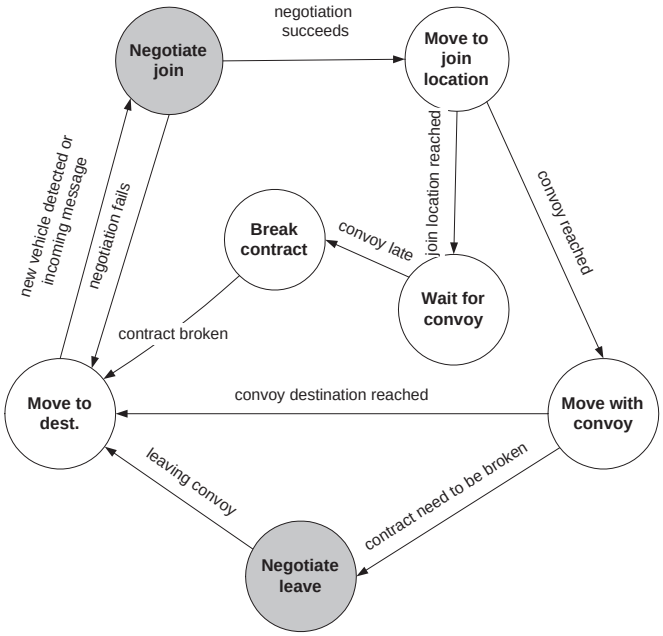


Fig. 1. The simplified lifecycle of an embodied agent which moves towards the destination optionally joining convoys. For the sake of clarity we did not represent situations such as the agent simultanously negotiating with multiple convoys for joining, or negotiating the leaving of a convoy while negotiating for joining another.

and a commitment $B(L_{leave}, t_{leave})$ for the leave location of the agent.

Thus, the negotiation between the convoy and the agent is a multi-objective negotiation, with the $\langle L_{join}, L_{leave}, t_{join}, t_{leave} \rangle$ quadruplet being the minimal negotiation set. In addition to these four objects, the negotiation might involve cost, penalties for contract breaking or other temporal constraints. In the following, we will briefly discuss the four negotiation objectives and their interrelationships.

L_{join} **the location to join the convoy.** The interest of the agent is to negotiate a join location which is as close to its current location as possible, or to be in the general direction of the destination. The choices of the convoy are (in order of preference) (a) to negotiate a join location L for which he already has an $A(J,t)$ commitment, (b) a location for which it has a $B(J,t)$ commitment, (c) a location which is on the current projected path of the convoy and (d) a location which is close to the current projected path of the convoy. Intuitively, (a) does not involve any new commitment for the convoy (if it manages to negotiate a join time earlier or the same as the previous commitment), (b) requires only a temporal commitment, without new restrictions on the path of the convoy. A location of type (c) restricts the ability of the convoy to change its path (although its current path remains valid), while a point of type (d) requires the convoy to change its path. These preferences will be inevitably reflected in the negotiation strategy of the convoy. One additional complexity is the number of negotiation choices. While the points of type (a) and (b) are coming from a limited set of discrete choices, the points (c) are coming from a one-dimensional while points (d) from a two dimensional continuum, limited only by the resolution of the raster maps on which the systems operate. This leads to an unrealistically large negotiation space. To reduce the negotiation space to a more realistic size, we chose to identify *segment locations* on the convoy path. By restricting the choice of the rendez-vous and leave locations to the segment points, we guarantee that the negotiation happens over a discrete set of choices. These locations have special properties, such as they are situated on the intersection between safe and danger zones, or on the intersection between the convoys path and the agents path. In the first pass, we can eliminate the points which are not feasible because of one of the following reasons:

(i) the convoy has already passed that segment location

(ii) the vehicle cannot reach the segment location before the convoy passes it

(iii) the segment location is unreachable by the vehicle (e.g. if it lies within a safe region surrounded by unreachable and danger zones)

t_{join} **- the joining time.** Once the agent and the convoy had identified its join location they need to negotiate the join time. In broad lines, the agent negotiates for the latest possible join time (to increase its safety margin in getting there), while the convoy for the earliest time (because that minimizes its commitment in waiting for the agent). The join time has to be at least the minimum time needed by the agent to reach the join point (the convoys minimal arrival time is not strictly relevant, as its commitment is to leave after the negotiated time). This is a simple linear negotiation, which (for all other negotiation objectives fixed) can be resolved with a monotonic concession protocol with Zeuthen strategy [3]. However, we need to observe that once the hard requirement of $\tau(L_{current}, L_{join}) < t_{join} - t_{current}$ is met, the rest of the negotiation is only about safety margins. Thus, an agent is more likely to concede in this parameter, which does not affect its predicted performance.

L_{leave} **- the leave location.** For this location, the interest of the agent is to negotiate a location as close as possible to its final destination (except the case when it is planning to join another convoy on the leave location). The interest of the convoy are, in the order of preference (a) a location for which an existing B commitment exists, (b) a location for which an existing A commitment exists, (c) a point on the current planned path and (d) a point close to the current planned path. Note that the order of preferences for types (a) and (b) is reversed for this point compared to the join point. Similarly to the join location, for types (c) and (d) we consider segmentation approaches. An additional problem which needs to be considered by the convoy is that at every leave location the resources of the convoy are diminished and at the last leave location we end up with two independent agents, not with a convoy and an agent. Thus, the interest of the convoy might be to negotiate for leave points as far down as possible on its projected path. The ideal organization is a single leave

point where all the participant agents leave for their individual destinations.

t_{leave} **- the leave time**. This parameter represents the guaranteed arrival time at the leave location. t_{leave} has a lower bound, limited by the physical time a convoy needs to reach the location on the optimal path, while still meeting its other commitments. The upper bound of this parameter is given by the limit at which it is not worth anymore for an agent to join the convoy $t_{leave}^{upper} = t_{current} + \tau_{agent}(L_{current}, L_{destination}) - \tau_{agent}(L_{leave}, L_{destination})$. Evidently, the interest of the agent is an earliest possible time - preferably the lower bound. The interest of the convoy is to minimize its commitment, by committing to as late time as possible. By accepting the lower bound, the convoy is essentially committing that it will not change its current path. This limits its ability to accommodate agents joining in the future.

B. Other cases

Besides the case when a single agent is joining a convoy, there are other possible cases, such as a convoy joining another convoy, an agent leaving a convoy or the splitting of a convoy into multiple convoys. The common denominator of these cases is that the interests of the individual agents remain the same, while the negotiation objects are variants of the single agent joins convoy case.

A convoy is joining another convoy. The complete set of commitments of the joining convoy needs to be accepted by the joined convoy.

Splitting a convoy. The splitting of convoys is worthwhile if the two resulting convoys can negotiate better terms than the larger convoy. Splitting will always happen such that both parts are at least two agent convoys. The reason for this is that if the agent would have had a better path alone than with the current convoy, it would have not joined the convoy in the first place. With four agents, however, it is possible that the agents joined in the order A_1, A_2, A_3 and A_4, but a combination of convoys (A_1, A_3) and (A_2, A_4) can offer better performance than the larger convoy of four agents.

IV. Experimental results

To test our coalition formation algorithms, we have tested them through simulating a realistic scenario based on the environment of the hurricane Katrina flooded New Orleans. The agents were implemented in the YAES simulation environment [4]. The physical environment is a 0.9x1.5 km large area of New Orleans, represented through a satellite photo with a resolution of 2 meters/pixel, obtained from Google Maps (Figure 2). The safe, unsafe and unaccessible areas were obtained partially from image processing, and partially manually edited. The scenario considers the movement of three agents from their starting points Start-1, Start-2 and Start-3 to their destination points Dest-1, Dest-2 and Dest-3 respectively. We assume that the agents are moving at the very slow speed of 1.2 km/h. The latency in preparing and delivering the messages is assumed to be 1.2 seconds, while the communication range of the agents is 100 meters - realistic for walkie-talkie type device in an urban environment.

The negotiation happens in real time and the agents keep moving while negotiating. Thus, offers can become invalid if they are answered too late, as both agents have changed their positions. To limit the time taken by the negotiation process, we introduce the negotiation limit ϵ. This limit is applied only to the part of the negotiation dealing with the temporal values t_{join} and t_{leave}; the agents have an unlimited time to negotiate join and leave locations. For this simulation, we have chosen a value of ϵ = 30 seconds, which limits the agents to at most 25 exchanged offers (since 25 offers*1.2s/offer=30 seconds).

Let us now evaluate the flow of the negotiation. As agents start moving towards their destination, their initial path goes through the path identified by note 1 on Figure 2. After traveling some distance, Agent-1 and Agent-2 come within communication range of each other and start negotiations for coordinating their movements. Table I shows the offers and counter offers made during this negotiation process. Please note that the location of the agent is shown as distance in meters from the top left corner of the map.

Offers 1 and 3 are rejected by Agent-1 because it cannot satisfy the leave constraint; the time it takes to reach the proposed join location and then move with Agent-2 as convoy does not provide any improvement with respect to Agent-1's original path. Similarly Agent-2 rejects the first offer from Agent-1. However, the fourth offer, which originated from Agent-1 for Agent-2 is accepted by Agent-2 because it satisfies both join location and leave location constraints. As stated earlier, after agreeing on join and leave locations, the negotiating parties have ϵ time to negotiate over join and leave time. The actual time for Agent-1 to reach join location $(132, 472)$ is 1.58 minutes. The time conveyed through negotiation object is $[1.58 + 0.5 =]2.08$ minutes. The negotiation space for join time for Agent-1 is $[2.08, 2.58]$. Similarly the time to reach leave location is $[48.42+0.5 =]48.92$ minutes. The negotiation space for leave time for Agent-1 is $[48.42, 48.92]$. The time it will take the Agent-2 to reach join location is 2.38 minutes. The negotiation space for Agent-2 join time is therefore $[2.38, 2.88]$ and for the leave time is $[48.44, 48.94]$. During time negotiations, Agent-2 gradually increases its offer of join time and decreases leave time. Agent-1 does the opposite thing and after some exchanges, the parties agree on the constraints $(A((132, 472), 2.38), B((1390, 390), 48.84)$. Please note that the negotiation ending criteria in this case was reaching the upper limit of Agent-1 on join time. In this particular instance, Agent-2 was able to increase join time by 0.5 minutes, but the leave time was not decreased.

So Agent-1 and Agent-2 meet at join location $(132, 472)$ after around 2.38 minutes of reaching agreement and form a convoy. They also agree that Agent-2 will leave the convoy at location $(1390, 390)$ after around 48.8 minutes. Note that the time for Agent-2 to reach its location using only the safe zones was 55.7 minutes. The time to reach the same location using the convoy is 52.3 minutes (which includes join time, convoy time and time to reach its destination from leave location).

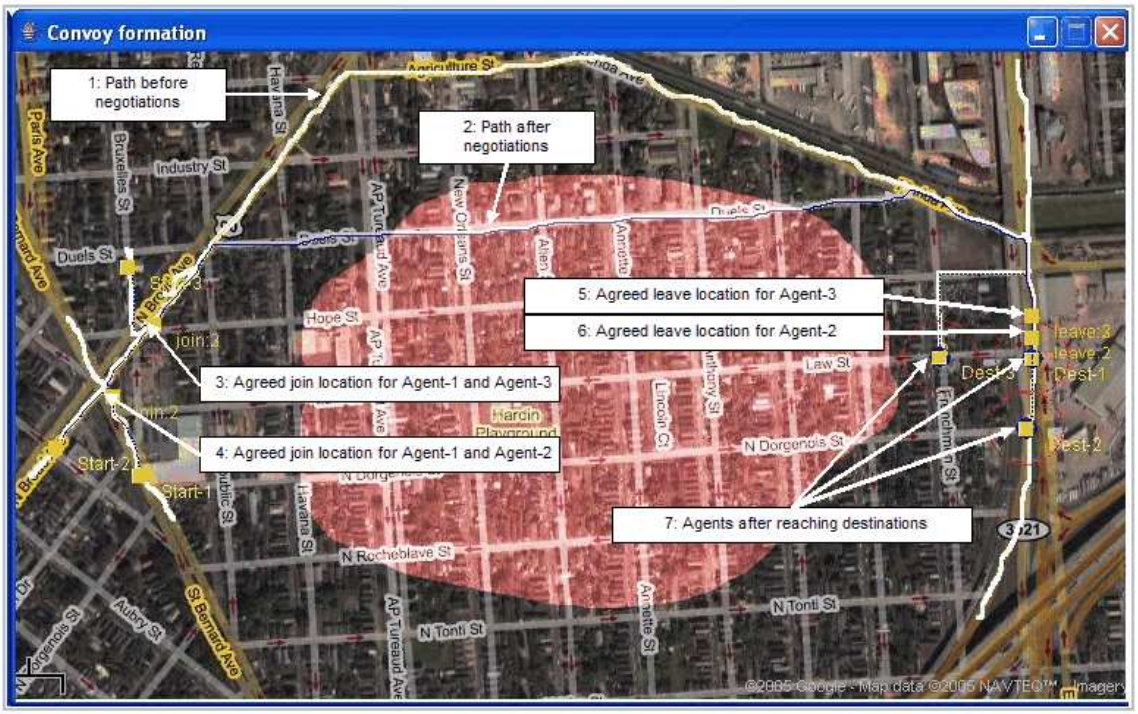


Fig. 2. An example run of the coalition formation algorithm, on a map representing an area of New Orleans flooded in the aftermath of Hurricane Katrina. The area marked in the center of the map is a danger area, which can not be traversed by individual agents, but is accessible for convoys.

The convoy then encounters Agent-3 during travel and a similar negotiation process takes place between convoy leader (i.e. Agent-1) and Agent-3. This process results in another agreement being formed between convoy and Agent-3. The join and leave locations of this agreement are also identified on Figure 2.

The convoy, now consisting of three agents, moves through the danger region and the agents leave the convoy at the agreed leave locations.

It can be seen from Figure 2 that the leave location of the agents is not an optimal location. The reason for that are two fold. First of all, the agents do not know about each other's path. So they cannot determine optimal leave location on other agent's path. Secondly, an approximation could be obtained by providing feedback to the offering party to successively decrementing the leave location. But this would require increased negotiation steps and computational time. Also, by the time we agree on the leave constraint, the join constraint would become invalid.

V. Related Work

The field of multi-agent negotiation is influenced by economic models, game theory and artificial intelligence. Jennings et. al [2] defines negotiation as a search process where multiple agents search through the negotiation space to reach agreements and discusses several negotiation models including game theoretic, heuristic based and argumentation based models. Kraus [5] provides a more in depth study of strategic negotiations in multi-agent environments.

Coalition formation between agent residing in the physical world has been the object of study of collaborative robotics. One recent effort is the DARPA Software for Distributed Robotics (SDR) program where researchers from SRI International, Stanford University, the University of Washington, and ActivMedia Robotics are designing and implementing a computational framework for the coordination of large robot teams, consisting of at least 100 small, resource limited mobile robots (CentiBOTS) on an indoor reconnaissance task. The Robocup robotic soccer challenge is also a source of research in coalition formation schemes [6]. Alami et al [7] presents a scheme of operating a large number of mobile robots using plan merging paradigm. Their scheme is based on local knowledge and incremental planning in a distributed manner. They attempt to resolve the spatial movement conflicts between mobile robots. Although we have a similar problem domain, our effort differs in that (1) we use negotiations for coordinating the movement and *2) our general goal has been to make mobile agents to agree on a meeting and leaving location rather than resolving the spatial movement conflicts.

Although team formation is frequently considered a centralized activity, where a manager assembles teams based on optimization criteria, several research efforts have dealt with negotiation based team formation models. The DARPA Autonomous Negotiating Teams (ANTS) program was one of the focus points of this effort. Examples of papers using

Offer No.	Sender	Receiver	Location of the sender $(meters)$	Join constraint $< (meters, meters), minutes >$	Leave constraint $< (meters, meters), minutes >$
1	Agent-2	Agent-1	(86.6, 501.2)	$A((122, 462), 1.93)$	$B((1390, 496), 54.26)$
2	Agent-1	Agent-2	(159, 531.2)	$A((134, 492), 1.784)$	$B((1390, 400), 49.13)$
3	Agent-2	Agent-1	(87, 500.6)	$A((132, 442), 2.46)$	$B((1390, 486), 53.84)$
4	Agent-1	Agent-2	(158.8, 530.6)	$A((132, 472), 1.98)$	$B((1390, 390), 48.842)$
5	Agent-2	Agent-1	(87.4, 499.9)	$A((132, 472), 2.18)$	$B((1390, 390), 48.742)$
6	Agent-1	Agent-2	(158.7, 529.8)	$A((132, 472), 2.08)$	$B((1390, 390), 48.842)$
7	Agent-2	Agent-1	(87.7, 499.2)	$A((132, 472), 2.28)$	$B((1390, 390), 48.742)$
8	Agent-1	Agent-2	(156.2, 529.1)	$A((132, 472), 2.18)$	$B((1390, 390), 48.842)$
9	Agent-2	Agent-1	(88.8, 498.1)	$A((132, 472), 2.38)$	$B((1390, 390), 48.742)$
10*	Agent-1	Agent-2	(157.1, 528.6)	$A((132, 472), 2.38)$	$B((1390, 390), 48.842)$

TABLE I

THE OFFERS EXCHANGED BETWEEN AGENT-1 AND AGENT-2

negotiation for agent team formation[8], [9]. Some of these papers are concerned with a multi sensor target tracking problem [10], [11]. Sariel and Balch [12] use an auction based approach for task allocation in multiple robot map exploration problem.

The CoAX - Coalition Agents Experiment series demonstrated the utility of agent technology for coalition operations in a series of technology integration experiments [13], [14].

[15] is one of the classical books on the topic of time constrained negotiation. In our negotiation model, both parties lose if an agreement can not be reached in given time. So both parties are willing to accept any offer (even non-optimal) that can satisfy the join and leave constraints.

Part of our problem domain also resembles with multi-agent meeting scheduling problem. Crawford and Veloso [16] provides a good introduction to existing work in this domain in which the focus has been mainly to make multiple agent agree on a given time slot for a meeting, under static or dynamic user preferences. Our work differs in that our mobile agents schedule for spatial locations under dynamic temporal constraints.

VI. CONCLUSIONS

This paper presented an agent-based coalition formation approach for disaster response applications. We find that many of the formal negotiation models are applicable, but the constraints of the physical world, such as temporal and spatial distances, communication constraints, as well as real-time operation requirements add new requirements. Future work include extending the coalition model to task oriented domains, development of improved algorithms for real-time operation as well as implementation on physical embodied agents.

REFERENCES

[1] P. Scerri, D. V. Pynadath, N. Schurr, A. Farinelli, S. Gandhe, and M. Tambe, "Team oriented programming and proxy agents: The next generation." in *PROMAS*, ser. Lecture Notes in Computer Science, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, Eds., vol. 3067. Springer, 2003, pp. 131–148.

[2] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge., "Automated negotiation: prospects, methods and challenges," *International Journal of Group Decision and Negotiation*, vol. 10, no. 2, pp. 199–215, 2001.

[3] F. Zeuthen, *Problems of monopoly and economic warfare*. Routledge and Sons, London, UK, 1930.

[4] L. Bölöni and D. Turgut, "YAES - a modular simulator for mobile networks," in *Proceedings of the 8-th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems MSWIM 2005*, October 2005.

[5] S. Kraus, *Strategic Negotiation in Multi-Agent Environments*, ser. Intelligent Robots and Autonomous Agents. San Francisco, CA: The MIT Press, 2001.

[6] J. Anderson, B. Tanner, and J. Baltes, "Dynamic coalition formation in robotic soccer," in *Proceedings of the AAAI-04 Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems*, San Jose, CA, July 2004. [Online]. Available: citeseer.ist.psu.edu/667681.html

[7] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and S. Qutub, "Operating a large fleet of mobile robots using the plan-merging paradigm," in *IEEE International Conference on Robotics and Automation (ICRA'97)*, 1997, pp. 2312–2317.

[8] L.-K. Soh and C. Tsatsoulis, "Allocation algorithms in dynamic negotiation-based coalition formation," in *AAMAS Workshop on Teamwork and Coalition Formation*, Jul 2002, pp. 16–23.

[9] F. Dignum, B. Dunin-Keplicz, and R. Verbrugge, "Agent theory for team formation by dialogue," in *ATAL '00: Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*. London, UK: Springer-Verlag, 2001, pp. 150–166.

[10] L.-K. Soh and C. Tsatsoulis, "Reflective negotiating agents for real-time multisensor target tracking," in *IJCAI*, 2001, pp. 1121–1127. [Online]. Available: citeseer.ist.psu.edu/soh01reflective.html

[11] R. Mailler, V. Lesser, and B. Horling, "Cooperative negotiation for soft real-time distributed resource allocation," in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press, 2003, pp. 576–583.

[12] S. Sariel and T. Balch, "Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments," in *The Twentieth National Conference on Artificial Intelligence (AAAI), Integrating Planning into Scheduling Workshop, 2005*, 2005.

[13] D. Allsopp, P. Beautement, J. Bradshaw, E. Durfee, M. Kirton, C. Knoblock, N. Suri, A. Tate, and C. Thompson, "Coalition agents experiment: Multi-agent co-operation in an international coalition setting," *Special Issue on Knowledge Systems for Coalition Operations (KSCO), IEEE Intelligent Systems*, vol. 17, no. 3, pp. 26–35, May/Jun 2002.

[14] D. Allsopp, P. Beautement, M. Kirton, A. Tate, J. Bradshaw, N. Suri, and M. Burstein, "The coalition agents experiment: Network-enabled coalition operations," *Special Issue on Network-enabled Capabilities, Journal of Defence Science*, vol. 8, no. 3, pp. 130–141, Sep 2003.

[15] S. Kraus, J. Wilkenfeld, and G. Zlotkin, "Multiagent negotiation under time constraints," *Artificial Intelligence*, vol. 75, no. 2, pp. 297–345, 1995.

[16] E. Crawford and M. Veloso, "Opportunities for learning in multi-agent meeting scheduling," in *Proceedings of the Fall AAAI Symposium: Artificial Multi-Agent Learning, 2004*, 2004, pp. 96–102.

Distributed Planning Algorithm for Coalition Logistics in Semi-trusted Environment

Martin Rehák, Michal Pěchouček and Přemysl Volf
Department of Cybernetics and Center for Applied Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic
{rehakm1,pechouc,volf}@labe.felk.cvut.cz

Abstract

We present a distrusted approach to coalition logistics planning that provides critical properties for application in adversarial and semi-trusted environment: planning and communication efficiency, well-defined levels of information sharing and confidentiality, tight integration of trustfulness with the planning and stability of the distributed plans. To achieve this goal, we combine multi-agent negotiation with efficient fuzzy and flexible linear programming techniques from operation research field. Alternating rounds of global optimisation and restricted negotiation split the task into subtasks, create teams, assign them to the tasks and provide a task-resource mapping. Resulting plan execution can be easily verified and verification results can be used to update the trust and social models and potentially to perform re-planning immediately.[1]

1 Introduction

One of the important problems of any coalition operations is a transportation logistics. Each coalition member can manage its own logistics independently, causing significant inefficiencies, resource overbooking and delays in operation. Alternatively, the logistics can be managed by cooperation among the autonomous coalition members.

We study coalition planning problem in the OOTW (Operations other than War) environment, where besides the obvious technical requirements on technical (hardware-level) interoperability and a shared knowledge model (ontology) the cooperative approach requires:

1. methods for building and using *trust models* about coalition members,
2. techniques for handling imprecise information and handling temporary *communication inaccessibility*,
3. knowledge sharing methods preventing unwanted *knowledge disclosure*,
4. *distributed reasoning* (planning) and negotiation algorithms and
5. agents ability to act in *cooperative* while also in *competitive* modes.

In this contribution we intended to address primarily the issue 1, 3, 4 and 5 from the list above. The issues of communication inaccessibility coalition planning has been discussed in [19]. The problem of transportation logistics in OOTW environment can be easily generalised so that the investigated concepts can be used in other semi-trusted, collective activity oriented domains.

The technique we suggest is not only applicable for current coalition operations, where the agents are typically materialized by manned command centers, but even more in the context of the next generation of systems with autonomous vehicles [1], where we may wish to automate not only driving and control, but also the higher level functions – making the vehicles with embedded agents responsible for transportation scheduling and road planning following the requests from the supplied entities.

Unlike classical definitions of competitive and self-interested behaviour, adversariality [18] is understood in this article in the context of a special agents' behaviour that is results in a partial loss of collective welfare of the collective of agents. Adversariality models agent harmful and malicious behaviour in the coalition. While agent's adversariality in the system can be both voluntary or intentional, often caused by coalition member with side interests or involuntary or unintentional, given e.g. by system infiltration.

In the transport logistics domain we have modeled two types of adversarial behaviour: (*i*) adversarial agent stealing the cargo and thus preventing it from being delivered and (*ii*) adversarial agent sharing the transport plan details with a third party that may or may not implement a cargo hold-up operation.

In our work, we handle adversariality by: *(i)* **limiting information disclosure** to other agents, respecting each

[1] Note to Reviewers: Current length of the submission was approved by PC. Final version will be compliant with 6-page limit.

agent's private preferences, and keeping them undisclosed, *(ii)* **integration of trust model** [17] and methods of reasoning about competitive and adversarial agents. In such an environment we were motivated by suggesting a planning solution that would **stable** - we want a stable planning solution even if a trustworthiness or availability of partner agents changes slightly and finally and **efficiency** - we want to be able to find a task decomposition and allocation within reasonable time and with a small number of messages.

The algorithm we present builds on existing multi-agent solutions for the same class of problems. Extended Contract-Net-Protocol as defined in [6] and further extended towards practical application by [14] achieves the same result using the negotiation between coalition leader and perspective members. When the perspective coalition leader wishes to solve the task, it asks other agents to cover the task completely or at least partially. Agents submit their bids, the best ones are selected and provisionally granted the task. The rest of the task is auctioned again and new auctions are organized until the whole task is covered. If the remaining task can not be covered, the algorithm must achieve consistency by backtracking – revocations of provisionally granted tasks and auctioning new ones. Even if we have a unique coalition leader, the planning problem is completely decentralized and requires intensive communication. Consequently, this approach presents performance problems when it prepares the initial plan in large state spaces – even if such planner compares favorably with humans [15, 8], it can be easily beaten by mathematical programming techniques. On the other hand, the agent approach brings more flexibility than mathematical programming as the agents may combine many sources and types of knowledge to prepare the plan, each agent contributing its knowledge, reasoning and resources. Agent's don't need to be aware of each other's mental states, provided that they are syntactically and semantically interoperable.

In this contribution, we integrate classical AI and operational research 'heavy-duty' solvers in the context of multi-agent systems. We argue that the abstract models of collaboration in agent systems as they are now used within the multi-agent system community have severe drawbacks – they are well suited for simple reasoning and limited amount of knowledge, while little scalable. Their performance tends to degrade with increasing problem complexity. Therefore, we propose that the AI/OR techniques are a very good fit for agent reasoning due to their high performance and little or no scalability problems. The traditional problems related to their application – restrictive applicability conditions (e.g. linearity) are solved by modern methods [3] and on the other side, acquaintance models [13] provide the necessary knowledge inputs for the model, as well as an efficient mechanisms for its maintenance. As the mechanism we propose is intended to function in adversarial environments, we need to augment the social model with trustfulness information, using trust and reputation models presented in [17] or other [16]. Such trust mechanism must comply with following requirements: (i) trust and reputation must be integrated with the planning mechanism, (ii) model must be robust with respect to environmental noise (natural failure), (iii) its inputs must be compatible with the observed plan outcome.

Section 2 provides the formal statement of the problem we address and describes the solution algorithm. In Section 3 we discuss some properties of the designed algorithm.

2 Algorithm Presentation

In the logistics planning problem we consider, we address the transport of goods from initial to terminal location[2] using the resources belonging to self-interested and potentially adversarial agents. Therefore, we must select appropriate routes from the plan base, combine them and allocate resources to the tasks in the plan in order to maximize the expected amount of delivered goods. In the formal problem presentation below, we present the problem from the perspective of the coalition leader – the agent denoted A_0 that organizes a coalition.

2.1 Problem Formalization

Formally, we follow the approach proposed by [20] and instead of decomposing the plan into the action-state graph, we will describe it using actions and objectives (called objects in [20]). Therefore, we define an ***abstract plan*** (e.g. route plan) as a directed bipartite graph, where one side is composed of ***objectives*** (typically corresponding to locations), defined by the set $O = \{o_0(initial), o_1, o_n(terminal)\}$, with each member defined as $o_i = (prer_{o_i}, allows_{o_i})$, where both the $prer_{o_i}$ and $allows_{o_i}$ are subsets from the Ac, while the other graph side contains ***actions*** (transports) linking the objectives, defined in the set $Ac = \{a_1, a_2, ...a_m\}$, where again $a_i = (prer_{a_i}, allows_{a_i})$ and sets $prer_{a_i}$ and $allows_{a_i}$ are subsets of O. By definition, we always start from a single *initial objective* o_0 (with no prerequisites: $prer_{o_0} = \emptyset$) and terminate in a a *terminal objective* that corresponds to the achieved goal state: $allows_{o_n} = \emptyset$. [3] Besides the structural information, we also keep Θ_{a_i} for each action – an estimate

[2] This formal simplification doesn't reduce the generality of our approach - in case of need, we may define formal zero-cost actions between the initial/terminal objective and the real terminal objective for each part of the cargo, provided that we impose appropriate restrictions on these actions.

[3] Therefore, in our graph, the nodes are defined as $Ac \cup O$, while the directed edges describe the relations expressed in $allows$ and $prer$ sets of each action or objective. We may also note that the global state of the system is defined by the state of all objectives.

of action success likelihood obtained from trust model in a same manner as individual agent trustfulness.

Batches constitute the cargo that is transported. Each batch p_i, from P is defined by its size $size(p_i)$ and *type* (liquid, bulk, etc...) that defines the resources that may carry it. By definition, all batches can be split during transport; we denote $p_i^{a_j}$ the part of the batch allocated to action a_j.

The transport problem is being solved by ***agents*** from the set $Ag = \{A_0, A_1, ...A_k\}$. Each agent A_i is characterized by its trustfulness $\Theta_{A_j}(A_i)$ as it is perceived by agent A_j, and its complement: *distrustfulness* $\Delta_{A_j}(A_i) = 1 - \Theta_{A_j}(A_i)$. Trustfulness $\Theta_{A_j}(A_i)$ is modelled as a *fuzzy number*, following [17]. This representation allows to correctly represent the uncertainty and can be used in the planning as described in Section 3 Agent is therefore modelled by leader the as a tuple $A_i = (\Theta_{A_0}(A_i), res_{A_0}(A_i))$, where the set $res_{A_0}(A_i)$ models leader's knowledge about agent's resources. Each agent controls one or more ***resources*** as defined in its set res_{A_i}. All resources, regardless of the agent they belong to belong form a set $R_{A_0} = \{r_1^{A_i}, r_2^{A_j}, r_l^{A_j}\}$, where the super index of each resource denotes the agent to which this specific resource belongs. Each resource is described by a tuple $r_i^{A_j} = (A_j, allowed_{r_i}, cap_{r_i})$, where the A_j denotes the owner agent of the resource, $allowed_{r_i}$ is a set of actions (transports) to which the resource can be assigned, and cap_{r_i} its capacity.

Tasks are a result of the planning process. They form a set $T = t_{a_1}, t_{a_2}, ..., t_{a_m}$, and each task corresponds to one action. Task is defined as $t_{a_i} = (batch_{t_{a_i}}, com_{t_{a_i}})$, where $batch_{t_{a_i}}$ is a set of batches transported in the task and $com_{t_{a_i}}$ is a set of ***commitments*** – each commitment[4] $c = (a_i, A_j, r_k^{A_j}, p_l^{a_i}, cap)$ is an assignment of a specific resource r_k (and consecutively its owner A_j) to one partial batch $p_l^{a_i}$ from the set $batch_{t_{a_i}}$ and cap determines the capacity that is to be assigned. If the r_k capacity allows it, one resource can be committed to more than one batch/action and a single partial batch $p_l^{a_i}$ can be covered by several commitments – in such case, we denote $cap(r_k^{a_i})$ the aggregate size of all commitments from the task t_{a_i} to which the resource r_k is committed. Commitments of resources relative to a single task define a ***team*** from the set $E = e_{a_1}, e_{a_2}, ..., e_{a_m}$. Each team $e_{a_i} \subset Ag$ contains all the agents contributing their resources to the task t_{a_i}. ***Coalition*** Co is defined as a union of all teams from the set E.

2.2 Public, Semi-Private and Private Information

Sharing the information about resources, plans, goals and intentions is significantly different from the cooperative agent systems. In adversarial environments, agents must seriously consider the possibility of information misuse and try to find the equilibrium between minimum information disclosure and cooperation and planning efficiency. Therefore, following [12], we define three types of information:

Public information is accessible to any agent in the system. It includes information about agent identity, existence, location and basic annotation of provided services - *type* of the resources $res_{A_0}(A_i)$ it offers, but without any information concerning their capacity, number or restrictions.

Semi-private information facilitates the planning process. It is mutually shared within groups of trusted cooperators that collaborate frequently and enables them to prepare the plans easier than by negotiating through all possible options [12]. For each agent A_i, it includes the information about its resources *aggregated* by type and including the restrictions regarding their use on the set Ac. Such compromise provides enough knowledge for the first stage of the planning process, and detailed task allocation is then finalized in course of negotiation without exposing more data than necessary[5].

Private information is reserved only to the owner agent and never shared with anyone else - it contains the detailed information about its resources, including their individual capacity, restrictions, locations and other information.

2.3 Algorithm Overview

This section provides an overview of the planning algorithm we suggest, combining the social model and linear programming planner with focused and well-targeted negotiations in the later stages of the process. The planning process proceeds as follows (see also Fig 1):

1. **Initial Planning**: Team leader uses its social knowledge and planning capabilities in order to prepare initial plan. This happens in two phases: *(i)* abstract plan construction and *(ii)* task allocation to the agents.

2. **Local Plan Evaluation**: Initial plan is evaluated by the respective agents: *(i)* the members evaluate the plan and make an attempt to trade the commitments within teams and *(ii)* proposals are sent by members back to the leader.

3. **Coherence & Verification**: Proposals are included in the detailed planning that ensures the plan coherence.

4. **Plan Execution**: Final commitments are received by members, may be swapped and the plan is executed.

2.3.1 Initial Planning

In the first phase of the plan, we assume that the coalition leader A_0 has a goal to accomplish and is obliged to form a

[4]Formally, until being evaluated and updated by bidding agents, commitments must be regarded to as mere *commitment opportunities*.

[5]Note that the sets R as perceived by various agents are not identical due to the fact that they don't have the access to the same information.

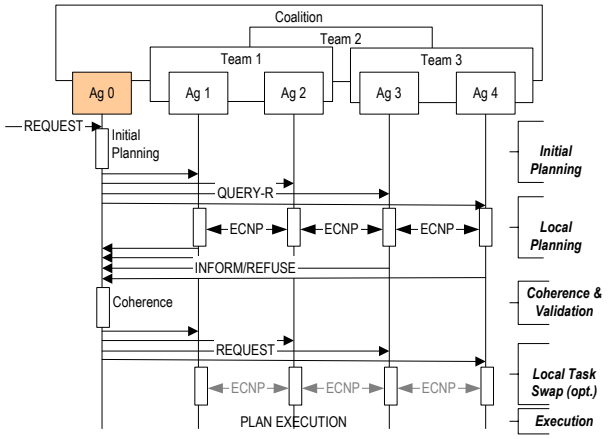


Figure 1. Overview of the protocol phases: Agent A_0 is a coalition leader and has decomposed the global task into three tasks.

coalition with other agents. It uses its social knowledge to draft a preliminary plan in the following steps.

Constructing the Abstract Plan. The first step is a preparation of the abstract plan – an action-objectives bipartite graph capturing the relationship between initial and terminal objectives (states) – typically covering alternative solutions. The graph must contain at least one path connecting the initial and terminal objective – if such path can't be identified, agent A_0 is unable to solve the planning problem.

Constructing the abstract plan is a computationally exponential problem in complex domains. Recent advancements in the field of AI planning provided very efficient techniques for constructing the plans in reasonable amount of time such as GraphPlan [10] or SAT-Plan [2]. These techniques implement a sophisticated breadth-first search based on expansion of the bipartite graph or iterative propositionalization of the planning problem.

Task Allocation. Once an acceptable abstract plan is established, leader proceeds with the **(ii)** allocation of batches and resources to individual actions in the plan, while respecting the constraints defined in the objectives. Note that for sake of computational efficiency, some actions and objectives from the abstract plan can be removed during this phase if there are no resources or batches to allocate to them. Then, we use a fuzzy linear programming (FLP) that either provides an acceptable initial task allocation T, or identifies a constraint that prevents the agent from finding the solution.

The constraints we define for the problem are the following. The first equation expresses the node equilibria - conservation of goods in each node.

$$\forall o_i \in O \setminus \{o_0, o_n\}, \forall p_j \in P : \sum_{a_k \in prer(o_i)} size(p_j^{a_k}) \cdot \Theta_{a_k} = \sum_{a_l \in allows(o_i)} size(p_j^{a_l}) \quad (1)$$

where the Θ_{a_k} represents the estimated action trustfulness (probability of completion) taken from the trust model (e.g. delivery ratio in our case) - it allows us to model the probable losses in the actions from the set $prer(o_i)$. It may range from 0 – no hope of delivery – to 1 , resulting in the same amount of resources allocated for outgoing cargo.

The initial node has a simpler relation, declaring that we can't take away more cargo than available:

$$\forall p_j \in P : size(p_j) \geq \sum_{a_l \in allows(o_0)} size(p_j^{a_l}) \quad (2)$$

while the terminal node doesn't introduce any constraint.

Furthermore, for each action a_i (elementary transport) and each batch p_j, we must ensure that the commitments cover the whole partial batch $p_j^{a_i}$ ($size(p_j^{a_i} \leq p_j)$ due to the possible parallelism):

$$\forall a_i \in Ac, \forall p_j \in P : p_l^{a_i} = \sum_{c \in com_{ta_i} : batch(c) = p_l^{a_i}} cap(c) \quad (3)$$

then, we must also make sure that no resource is used beyond its capacity:

$$\forall r_i \in R : cap(r_i) \geq \sum_{a_j \in Ac} cap(r_i^{a_j}) \quad (4)$$

besides these restrictions, we need to set-up the **utility function** for which we optimize:

$$U_m = \alpha \cdot \sum_{p_i \in P} size(p_i^{o_n}) - \beta \cdot \sum_{c_j \in C} size(c_j) \Delta_{A_j}(ag(c_j)) \quad (5)$$

, where $p_i^{o_n}$ denotes the part of the batch p_i delivered to the terminal objective and $ag(c)$ the agent committing to c.

We minimize the expected amount of the cargo lost (second sum) and we balance the cost of losses and value of delivery (first sum) by setting the constants α and β to domain-appropriate values.The ultimate goal is to allocate the resources of the coalition members to cover most of the delivery, while minimizing the risk of the attack.

Once the solution of the above FLP problem is identified (see Section 3), leader determines all perspective coalition members (owners of resources assigned to various tasks) and queries each perspective member whether it is capable and willing to participate. Therefore, each perspective coalition member A_i is sent a structure: $cma_{A_i} = (A_0, coalmem, assign)$, where A_0 is a coalition leader, set

$coalmem$ lists all coalition members and set $assign$ lists the relevant information about tasks the agent's resources are assigned to, defined as $(e_{a_j}, com_{t_{a_j}}(A_i))$, where j is an action (task) index and $com_{t_{a_j}}(A_i)$ are commitments suggested to agent A_i on task t_{a_j}.

2.3.2 Local Plan Evaluation

When the coalition members A_i (selected by the leader in the previous step) receive the coalition proposals from the leader, they must use their private knowledge to create the bid reflecting their preferences and local situation. At first, the agents must decide whether they trust the coalition leader and members sufficiently to cooperate with them, typically putting emphasis on their trust in the leader ($\Theta_{A_i}(A_0)$) and agents within the same teams ($\forall e_k : A_i \in e_k \forall A_j \in e_k \Theta_{A_i}(A_j)$). If the agent is confident enough with the coalition and proposed commitments, it will try to assign its resources to its commitments.

At this level, we handle several issues that are ignored by the leader's first-level planning – resource granularity (unknown to the planning agent due to the privacy issues) and relations between the resources assigned to different tasks. In the first round, each agent assigns its resources to the commitments that are the best fit for available resources, trying to cover all commitments. Then, it will offer the excess capacity of the resources assigned to the task t_{a_j} to all members of the team e_{a_j} using the multi-phase auction mechanism described in [6]. This step is designed to eliminate the resource allocation inefficiencies that are due to the possible leader's lack of knowledge about actual resources or a side effect of selected planning method. More formally (see also Fig. 2), to start the the negotiations, each agent A_i working on task t_{a_j} broadcasts a CFP message containing its free capacity to all team $e_{t_{a_j}}$. If the other team members are interested in using this capacity for the task they were allocated, they submit their bid. Agent A_i selects one or more bids and answers them with a temporary grant, making them binding *for the bidders*; other are temporarily refused. When the agent A_i participates in several teams, it can now reshuffle its resources between the tasks to use them in an optimal manner. Once the resource reallocation is terminated, all compatible temporary grants are confirmed, while the others may be refused (In case we the agent has replaced the original resource with a lower-capacity one.). If appropriate, agent can now offer the new free capacity for trading using the same protocol.

Note that the auctioning and negotiation takes place only within the single task team, therefore minimizing the knowledge dispersion and communication load. On the other hand, agents may therefore miss a better task allocation. Once the negotiation is finished, all team members send their answers to the coalition leader. The answer is a

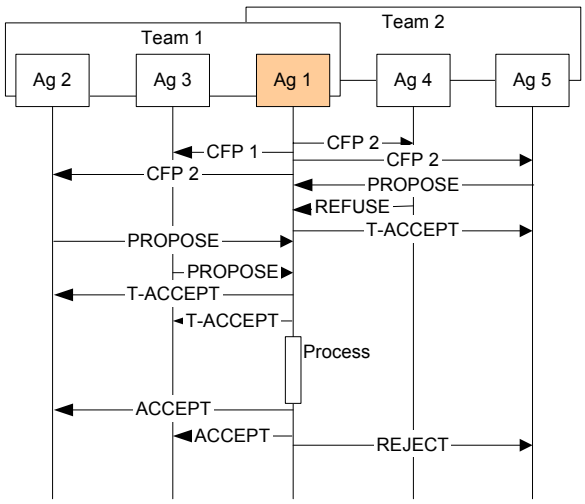


Figure 2. Use of the ECNP to allocate agent's resources across two different teams. Agent A_1 first temporarily accepts the offer from A_5, but later on finds a better resource allocation and prefers to commit larger resource to *team 1*. Therefore, it rejects the bid from A_5.

list of commitments that are actually *binding* for each agent, but may differ from those originally assigned to the agent as: **(i)** the agent is not always able to cover the whole assigned commitment and commits only to a part of the original commitment or **(ii)** it notifies the coalition leader about the transfer of the whole commitment or its part to other coalition member (this member lists this commitment in its turn as covered). When the agents submit their binding commitments to the coalition leader, they have an alternative to offer the free capacity of the resources they've allocated to the task to the coalition leader - the leader may include use it to cover other batches from the same task, as specified by relation 6. While this remains an attractive optimization feature, this approach has two major drawbacks – the leader can easily guess the capacity of agent's resources and the free resources can not be used on another task.

2.3.3 Coherence & Verification Phase

In this phase, coalition leader receives the answers from the coalition members and must re-combine them into a globally coherent plan. As the initial planning has produced a coherent plan, the plan is coherent when all proposed commitments were covered by members. If not, the leader must add all updated commitments/refusals from the agents to the initial plan and perform the new calculation to make sure

that the condition 1 is valid for the final plan.

Updated commitments are included as follows (refusals or previously unassigned commitments are considered as commitments with 0 capacity):

$$\forall a_i \in Ac, \forall r_j \in R : cap(r_j^{a_i})^{prop} \geq cap(r_j^{a_i})^{final} \quad (6)$$

It is at this stage of planning process when we also detect the failure to execute the plan altogether – the proposals (or actually refusals) from the members may be mutually incompatible. If the coalition leader manages to find an acceptable planning outcome, it prepares the final commitments (with the quantities assigned that are less or equal to the binding ones proposed by members) and re-submits them to the coalition members.

2.3.4 Plan Execution

As the proposals by the agents were binding, coalition members shall be all able to start performing the assigned tasks immediately. Alternatively, when the final commitments are lower than the ones they have proposed, they may change their resource allocation or trade the assignments with their peers in the team in the same way as in the Local Plan Evaluation phase, provided that they manage to honor their commitments.

3 Algorithm properties

In this section, we will analyze the above-described algorithm and discuss several interesting properties it presents: computational efficiency, preservation of private information and stability of the solution with respect to environmental perturbations.

Reduced Communication. The present algorithm represent a special approach to distributed planning that in parts uses the classical AI planning algorithms in combination with multi-agent, negotiation based approach to plan generation. Unlike state-of-the-art approaches such as *Partial Global Planning* [5] the *initial planning phase* of the presented approach substantially constrains the space of possible negotiation and thus makes the planning process substantially more scalable. On the hand it still allows the agents to influence the plan that is to be imposed on their operation by task delegation enabled by ECNP protocol during the *local plan evaluation phase*.

Obviously, the desired situation is that the amounts of decision making and computation is somewhat balanced between these two phases of the planning algorithm. Finding this equilibrium affects the efficiency, stability and the amount of private knowledge disclosure during the planning process (as explained in Section 1). This is why the right amount social knowledge stored by the perspective coalition leader (depends on the amount of knowledge the agents are happy to share) affects the right fitting the operation of the algorithm. With an increasing amount of knowledge shared within the community (therefore higher knowledge disclosure), the initial planning phase is getting more precise resulting in substantially less communication traffic (and thus further knowledge disclosure) in the local plan evaluation phase. It has been studied recently how the amount and structure of a priori shared social knowledge affects the coalition formation process [11].

Operation-research integration with negotiation and cognitive methods is natural and seamless: output of most trust models in existence today can be transformed into the fuzzy number-form and this representation fits well into the context of modern FLP methods as shown below. On the other hand, the individual team members don't need any notion of FLP techniques - they only reason about the coalition, their teams and negotiate within their teams to achieve optimal resource allocation.

Contraction of the solution space is another key feature – each step of the planning, centralized or distributed, reduces the solution space. Initial planning performs the greatest reduction, as the actions/tasks are selected, resources preallocated and agent teams created. Local planning phase then further clarifies resource allocation and team composition and the results of this phase are incorporated as additional restrictions for the FLP planning problem solved in the coherence and validation phase – we effectively ensure that any overall solution will respect the commitments received from coalition members and can be executed. Optional re-allocation step doesn't break our assumptions, it only permits team members to trade resources in a situation where the batch sizes may have been reduced. If the plan can not be implemented due to the member refusal or resource incompatibility, the situation is detected in the coherence planning step. LP method used identifies the interfering restriction and can direct the coalition leader towards plan reconfiguration. Therefore, in the global algorithm as suggested, we don't allow any backtracking (except the team-scale negotiation).

On the other hand, the algorithm as presented doesn't guarantee that the result it returns will be the optimal plan. We don't consider this as a serious drawback, because none of the comparably efficient algorithms currently in use can guarantee such result.

Stability of Flexible & Fuzzy Linear Programming One of the important properties of the trustfulness $\Theta_{A_0}(A_i)$ (and distrustfulness $\Delta_{A_0}(A_i)$) values is their uncertainty,

emphasized by the fact that they are modelled as fuzzy numbers. There are two approaches how to use these values in the LP algorithms: either to solve a *flexible linear programming* problem, or to *defuzzyfy* the values and solve a classical LP problem.

Flexible linear programming techniques [3] that work with fuzzy coefficients provide us with a unique feature - a stability of the solution with respect to small changes of coefficient (e.g. trustfulness) values defined as symmetrical triangular fuzzy numbers. Problem formulation remains the same, but we must solve a non-linear optimization problem in order to obtain the solution – a major disadvantage of the approach. On the the other hand, once we have an appropriate solver, we may effectively adjust the stability of the solution by varying the width of the trustfulness values – by restricting their width, we approach the unstable classical linear programming problem, while the widening of trustfulness representation ensures the stability with respect to bigger perturbations. This ability is a very desirable feature when the agents encounter an intelligent adversary in an unknown environment – agents can adjust their planning to be robust when they still gather the information about the environment and reduce the predictability of their behavior in later phases. The shape representing the trustfulness $\Theta_{A_0}(A_i)$ supports this adaptation, as it "narrows" with the increasing number of data.

In the alternative approach, $\Theta_{A_0}(A_i)$ and $\Delta_{A_0}(A_i)$ must be defuzzyfied before they are inserted into the planning constraints of the normal LP problem. Defuzzyfication to use depends on the definition of $\leq$ relation between fuzzy numbers – as we follow the FLP $\rightarrow$ LP transformation method method detailed in [3] we use the center of gravity to compare two fuzzy numbers. This relation between fuzzy numbers returns a *crisp* output. During the transformation, we replace the fuzzy numbers in the constraints and utility function by the center of the core of the trustfulness values and solve the resulting problem.

We may also defuzzyfy the $\Theta_{A_0}(A_i)$ and $\Delta_{A_0}(A_i)$ using the center of the core method - this approach is more sensitive to the noise, especially with limited number of data, but as the agent gathers more experience, it converges to the center of gravity method due to the shape of the trustfulness function $\Theta_{A_0}(A_i)$ as defined in [17].

4 Prototype Deployment

Presented protocol is currently being integrated with ACROSS scenario (see Fig. 3) to solve humanitarian aid logistics problem on a simplified simulated version of Java island in Indonesia. We have taken existing system based on generic multi-agent approach to coalition planning and introduced a new agent with planning capability. This agent simulates a humanitarian organization that organizes a delivery of goods from the initial location (Jakarta) to several cities throughout the island.

Figure 3. Start of the operation as shown in 3D simulation – real batch positions and operations are shown for the selected plan.

This agent (denoted HumRed) assumes the role of a coalition leader (A_0). Other agents, who actually provide their vehicles (resources) for the transportation are coalition members and don't use any FLP solvers, they merely assign their resources to tasks suggested by coalition leader. Therefore, coalition leader implements *initial planning* and *coherence and validation phase* (Fig 4).

While the overall algorithm is already integrated with ACROSS environment, we are currently proceeding with implementation of its more advanced features (most notably ECNP negotiations within teams and more advanced FLP solvers) to improve the quality of the resulting plan. Advancing the implementation state of these features would also allow us to formally verify the protocol and judge the relative impact of its various features and phases on plan quality.

5 Conclusions and Future Work

In our contribution, we have presented a combined planning algorithm that can be used to efficiently create a shared plan in an adversarial environment, featuring only a limited and controlled information disclosure by self-interested agents. Adversarial behavior of agents and environmental reasons of failure (actions with low action trustfulness) can be detected and provide an input for the embedded trust model, that in its turn provides an input for further planning.

One of the important open issues of this research is the concept of *plan diagnosis*. Plan diagnosis [20] is important

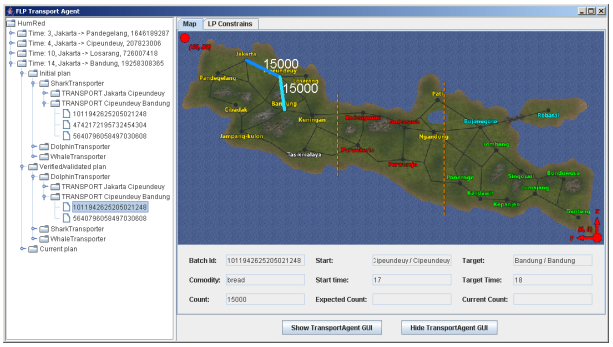


Figure 4. Internal planning process of the coalition leader – initial plan and coherent plan based on input from coalition members.

to achieve long-term efficiency by elimination of untrustful cooperators and bad actions (paths). We can not assume that the state of the problem is observable – implicitly, we assume that only the initial and terminal objective status are known by the coalition leader and that the state of some of the intermediary objectives **may** be known. Integration of the latest results from the monitoring selectivity problem [9] into a trust model update is a challenging problem for future research.

Another key challenge for the future research in this area is to investigate the plan execution phase and allow *intelligent replanning*. Re-planning can occur as a result of a coalition leader detects a failure in completion of one or more commitments or upon a request from the coalition members. If these commitments fully or partially precondition other tasks (and their commitments), it may be advantageous to re-plan the plan in order to eliminate inefficient future commitments. Such operation is analogous to coherence phase (see Section 2.3.3), but for each commitment with known outcome, we can fix the commitment size to the real delivered value, or limit it by using the information about partial deliveries/losses. In the same manner, the agent can react when some task was more successful than expected and more resources are necessary for subsequent transport. Integration of the classical work on joint commitments [4] and shared plans [7] as well as various penalty mechanisms allowing agents to deliberate on dropping the commitments is an important component of the future research in this area.

Acknowledgment

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-04-1-3044. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon[6].

References

[1] D. G. C. 2005. Darpa grand challenge 2005. `http://www.darpa.mil/GRANDCHALLENGE/`, 2005.

[2] M. Baioletti, S. Marcugini, and A. Milani. C-satplan: a satplan-based tool for planning with constraints. In *Proceedings of AIPS-98 Workshop on Planning as Combinatorial Search*, 1998.

[3] C. Carlsson and R. Fullér. *Fuzzy Reasoning in Decision Making and Optimization*. Physica Verlag, Springer, Heidelberg, 2002.

[4] P. Cohen and H. Levesque. Intention is a choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[5] E. H. Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchi- cal behavior space. *IEEE Transactions on Systems, Man, and Cybernetics – Special Issue on Distributed Artifcial Intel- ligence*, 21(6):1363 – 1378, November 1991.

[6] K. Fischer, J. P. Muller, M. Pischel, and D. Schier. A model for cooperative transportation scheduling. In *Proceedings of the First International Conference on Multiagent Systems.*, pages 109–116, Menlo park, California, June 1995. AAAI Press / MIT Press.

[7] B. Grosz and S. Kraus. The evolution of SharedPlans. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agencies*, pages 227–262. Kluwer Academic Press, 1999.

[8] J. Himoff, P. Skobelev, and M. Wooldridge. Magenta technology: multi-agent systems for industrial logistics. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 60–66, New York, NY, USA, 2005. ACM Press.

[9] G. A. Kaminka and M. Tambe. Robust multi-agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research*, 12:105–147, 2000.

[10] I. Miguel, P. Jarvis, and Q. Shen. Flexible graphplan. In W. Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence*, pages 506–510, 2000.

[11] M. Pechoucek, V. Marik, and J. Barta. Role of acquaintance models in agents private and semi-private. *Knowledge Based Systems*, accepted to publication, to appear Spring 2006.

[12] M. Pěchouček, V. Mařík, and J. Bárta. A knowledge-based approach to coalition formation. *IEEE Intelligent Systems*, 17(3):17–25, 2002.

[13] M. Pěchouček, V. Mařík, and O. Štěpánková. Role of acquaintance models in agent-based production planning systems. In M. Klusch and L. Kerschberg, editors, *Cooperative Infromation Agents IV - LNAI No. 1860*, pages 179–190, Heidelberg, July 2000. Springer-Verlag, Heidelberg.

[6]The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

[14] D. Perugini, D. Lambert, L. Sterling, and A. Pearce. A distributed agent approach to global transportation scheduling. In *The 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003)*, pages 18–24, Halifax, Canada, 2003.

[15] D. Perugini, D. Lambert, L. Sterling, and A. Pearce. Agent-based global transportation scheduling in military logistics. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1278–1279, Washington, DC, USA, 2004. IEEE Computer Society.

[16] S. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multiagent systems. *The Knowledge Engineering Review*, 19(1), 2004.

[17] M. Rehák, Lukáš Foltýn, M. Pěchouček, and P. Benda. Trust model for open ubiquitous agent systems. In *Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference*, number PR2416 in IEEE, 2005.

[18] M. Rehák, M. Pěchouček, and J. Tožička. Adversarial behavior in multi-agent systems. In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications IV: 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005*, number 3690 in LNCS, LNAI, 2005.

[19] D. Šišlák, M. Pěchouček, M. Rehák, J. Tožička, and P. Benda. Solving inaccessibility in multi-agent systems by mobile middle-agents. *Multiagent and Grid Systems*, 1(2):73–87, 2005.

[20] C. Witteveen, N. Roos, R. van der Krogt, and M. de Weerdt. Diagnosis of single and multi-agent plans. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 805–812, New York, NY, USA, 2005. ACM Press.

Reflective-Cognitive Architecture: From Abstract Concept to Self-Adapting Agent

Lukáš Foltýn[1], Jan Tožička[1], Milan Rollo[2], Michal Pěchouček[1], Pavel Jisl[1]
Gerstner Laboratory[1] and Center for Applied Cybernetics[2]
Department of Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic
{lfoltyn|tozicka|rollo|pechouc|jisl}@labe.felk.cvut.cz

Abstract

Multi-agent systems penetrate into interesting domains of computing with limited resources, programme code reusability and automated code generation. This paper presents general framework of Reflective-Cognitive agent architecture which enables the agent to alter its own code in runtime according to the changes in the environment. We also present results of architecture implementation showing the plausibility of created prototype.

1. Introduction

Multi-agent system consists of a number of agents running on a variety of host computers or other devices. System structure may dynamically change, agents may migrate among the hosts with diverse computational resources and even the parameters of the environment the agents operate in are usually not constant during the system's whole lifecycle. Agents thus have to adapt to these changes to keep the system functional.

There are two ways how to deal with this problem: (i) either the developer may take into account possible changes of the environment during the design time or (ii) reflective architecture that allows agent to adapt automatically can be used. This paper presents the second case - general framework of Reflective-Cognitive agent architecture which enables the agent to alter its own code in runtime. This architecture allows creation of lightweight agents that are able to adapt event to the situations that were not known during the design time.

Classically, in the software engineering we distinguish two types of reflection: **structural** [1, 2] and **behavioral** [5] reflection. In our work we adopt the interpretation published in recent contribution [7]: using the *Structural Reflection* the system structure can be dynamically modified, while *Computational (Behavioral) Reflection* modifies the system semantics (behavior). Examples of the respective cases provided by the authors are data structure modification and algorithm modification.

Applying the above definitions to the multi-agent reflection, we use structural reflection to change agent's private data – its state, acquaintance models, etc. The behavioral reflection is used to change the agent's reasoning algorithms – a part of the agent that interprets its data. The structural reflection can change agent's behavior in only limited way as all possible patterns of behavior have to be already implemented in agent code. The behavioral reflection implements an introspective force that is able to change the behavior of the agent in the most general way – on the code level, where we can compose or generate new algorithms incorporate them as agent's future behavior.

The paper is organized as follows. In Section 2 we present definitions of reflection in multi-agent systems, motivation for using reflection and possible component approach to the reflection. In Section 3 we present proposed Reflective-Cognitive agent architecture. Section 4 is devoted to the specific task of the reflective process – the creation of new algorithm. In the Section 5 we present an illustrative experiment with RC agent prototype within complex scenario.

2 Reflection in Multi-Agent Systems

A multi-agent reflection can be viewed on a macro- and micro- levels. This is why we distinguish between three different *types of reflection* in multi-agent systems. (i) **Individual Reflection** - revision of agents' isolated behavior, that does not necessarily need to result from agents' mutual interaction. Individual reflection is an operation that works with agent's awareness of its own knowledge, resources, and computational capacity. (ii) **Mutual Reflection** - revision of agent's interaction with another agent. This kind of revision is based on agent's knowledge about the other agent, trust and reputation, knowledge about the other

agent's available resources, possibly opponent's (or collaborator) longer term motivations and intentions (all these kinds of knowledge are refereed to as *social knowledge* and are stored in agent's acquaintance models). (iii) **Collective Reflection** - revision of agents' collective interaction. This is the most complex kind of reflection. Here we assume revision of the collective behavior of the group of agents as a result from their complex interaction. Collective reflection can be achieved either by: (a) a **single reflective agent** (e.g. a meta-agent) that is busy with monitoring the community behavior and updating agents' behavior or (b) **emergently** by the collective of agents, each carrying out its specific cognitive/reflective reasoning. In the collective reflection the agents update not only their social knowledge bases and reasoning processes but also they make the attempts to revise other agents' acquaintance models and possibly reasoning (unlike in the case of mutual reflection).

2.1 Motivation of Reflection Deployment

We will concentrate on the notion of autonomous adaptation using the reflection process described above, as this notion is critical for future open ubiquitous (pervasive, ambient) ad-hoc systems. Once we deploy the diverse elements of these systems, they must be able to integrate themselves into the functional organism and to maintain themselves operational even in a long-term perspective. Autonomous adaptation to the changing environment is critical, as it will significantly increase the usability of ubiquitous systems by (i) extending the system lifespan by increasing collaboration efficiency, (ii) extending the average lifespan as system will remain operational even after significant environmental or device changes, (iii) limiting or minimizing the human maintenance operations, (iv) enabling an easy extension of system functionality using the collective reflection process or (v) facilitating the transfer of the knowledge from the existing system into the new one during the replacement phase.

While we have specifically addressed the embedded, highly distributed multi-agent systems in the overview above, most points apply to all types of multi-agent systems.

Our motivation is to create an architecture which is able to adapt to the limited computational resources and changes in the environment by means of runtime code alternation, to share and incorporate previously unknown code and to keep this architecture as lightweight as possible including simple code development and deployment.

2.2 Component Approach

One of the goals of the architecture is to introduce a new approach how to implement agent's algorithms. We have decided to shatter agent's algorithms into smaller parts we refer to as *components*. The code splitting of the algorithm is done with respect to component's functionality (and reuse) or with respect to the data flow in the whole algorithm. Each component is a self-contained java class. Properly chained components form an algorithm which we call a *plan*. To build the plan, Reflective Layer (see below) uses the components as *black boxes* described by their inputs, outputs and meta-data. Plans can be triggered by an external or internal event and they implement agent's reaction to the event.

Advantages of component approach are as follows. (i) **Programme Code Sharing** - the code which is shared across several algorithms can be concentrated in a single component. (ii) **Programme Code Assembly** - using components to assembly algorithm gives us a great portion of freedom when trying to adjust the algorithm to the current state of the environment. Building blocks in the form of components are easy to understand and to manipulate. (iii) **Programme Code Exchange** - smaller portions of the code in the form of components are easier to exchange among agents than complex algorithms. Exchange of components among the agents allows a great deal of flexibility and at the same time the communication links are less loaded. Each agent can decide which components to integrate and combine them with its local code-base. (iv) **Programme Code Alternation** - the introduction of a single new component can result in alternation of multiple algorithms so that they can reflect new situations in the environment. Altering the code on the level of components is far more efficient than on the level of the whole algorithms. This concept is close to Aspect Oriented Programming (AOP) [4]. (v) **Algorithm Efficiency Improvement** - using component metrics provided by the Cognitive Module (see below), we can detect bottleneck of the algorithm and initiate a negotiation to find more efficient substitution. Considering component to be a black box and swapping it with component of the same functionality, enables easy manipulation with the algorithm code.

There is one *drawback* of the component-based approach. **Component Running Costs** – the principal disadvantage of the component approach is the cost of processing related to the data exchange among components. In order to minimize these costs, we concentrate on efficiency improvement of this code.

3. Reflective-Cognitive Agent Architecture

In a layered structure of the Reflective-Cognitive (RC) agent, the classical agent architecture forms the lower half of the structure – the *Reasoning Layer*. On the top of this layer, we add the *Reflective-Cognitive (RC) Layer*. The whole structure is shown in Figure 1.

Reasoning Layer handles regular agent operation – it performs agent-specific actions (interaction with agent-

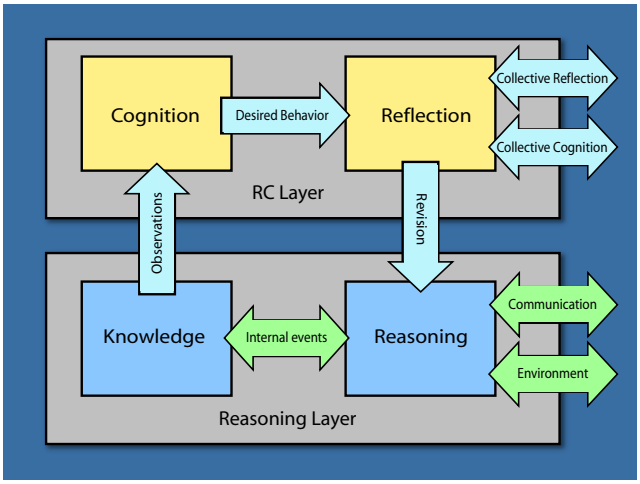


Figure 1. Reflective-Cognitive abstract architecture

specific technical resources, environment sensing), social interactions and finally planning and resource management.

To implement the reflective behavior, the **RC Layer** uses two main modules – *Cognitive Module*, that maintains the model of the agent in its environment and identifies the possible adjustments, and a *Reflective Module* that performs proposed modifications on the level of plans and components and delegates them to the Reasoning Layer.

3.1. Reasoning Layer

Reasoning Layer doesn't provide any reflective functionality. It stores and executes all agent's algorithms.

Algorithm instances (called sequences) are executed in a reaction to one of the following event types: (i) **environment event** which is invoked by the change of the surrounding environment, (ii) **internal event** which is invoked by the change of agent's internal state and (iii) **inter-agent communication** which is invoked by reception of a message from another agent.

The agent can also receive messages that it is not able to handle in an actual configuration. There can be two reasons for that: (i) RC Layer knows the appropriate processing algorithm, but decided not to load it into the Reasoning Layer, or (ii) appropriate algorithm is not present even in the RC Layer. In both cases, agent reacts to such messages by sending the *NOT-UNDERSTOOD* reply. Reflective process can later decide to incorporate the appropriate component or new algorithm (in the case (i)) or to query the other agents for appropriate components (case (ii)).

3.1.1 Knowledge Module

Components and plans need to share data which reflect agent's or environment's state. This is achieved using a structure called Knowledge Module. **Knowledge Module** is in our case implemented using *blackboard architecture*. Shared knowledge, stored and identified under *keys*, is accessible to all components in all algorithms. Knowledge alternation generates event which could be used for plan triggering.

3.1.2 Reasoning Module

The **Reasoning Module** is responsible for agent's functionality. Within the Reasoning Module all algorithms are maintained. It stores all used *plans*, all *components* used within these plans and all *sequences* (instances of running plans) – see Figure 2.

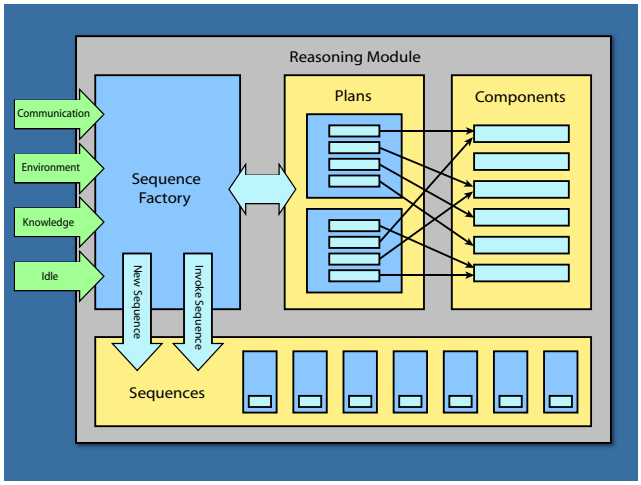


Figure 2. Reasoning Module – block scheme

After the event occurrence Sequence Factory selects appropriate plan from **Plans** storage and launches new **Sequence** object that manages execution of the plan.

From the technical point of view *component* is a piece of code, accompanied by *meta-description*, which could be externally executed and which typically produces well-defined outputs. Once triggered, the computational part of the component is executed. Usually code execution causes knowledge update or starts or continues in some message conversation.

While processing the plan, the sequence step by step invokes components, waits for their completion and provides them with a reference to their shared, plan-instance local data structure and the global blackboard. Therefore, local data structure modifications don't affect other plan instances. According to the output returned by the component after its completion, the next component from the plan is selected or the plan execution is finished. Each sequence runs in own thread.

3.2. Reflective-Cognitive Layer

The RC Layer is responsible for the reflective features of the architecture. It consists of two parts – *Cognitive Module*

and *Reflective Module*. The function of RC Layer is optimization of agent's behavior and its priority is low. Thus it is executed only in agent's idle time with relatively low probability.

3.2.1 Cognitive Module

The architecture of **Cognitive Module** is based on meta-agent abstract architecture [9]. It has two principal functions. It maintains *model* and it runs *meta-reasoning process* working on the model data.

The *model* contains the knowledge about the agent's environment, social neighborhood and the agent itself. The knowledge included in the model is less specific than the knowledge on the object level and it tends to represent historical experience rather than the current state only.

The goal of the cognition is to identify significant changes of the environment and to evaluate how effective/successfull are the actual algorithms using *meta-reasoning*. From the individual reflection's point of view various metrics can be used e.g. metrics based on the system load (the processor load or the memory consumed), agent's overall success (the ratio of won auctions), etc.

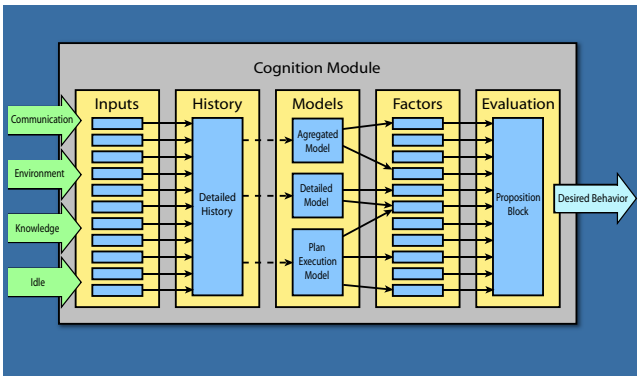


Figure 3. Cognitive Module – block scheme

Block Scheme of the Cognitive Module - The input data for the model in the cognition are gained by observation of the Reasoning Layer. To obtain fresh data from the Reasoning Layer, the cognition registers knowledge listener called **Inputs** within the blackboard. They update (asynchronously with the whole RC Layer) **History** component by data elements we call *Flags*. The information contained in the flag data part can be a very coarse digest of data from the knowledge. Each *Input* could be registered as a multiple event and knowledge listener and it could be very complex when transforming the knowledge. *Flags* are recorded in exact order as they have arrived thus causality of events is preserved. Once the RC process is triggered, **Models** are updated. With respect to data grouping we can distinguish two types of models - *Detailed* and *Aggregated*. In the *detailed* model we explicitly care about the order of events as they appeared. This gives us possibility to analyze causality of events in the system. In *aggregated* model we care about how many times certain event appeared between two dividing flags, thus providing us with pre-sorted data for statistical analysis. *Plan Execution Model* is special model which gathers data about plans' execution. This is used for plan retirement analysis. All *Models* are refreshed after the RC process was triggered. The models in the cognition contain information which is used by meta-reasoning to judge how good the overall behavior of the agent is. To do so, data from models is mapped to a set of real value attributes - **Factors** - which reflect various aspects of the environment including agent's behavior (e.g. successful delivery ratio or the current system load). For each component, a weight vector for cognition Factors as a part of meta-description is given. The weight value tells how strongly the component is affected by the factor. When an overall suitability of the plan is computed in **Evaluation** - *Proposition Block*, the weight vector for each algorithm component is multiplied by current value of factor. When certain algorithm gives low score of overall suitability, it is marked for reflection as a candidate for redesigning.

3.2.2 Reflective Module

Reflective Module has multiple functions. Its block structure is shown in Figure 4.

Reflective Action Identification block is responsible for handling of the behavior improvement specification received from the Cognitive Module (such as sudden system load drop notification) and execution of the appropriate block that can handle this specification. In general these blocks can be divided into three categories: (i) incorporation of default plans, (ii) replanning of actual plans and (iii) component generation. When the reflective process is invoked for the first time, *default plans* are loaded. They enable agent to operate from the beginning of its life-cycle (as all agent's algorithms are in the form of plans). Efficiency of default algorithms can be later improved by the reflective process.

If some plan is recognized by the Cognitive Module as not suitable for some reason, the request for *programme reconfiguration* is sent to the Reflective Module. Plan reconfiguration can be achieved either by replacement of one of the components within the plan or by modification of the whole plan. Besides planning sequences of appropriate components the Reflective Module can also be used for creation of new components – *code generation*, based on the observed events in the environment (see separate Chapter 4). Altered and new plans are passed to the Sequence Factory in the Reasoning Layer.

Plan Composition Approaches - When a Reflective Layer composes a new algorithm as a specific *event occurrence* handling (event-driven behavior), the forward plan-

ning algorithm will be used. If agent wants to reach a *new goal* (goal-driven behavior), backward planning algorithm is used. The planner searches the space of possible plans and selects the best one using the given utility function. Components within the plan are chained according to the rules stated in their meta-description.

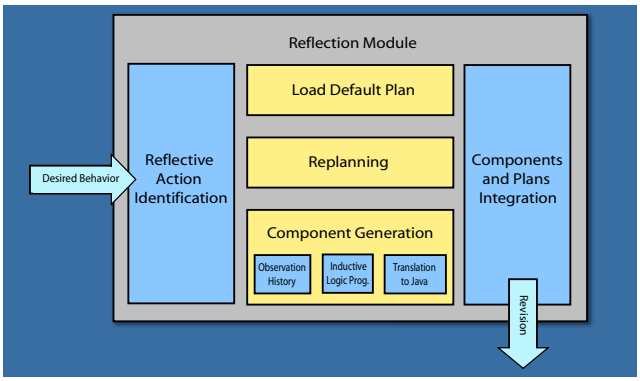


Figure 4. Reflective Module – block scheme

Algorithm Deactivation - in case of low rate or utility of a component or a plan, the cognition can suggest the reflection to remove it from all plans or to remove the whole plan. Component can be removed from the Reasoning Layer once the last running instance is completed. Note that the component code remains in the Reflective Layer and can be later incorporated in the Reasoning Layer again.

4. New Component Creation

The replanning can change the behavior of an RC agent by combining different components and thus adapt it to the dynamic environment. This approach covers a lot of different situations and can lead to the plans that have not been considered by system designer. Nevertheless, in some situations the agent does not know any component that could help it to solve some problem. In these situations, the agent can create new component. This process is invoked during the reflective process and can be very time consuming.

In our domain, we have implemented a component creation based on machine learning method ILP. Learned model is then translated into Java language and encapsulated into a component. In this prototype, we know what are the necessary inputs for this component, what type of knowledge this component produces and how to place this component into the created plan. All of these parameters are hardcoded into component creation module. In the future, we plan that component creation module should find out these properties autonomously. This will also allow the agent to create different types of components using the same module.

4.1. ILP

The subclass of machine learning methods known as *inductive logic programming* has been described e.g. in [6]. Briefly, ILP is a machine learning technique that can create a theory by generalizing given specific positive and negative examples. Created theory is common Prolog program, i.e. list of Horn rules.

Our principal finding is that it is feasible to implement the learning algorithm given the interpreted-Prolog (*tuProlog* interpreter [3] in our case) circumstance, resulting in the induction run-times peaking at tens of seconds on state-of-the-art hardware. An important question is however risen as to the scalability of this approach for future extensions of the problem setting (domain background knowledge, training data volumes, etc).

4.2. Prolog-to-Java Translation

As described above, risk evaluating rules created by ILP follow the syntax of Prolog programming language. In order to use these rules during the agent run without the necessity to run Prolog interpreter, we need to translate them into the Java language. Naturally, the Prolog rules are translated into Java *IF-THEN* statements. Once we have translated the rules into Java language it is necessary to encapsulate created code into a component. Newly created component is then incorporated into the agent's Plan knowledge base.

5. Experiments and Results

In this section we will demonstrate the plausibility of proposed architecture of reflective-cognitive agent. Presented experiments are just a small subpart the whole RC project and are intentionally selected to be simple and illustrative.

5.1. Case Description

The whole RC architecture including component creation has been implemented within ACROSS scenario [8]. In this scenario we solve a logistics problem in a competitive environment with self-interested agents. Java island is populated by location agents (cities with population) and transporter agents. Goods are created and consumed by locations and the surpluses are subject of trade among locations and delivered by transporters. Furthermore there are also several bandit agents that holdup loaded transporters. These bandits have their private restrictions where they will or will not rob the transporters they meet. Unlike common agent, thanks to its RC architecture, the transporter agent can create new component, that will predict the risk of bandit action (based on stored journeys history and bandit actions). Incorporating this component into its plans agent can reduce the risk of being robbed.

5.2. Experiment Setup

For our experiment we have run 18 cities spread out on the island and 6 bandits holding up the cargo transported by 10 transporter agents. Both the bandits and the transporters have been implemented using RC architecture. In addition, the transporters have been allowed to use ILP in order to create new component predicting the risk of hold-ups. Bandit have used a deterministic rule that specifies where the shall hold up the cargo.

The simulation has started without bandits. It has run 10 simulation days in order to reach some stable state. After that we have started the bandits and let the system stabilize in next 40 simulation days. At simulation day 50 we have allowed agents to use ILP.

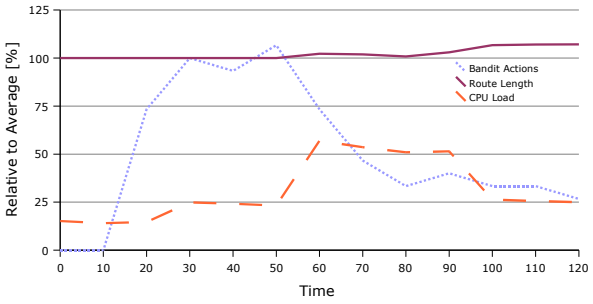


Figure 5. Course of the experiment when introducing adversarial actions in the system (day 10) and turning ILP on (day 50).

Figure 5 shows the trade-off between the number of adversarial actions and length of planned routes. Agent uses component creation mechanism to discover dangerous routes where adversarial agents are holding-up the transport. Observations sent to the ILP system are based on agent's own experience with behavior of the bandit agents. As soon as ILP creates a component for avoiding risky roads and the reflective process incorporates it into agent's plan, the number of adversarial actions significantly decreases. Expected consequence is that the average length of planned routes increases by several percents. These data are measured relatively to the *normal state* - no ILP running.

6. Conclusion

In this paper we have described the concept of agent reflection and cognition in distributed computational environment. The agents with RC architecture well adapt to the changes in surrounding environment and they better operate even in highly competitive or adversarial environments.

The *cognition*, agents' ability to understand the environment and its own reasoning process, is designed in a modular way so it can be easily extended to cover new properties of the system. Some parts of the cognition are domain independent (models, inputs for system load measurement, etc.) and some are domain dependent. The modular architecture of agent's *reasoning* layer can be easily updated or extended with new functionalities. The components (algorithm pieces) are easy to develop and deploy, although one must be careful with thread-related issues.

A series of experiments in a simulated logistic scenario have been undertaken (one of them has been presented in this contribution). These experiments lead us towards an assumptions that our RC architecture is well designed, lightweight and able to run on embedded devices. Integration on such devices would need to be carried out in order to provide proof-of-concept validation.

7. Acknowledgement

We gratefully acknowledge the support of the presented research by Army Research Laboratory project N62558-03-0819.

References

[1] S. Chiba. Load-time structural reflection in Java. In *proceedings of ECOOP 2000, LNCS*, volume 1850, pages 313–336. Springer-Verlag, 2000.

[2] P. Cointe. A tutorial introduction to metaclass architecture as provides by class oriented languages. In *FGCS*, pages 592–608, 1988.

[3] E. Denti, A. Omicini, and A. Ricci. tuProlog: A light-weight prolog for internet applications and infrastructures. In *Proceedings of the 3rd International Symposium on Practical Aspects of Declarative Languages*. Springer, 2001.

[4] T. Elrad, R. E. Filman, and A. Bader. Aspect-oriented programming: Introduction. *Commun. ACM*, 44(10):29–32, 2001.

[5] J. Malenfant, M. Jacques, and F. N. Demers. A tutorial on behavioral reflection and its implementation. In *Proceedings of the International Conference Reflection'96*, pages 1–20, San Francisco, CA, USA, 1996.

[6] S. Muggleton and L. D. Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[7] F. Ortin and J. M. Cueva. Non-restrictive computational reflection. *Comput. Stand. Interfaces*, 25(3):241–251, 2003.

[8] D. Šišlák, M. Rehák, M. Pěchouček, M. Rollo, and D. Pavlíček. $\mathcal{A}$-**globe**: Agent development platform with inaccessibility and mobility support. In R. Unland, M. Klusch, and M. Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.

[9] J. Tožička, J. Bárta, and M. Pěchouček. Meta-reasoning for agents' private knowledge detection. In M. Klusch, S. Ossowski, A. Omicini, and H. Laamanen, editors, *Cooperative Information Agent VII – Lecture Notes in Computer Science, LNAI 2782*, Springer-Verlag, Heidelberg, 2003.

Forming coalitions to leverage complementary expertise

Stéphane Airiau and Sandip Sen
Department of Mathematical & Computer Sciences
University of Tulsa, Tulsa, Oklahoma, USA
{stephane, sandip}@utulsa.edu

Abstract

Agents can form coalitions to leverage cooperation potential. In the absence of global, complete information and control to facilitate the coalition formation process, it is difficult for self-interested agents to decide on which coalition to form: a coalition that is beneficial for one agent may not be for another. To better understand the core issues of coalition formation, we develop a framework where we assume that (a) the competence profile of individual agents is known, (b) the composition function determining the competency of a coalition from the competencies of its members is known and (c) the environmental reward structure is common knowledge. Under these conditions, we study two coalition formation mechanisms: one that is completely decentralized, and one where few agents take the lead to build the team. We compare the mechanisms in two different settings: in one there is no competition between the coalitions, and in the other, the agents compete for environmental niches. Early results show that both mechanisms manages to distribute the agents between the different niches, the decentralized one performs marginally better and coalition sizes differ.

1 Introduction

In an open society of self-interested agents, using the opportunity to collaborate and form a coalition may greatly impact the performance of some agents. For a rational agent, forming and organizing a coalition takes resources: for example, in [1] there is a need to recognize the potential agents that can join a coalition, evaluate the different coalitions, and negotiate with the appropriate agents to work together. In the literature on coalition formation [2, 3], the ability of an agent to perform a task is typically assumed to be known, and it is possible to evaluate the performance of a coalition. In real life, for a team to efficiently to accomplish a given task, the required expertise must be present in the group. Moreover, the outstanding competence of some agents in certain domains can bolster the group performance. Because of the expert agents' influence on the coalition, or because other members can then focus on other task requirements, the function that determines the ability of a coalition is not linear in the competences of the member of a coalition. In this paper, we propose an abstract model of coalition formation where the ability of a coalition is shaped by the different experts present in the coalition.

In our model, each agent is described with a competency vector that specifies its level of performance over all the domains of expertise present in the environment. This vector represents intrinsic skills of the agent. We consider that the competency of the coalition is boosted by a factor which depends upon the best agent in each domain within the coalition. The research question is then how can agents efficiently self-organize by forming balanced teams where their strengths and weaknesses complement each other.

In this paper we experiment with two simple mechanisms used by agents to self-organize into coalitions. The first mechanism is an incremental trial and error scheme where agents are introduced one by one into the system, and they can join existing coalitions or form new ones. In addition, if agents in a coalition recognize they are better off by leaving their current coalition and creating a new group, they are allowed to do so. In the second mechanism, few agents take the lead to form coalitions until all the agents are accommodated. We consider two different environments for testing the coalition formation algorithms. The first environment is designed to make the agents seek complementary agents in order to improve performance. The environment contains a target competence vector that determines the success of individual agents or coalitions. The better aligned a coalition is to the target competency, the better the performance. Since the agents cannot change the intrinsic competences, the agents have an incentive to form coalitions to leverage complementary expertise to improve performance. The second environment is more realistic, since it involves multiple target vectors, each providing a bounded amount of utility. These vectors are niches of opportunity in the marketplace, requiring a specific balance of compe-

tencies to be successful, and having a maximum capacity. We present initial results for this last environment.

2 Societal model

Consider a society of self-interested agents, living in an environment $\mathcal{E}$ that can be described as a k-dimensional space $\mathcal{E} = \Pi_i D_i$, where each space D_i is a domain of competence. Each agent i is described by a vector of competencies $\vec{c_i}$ that represents the ability of the agent in the domain D_i. An agent cannot modify its competencies: these are intrinsic and represent its natural abilities.

If agents form a coalition, the competency of the coalition is not a linear function of the competencies of the individual agents. One expert agent for domain D_i can guide other agents to perform task in D_i. Or, if one expert of D_i performs all subtasks related to D_i, the other agents can spend their resource on other domains. Hence, we believe that experts can radically influence the ability of a group. We model this assumption by *boosting* the average competence of the coalition by a factor that is function of the competency of the best agent in this domain. The competency of a coalition $\mathcal{C}$ for a domain j is defined as:

$$f(\max_{i \in \mathcal{C}} \vec{c_i}(j)) \cdot \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \vec{c_i}(j),$$

where f is an increasing function with $f > 1$. This scheme also allows us to treat a coalition of agents as a single agent k represented by competency $c_k(j)$.

2.1 Aligning competence to a single target vector

We first consider an environment with one favorable profile of competence $\vec{V_E}$. This profile provides the accurate balance of competence between the domains for a coalition to perform well in the environment. Also, it can be viewed as a niche is the environment: agents aligned to this vector will prosper in this particular niche. This target vector is common knowledge among agents. The utility of an agent or a coalition depends on the alignment of its ability with $\vec{V_E}$. Since the ability of an agent is fixed, to maximize their utility, the agents must seek complementary agents and form a coalition which is better aligned with the target.

The reward of an agent or a coalition $\mathcal{C}$ is function of the projection of its competence vector on $\vec{V_E}$ and of the angle of these two vectors. Two coalitions may perform equally well for different reasons. One may have powerful abilities (the length of the competence of $\mathcal{C}$ is large), but is not well-aligned. The other is not as able, but is better aligned (in the sense the angle is small). The reward $R(\mathcal{C})_{\vec{V_E}}$ of a coalition of agents $\mathcal{C}$ with a competency vector $\vec{c}$ is given by

$$R(\mathcal{C})_{\vec{V_E}} = (V_{\mathcal{E}}|\vec{c}) * cos^p(\theta), \quad (1)$$

where θ is the angle between $\vec{c}$ and $V_{\mathcal{E}}$ and we used $p = 2$. If two vectors have the same projection on $V_{\mathcal{E}}$, the vector with the closest angle will received a higher reward. Note that this reward depends only on the competence of the group, and is independent of other agents in the environment.

2.2 Competing for different niches

In the second model, we consider a pool of agents and a pool of niches of opportunity represented by a set of environmental vectors. We assume that each niche has a fixed capacity (total utility to give away) that is shared by all coalitions. Each coalition $\mathcal{C}$ receives a utility from each niche $V_{\mathcal{N}}$ proportionally to $R(\mathcal{C})_{\vec{V_N}}$ and the utility of to one coalition depends on other coalitions. We expect the agents to form coalitions that will either be effective on a specific niche, e.g. being aligned to this particular niche thereby cornering most of its capacity, or try to be somewhat competent for multiple niches, getting a reasonable share of each targeted niche.

3 Coalition Formation Algorithms

In this section, we present two simple dynamic coalition formation schemes. The first is incremental and decentralized: agents enter the environment one at a time, and join an existing coalition or form a new one, and are allowed to collectively leave a coalition to form a new one. We assume that agents do not have memory of the competency of other agents (an agent does not remember the competency vector of an agent which was once member of the same coalition). The other scheme uses a set of leaders that recruit the agents where more leaders can be used if necessary. In the later case, agents are not allowed to leave a coalition. Agent i knows perfectly its own competencies, but it does not know the competence of all other agents in the society. To increase the efficiency of a coalition, agents are allowed to reveal their competency vectors to other members of the coalition. This can create an opportunity for agents to be deceptive, but in this work, we assume that agent are truthful.

3.1 Incremental Framework

We now present more details for the first coalition formation scheme. The sketch of the algorithm is presented in Algorithm1. A new agent in the environment can decide to join an existing coalition or can decide to form a new coalition on its own. During the process, agents within a coalition are free to examine the possibility to separate from the coalition and form a new group of their own. In this case, the remaining agents, have the choice of staying in the reduce coalition or collectively join an existing coalition. We

allow the agents to rearrange themselves until they do not find any better partition or when a maximum number of iterations has been reached. There is no guarantee that the partition of agents into coalition would be optimal.

Algorithm 1 Incremental coalition formation scheme.

while { no change for the last `max-it` iterations / `num-trials` < `max-trials` } **do**
  if all agents are not in the system **then**
    add an agent to the system by negotiating with existing coalitions.
  pick randomly a coalition $\mathcal{C}$ (single or multiple agents)
  for up to `num-max-subsets` trials **do**
    evaluate a random subset of $\mathcal{C}$, update best candidate `dissidents`
  if beneficial for subset `dissidents` **then**
    `dissidents` form a new coalition
    $\mathcal{C}\backslash$`dissidents` try to merge with existing coalitions
  `num-trials++`

3.1.1 Adding an agent in the system

When a new agent enters the system, it provides its competency vector to each coalition in the system. In response, each coalition provides the reward that the new agent will obtain if it joins that coalition. The new agent greedily chooses to join the group which offers the best reward if it exceeds its individual reward, else it forms a coalition on its own.

The coalition computes the payoff when the requesting agent is on its own, and when it was part of the coalition. The coalition makes an offer when the payoff of the new coalition is greater than the sum of the payoff of the agent on its own and the payoff of the current coalition. Many schemes can be used to share the remaining utilities, but it is not acceptable to lower the payoffs of the current members to accept a new one. We use a scheme that gives a percentage of the difference of the gain of the coalition to the newcomer, and share the remaining part of the gain proportionally to the current members. In this way, agents in a coalition are guaranteed to improve their own rewards when they accept a new agent. However, this mechanism is not fair: agents that joined early benefit from agents joining later on in the process. We plan to study whether or not it is possible to have a fair division of the utility within the group, with the constraints that agents already present in the coalition do not incur a loss of utility.

3.1.2 Splitting a coalition

In a coalition, agents know the competence and the reward of other agents. A subset of agents may realize they would be better off by forming a new coalition. As there are $2^{|\mathcal{C}|}$ possible dissident groups, we sample only few of these, unless $|\mathcal{C}|$ is small, and we allow the best dissident group to leave when it is beneficial. Let $r(i)$ denote the reward obtained by agent i in the current coalition, and R denote the reward obtained by a dissident subset of agents $\mathcal{D}$. A subset of the coalition is better off forming a new coalition when $R > \sum_{i \in \mathcal{D}} r(i)$. In order to guarantee that each agent increase its reward by leaving the current coalition, the proportion of reward obtained by the agents before they left the current coalition is maintained: each dissident agent i gets $\frac{r(i)}{\sum_{j \in \mathcal{D}} r(j)} \cdot R$ in the new coalition if formed.

The remaining agents in the coalition $\bar{\mathcal{C}}$ are very likely to suffer a drop of reward. They first try to merge with an existing coalition $\mathcal{C}_o$. It is beneficial for $\mathcal{C}_o$, allow $\bar{\mathcal{C}}$ to merge with it if $R(\mathcal{C}_o) + R(\bar{\mathcal{C}}) < R(\bar{\mathcal{C}} \cup \mathcal{C}_o)$. If no other coalition is willing to merge with them, agents in $\bar{\mathcal{C}}$ are left in that coalition. It is likely that the coalition will then split again in future iterations. Also, as new agents come into the system, the coalition will be able to recover from the loss of other agents.

3.2 Team Leader Approach

In order to facilitate the formation of the teams in the second protocol, we use some agents as leaders who pick agents to be part of their team. For this scheme, we first select randomly the set of leader agents. This choice provides a lower bound on the number of coalitions in the environment. In turn, each leader selects one agent from the pool of remaining agents. The leaders perform a greedy search to build their team: they pick the agent that maximizes the payoff of the team. The leader is responsible to ensure that the addition is beneficial for the current coalition and the newcomer agent (as in the incremental approach). The newcomer cannot refuse its assignment. The leader, however, can decide not to select any additional agents, and in that case, the process of forming its team is complete. If all the initial leaders complete their teams and some agents have no been picked, a new leader is randomly picked and the process goes on until all agents are members of some team. In this scheme, we are not interested in the process of sharing the utility within the group, we assume that the agents use a fair division of the utility. Also, agents cannot leave its assigned coalition.

4 Experimental Results

The goal of the simulations is to observe the emergence of efficient coalitions. We first present experiments with a single niche, with unbounded capacity, then move to initial experiments involving multiple bounded capacity niches.

4.1 Single Niche, unbounded capacity

The first environment we study consists of 6 domains and contains 40 agents. The competency vectors and the target vector are generated randomly: each entry is drawn from a uniform distribution in $[0, 1]$. First, we study the properties of the reward function used (see equation 1). We study the average payoff per agent for up to a million random teams of fixed size, showed in Figure 1. Note that in this domain, the performance of a team is independent of other teams. This plot illustrates that the environment is super-additive, since the average payoff is a monotonic increasing function of the number of agents. For teams larger than 10 agents, however, there is not much gain. The best team is formed by four agents and we expect that agents may form small efficient teams.

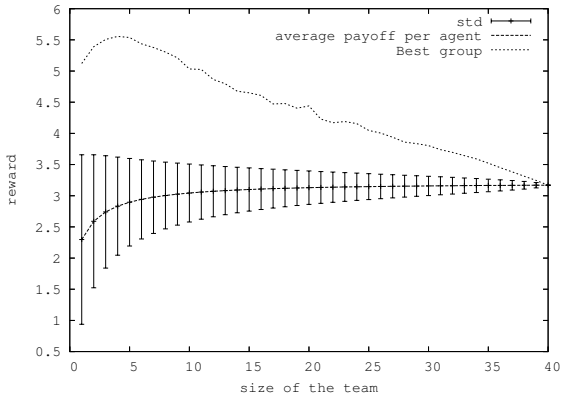


Figure 1. Random teams of fixed size.

Next, we present the results over runs using the incremental coalition formation protocol. the life cycle of a typical coalition is shown in Figure 2. The iterations on the x-axis of the graph represent either the entrance of a new agent in the environment or a split in a coalition. The simulation either stops because a maximum number of trials has been reached, or because no coalition has changed for a given number of trials. A typical coalition first increase in size, until a group of agents recognize an opportunity to leave the coalition (e.g. at iteration 26). In that particular case, a group of 7 agents leave the coalition. The remaining agents can not join another coalition, and hence the coalition continues at a reduced number. After the split, the coalition is getting few new agents that are entering the system. At iterations 18 and 34, the coalition accepts a group of agents from another coalition that suffered a split. At iteration 40, the coalition is splits again. But this time, the remaining agents are join an existing coalition and this coalition dies.

The dynamics of the reward per agent is presented in Figure 3. The average agent performs in a coalition better than

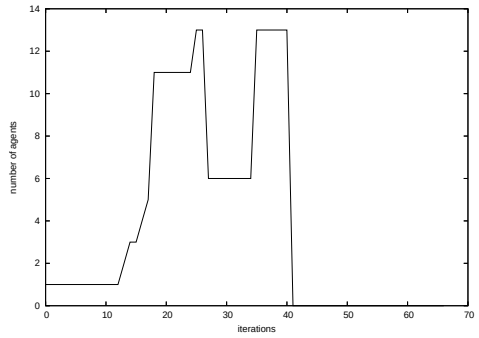


Figure 2. Life cycle of one coalition.

when it works on its own, which shows the boosting effect of a team. We also represent the reward obtained by the best group. The curve is not smooth, due to the different splits. Even the best team can split, showing that the groups were not stable. When all agents are in the environment, only splits can occur. We see that the splits does not much affect the average payoff per agents. The best team, however, is affected by these splits.

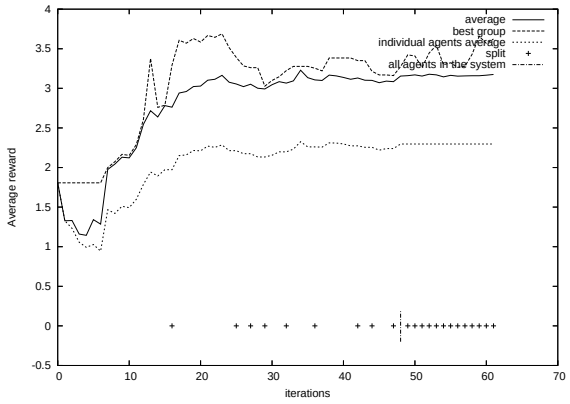


Figure 3. Incremental formation process.

In Figure 4, we present a typical process of team formation using four initial leaders. Each line in the chart represents one team. We can see during the first five iterations, each leader is adding one agent to their team. At this point, the second leader cannot improve its team. The third leader can, but not the fourth. A new round occur where the first and third leader add a new agent in their teams, and finally, only the third leader can improve its team. At iteration 10, it can no longer improve its team. Then, a new leader is randomly chosen and create its team. Subsequently, five additional leaders are needed to have all the agents in the system.

Teams that are formed after the completion of the initial leaders' coalitions may not be able to perform as well, since

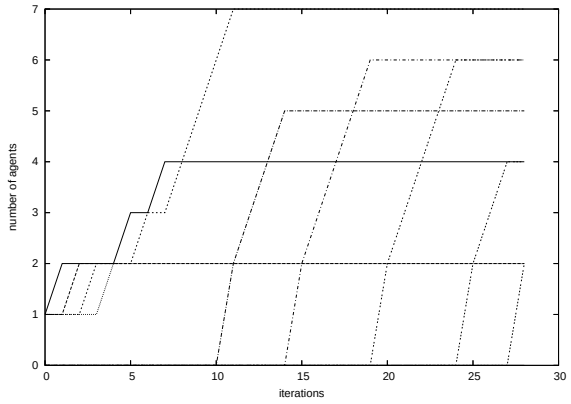


Figure 4. Evolution of the number of agent in each team during the formation process.

the choice of available agents is limited. However, Figure 5 shows it is not the case. The chart represents the evolution of the reward during the same formation process. We can see that the two best team originate from an initial leader. But the team starting at iteration 10, 14 and 19 perform better than the two other initially created teams. Note that this is the total performance of the teams and that the team starting at iteration 10, 14 and 19 contains also more agents.

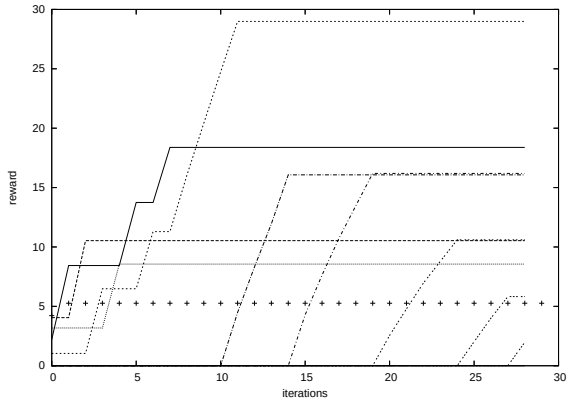


Figure 5. Evolution of the reward of each team during the formation process.

We compare the performance of the two methods to form the teams. We are interested in the creation of the team which have the best utility per agent. For these experiments, we used the same set of agents, but they are introduced in a different order (from one run to the next, the order at which the agents are introduced in the incremental process is changing, and the set of initial leaders is different). The results are presented in Table 1, the number in each cell is the average over 100 runs, and the number in parenthesis denotes the standard deviation. First, we note that the average utility per agent is better when we use the incremental process. This is probably due to the fact that in the incremental process, the agents can rearrange themselves into new coalitions by leaving current coalition to increase their payoffs. On the contrary, in the leader based process, the leader is concerned about the best result for the team, guaranteeing some surplus for the newcomer, but not the best surplus. The main difference between the two mechanisms lies in the size of the teams: the incremental process tends to create few teams, whereas the leader process creates a large number of small teams. This is probably due to the fact that in the leader based scheme, the leader first chooses an agent that complements itself the best, and it becomes harder to improve on these teams.

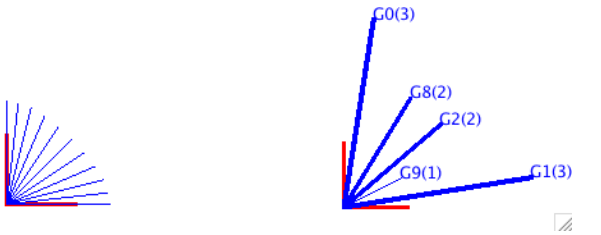
	Inc	Leader
best team (utility/agent)	3.462 (0.205)	5.0224 (0.078)
size of the best team	17.43 (11.38)	2.76 (0.767)
average utility/agent	3.156 (0.008)	2.9435 (0.025)
number of team	3.29 (0.689)	10.2 (0.85)

Table 1. Comparison of the coalition formation mechanism for a single niche.

4.2 Multiple Niches

We now present initial results in the environment with multiple niches. First we experiment with two domains in order to visualize the repartition of the coalitions. An example of grouping of agents into coalition is presented in Figure 6. The picture on the left presents the 11 agents in the system. There are two niches which are the bases of the space. In the particular instance shown, 5 coalitions are formed. We can see that G0 and G1 are trying to focus on one of the niches, whereas the other coalitions are more trying to exploit both niches. We observed that most of the time, the agents are forming coalitions, but some inefficiencies remains.

In Table 2, we see that in the leader-based scheme, the leader always pick an agent, and never defects, since the number of teams is always 4. In the incremental scheme, only two teams of largely variable sizes are created in different runs. As all niche utilities are completely distributed among all the teams, the metric of relevance in this domain is the average utility of best team members. The incremental scheme does slightly better in this domain also. Hence, in our domains, the distributed, incremental coalition formation scheme performs competitively with the more centralized leader based scheme.



Individual Agents After Coalition Formation

Figure 6. Example of a team

	Inc	Leader
best team (utility/agent)	0.14102 (3e-5)	0.15 (2e-5)
size of the best team	28.15 (24.9)	10 (0.0)
average utility/agent	0.13753 (0.008)	0.13753 (0.0)
number of team	2 (3.87e-2)	4 (0.0)

Table 2. Comparison of the coalition formation mechanism for multiple niches.

5 Related Work

As in [3], our environment is non-super-additive, at least one pair of potential coalition is not better off by merging into one. As argued in [2], we are interested in the dynamic process that leads to the formation of a coalition. In work such as[1, 4, 5], a dynamic coalition formation is considered. In [1, 2], a coalition leading agent manages the coalition formation process. It first prepares the coalition by defining the tasks to be shared and determines the set of candidate agents. Then, it simulates the formations to determine possible performance and risk of the formation failing, which is based on past interactions with other agents. Finally, it negotiates separately with the different agents; if a negotiation fails, the formation is halted and the leading agent restarts a simulation phase. The general approach is similar in [5]. Learning is also used to pick the negotiation strategy. Task allocation is performed when the coalition has already formed. [4] also considers multiple negotiation strategies based on the Contract Net Protocol. In these approaches, the coalition formation process succeeds or fails, but an entire coalition is considered as a whole at each time, whereas in our work, we consider that agents enter the environment one by one. Once the coalition is formed, the task is performed and it is assumed that the coalition dissolves. In our model, we consider long-term coalitions, as in [6]. In this work, buyers and sellers form coalitions with the goal of bringing compatible agents closer. Individual agents can join or change groups and this choice is based on the analysis of past interactions. In our model, we also want compatible agents to get closer, but only the competence is taken into account. The originality of our work also resides in the payoff structure. In our model, the competence of the coalition of agents is not the sum of the competence of its members.

6 Future work and Conclusion

We present a model of coalition where the aggregation of the competence of the agents is not linear. In order to form an insight into the dynamics of the coalition formation, we study two dynamic coalition formation processes. We are particularly interested in the distributed, incremental scheme for its applicability in real world group formation. Early results show that the agent can build effective coalitions in such a distributed manner both in domains with bounded or unbound niche potentials.

Our work is empirical, immature and lacks some theoretical guarantees. Adding one agent at a time raises the issue of a fair distribution of the utility. Agents already in a coalition should not lose utility due to incorporation of a new agent. With our current mechanism, agents that enters the coalition early have an advantage. We need to study alternative utility sharing schemes. It will be interesting to see the impact on the dynamics of the formation coalition. Results with multiple niches are preliminary and we need to perform a deeper study in the environment involving multiple niches with varying potential.

References

[1] M. Klusch and A. Gerber. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17(3):42–47, May/June 2002.

[2] M. Klusch and A. Gerber. Issues of dynamic coalition formation among rational agents. In *Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations*, pages 91–102, 2002.

[3] O. Shehory and S. Kraus. Feasible formation of coalitions among autonomous agents in nonsuperadditve environments. *Computational Intelligence*, 15:218–251, 1999.

[4] M. Sims, C. V. Goldman, and V. Lesser. Self-organization through bottom-up coalition formation. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS03)*, pages 867–874, New York, NY, USA, 2003. ACM Press.

[5] L.-K. Soh, C. Tsatsoulis, and H. Sevay. A satisficing, negotiated, and learning coalition formation architecture. *Distributed Sensor Networks: A Multiagent Perspective*, pages 109–138, 2003.

[6] J. Vassileva, S. Breban, and M. Horsch. Agent reasoning mechanism for making long-term coalitions based on decision making and trust. *Computational Intelligence*, 18(4):583–595, 2002.

Multidimensional Context Representations for Situational Trust

Martin Rehák, Miloš Gregor and Michal Pěchouček
Department of Cybernetics and Center for Applied Cybernetics, Czech Technical University in Prague
Technická 2, Prague, 166 27, Czech Republic
rehakm1@labe.felk.cvut.cz, pechouc@labe.felk.cvut.cz

Abstract

We propose a generic model for situation representation for agent trust models. As we argue that general trust models are inappropriate in complex environments and their use for advanced decision-making is limited due to the lack of detail, we introduce a generic framework for context representation in trust model. Presented method is independent of the trustfulness representation formalism in the underlying trust model. It doesn't prescribe any particular form for the set of reference contexts that holds the trustworthiness information in the context space: we define and evaluate the baseline approach with regular grid and discuss future extension towards more adaptive approaches using octant trees and fuzzy clustering techniques.

1 Introduction

This contribution addresses the need to efficiently represent complex contexts in the trust model, both for the observations (or impressions) and the current decision. The need to represent the context is common for all agent models that attempt to model a situational trust: trust in an agent in a particular situation [5]. To successfully model situational trust, we have to model the situation – represented by context – first.

A crucial problem to address is a selection of the mapping between the *situation* in the environment and the *context*: a formal representation of the situation in our model. Obvious problem to address is a selection of relevant situation attributes to include into the context, as well as issues of scale and precision. Obviously, successful situation model has to contain all the relevant attributes that influence the trusting decision. On the other hand, adding more characteristics than necessary makes it difficult to determine the similarity between two situations, assuming that the trusting situations are rarely completely identical. Our context model addresses these concerns by resorting to standard approaches from mathematics and artificial intelligence, that are well adopted for the task. The main modelling problems we want to address are:

- **computational** and **memory efficiency**: we need to devise a model that doesn't require storage of all observations and their processing for each trusting decision;
- **learning efficiency**: the model must use the observations efficiently, to allow fast bootstrapping even in very big context spaces.

2 Formal Model

We denote agents in the community $A, B, C, ...$, forming a set $Agents$. General trustfulness (without taking the context into account) of agent B as perceived by agent A is denoted $\Theta_A(B)$. WE define context space $\mathbb{C}$ as a metric space of contexts with distance function $d(c_1, c_2)$. The context of each trust observation $\tau_A(X|c_i)$ or trusting decision $\delta_A(X|c_i)$ (where A is the observing/deciding agent, X denotes the evaluated agent and c_i is a context of this observation/decision) represented by exactly one point in this space, but several different observations may fall into the same point c_i in $\mathbb{C}$.

2.1 Context Space

In general, we define the metric space $\mathbb{C}$ in several steps:

1. Identify all relevant features of the environment.
2. Define the Q-dimensional context space where each dimension q matches a relevant feature of the trusting environment.
3. For each dimension q, define its quantification (either discreet or continuous) and appropriate distance metric d^qthat correctly represents the feature.

4. Define a joint metric d on the full space $\mathbb{C}$, taking into considerations the domain characteristics and marginal metrics d^i[1].

There are 4 basic properties of any distance function $d : \mathbb{C} \times \mathbb{C} \rightarrow R$ that we need to respect while we define our own, domain dependent distance metrics:

1. non-negativity:
$$d(c_1, c_2) \geq 0 \tag{1}$$
2. symmetry:
$$d(c_1, c_2) = d(c_2, c_1) \tag{2}$$
3. zero distance $\Leftrightarrow$ identity:
$$d(c_1, c_2) = 0 \Leftrightarrow c_1 = c_2 \tag{3}$$
4. triangle inequality:
$$d(c_1, c_3) \leq d(c_1, c_2) + d(c_2, c_3) \tag{4}$$

Obviously, all marginal distance metrics d^q shall respect these requirements as well.

To combine the marginal distances into the d function, we will typically chose one of the special types of Minkowski distance:

$$d(c_1, c_2) = (\sum_{q=1}^{Q} |c_1^q - c_2^q|^p)^{\frac{1}{p}} \tag{5}$$

For many practical purposes, we chose the values of p to be 1, defining so called Manhattan distance that adds the marginal distances of each dimension, or 2, to define an Euclidean distance, or we pose $p \rightarrow \infty$, obtaining Chebyshev distance defined as a maximum of marginal distances.

For the purpose of determining the weights in our model, we need to define a (possibly domain dependent) function $f : \mathbb{R}^{0+} \rightarrow [0, 1]$, where the $\mathbb{R}^{0+}$ is an output range of the distance metric function. Naturally, we require f to be *non-increasing* on this interval – points that are farther away shall have smaller or at most identical weight compared to the points in proximity.

2.1.1 Example

To demonstrate the abstract notion introduced above, we will apply them on a realistic (albeit simplified) problem throughout this contribution.

To illustrate the abstract notions of metric space $\mathbb{C}$, we introduce an example of such space in a specific business situation. We model the trust reasoning of an agent that acquires furniture sets from several providers. In this domain, we model each trusting situation (either observation or decision) by three parameters: type, size and market state. *Type* defines the product we acquire: kitchen set, bathroom set and lounge sets are the cases we consider in the current version of the model. Type of the product defines the product sophistication and a set of techniques used for its elaboration and installation – obviously, each provider can be proficient in different set of operations that are more or less important for each product type. It is interesting to note that this dimension is not continuous, but discrete. *Size* of the contract is measured in terms of its price, while the *market state* (or *supply*) describes the ratio of the supply/demand on the given market. Both size (price) and market state are considered as real-valued variables, but varying greatly in their scale: one has an absolute scale (price), while the other will most often fall very close to 1.

The context space $\mathbb{C}$ is therefore three dimensional, with one discreet dimension and two continuous ones. The next step is a definition of marginal distances d^q for each dimension. In the *type* domain, we place our products on a "technicity" scale: bathrooms score the highest: 5, with the kitchens in the middle: 1 and the bedrooms as the least technical ones, with 0.2 value. Note that this doesn't mean that bathroom production is more difficult than bedroom: the distance merely describes the differences in the skills required and inverting the scale will not change the result thanks to the distance symmetry stated in Eq. 2. Our distance metrics is defined as follows, using the product values defined above:

$$d^{type}(c_1, c_2) = |ln(type_1) - ln(type_2)| \tag{6}$$

In the price domain, the metric shall describe the similarity between two contracts in terms of their price. We propose a measure

$$d^{price}(c_1, c_2) = |ln(price_1) - ln(price_2)| \tag{7}$$

. The logarithmic relation captures an intuitive notion of ratio: 100$ difference between two 200$ and 300$ contracts is much more important than the same difference between two contracts priced in millions.

We apply the same reasoning for the market state (supply) dimension:

$$d^{supply}(c_1, c_2) = |ln(supply_1) - ln(supply_2)| \tag{8}$$

Then we combine the above metrics using a slightly modified (weighted) "Manhattan distance":

$$d(c_1, c_2) = \alpha_1 d^{type}(c_1, c_2) + \alpha_2 d^{price}(c_1, c_2) + \alpha_3 d^{supply}(c_1, c_2) \tag{9}$$

2.2 General Algorithm

In the context space, we define one or more *reference contexts* r_i, forming a set $\mathcal{R}$ in $\mathbb{C}$. For each point r_i and each

[1] Alternatively, we may proceed without defining a distance metric for each dimension, if we define the overall metric directly.

partner agent X we (agent A) keep trustfulness estimate denoted $\Theta_A(X|r_i)$. This value can be a result of application of any relevant trust model – for example Regret [9], FIRE [4] or other [7, 1, 8], provided that the inputs to this model are weighted or selected to reflect the performance of agent X in the specific context r_i.

Therefore, each observation $\tau_A(X|c_o)$ is used to update the trustfulness of reference contexts $r_i$with the weights determined using the general formula:

$$w_i = f(d(c_d, r_i)) \tag{10}$$

, where f is a non-increasing function on $[0, +\infty)$ as defined above. This function represents the decay of the observation usefulness with increasing distance d of the particular reference context r_i – obviously, it is most useful when its distance $d(c_d, r_i)$ from the reference context is zero and it shall decrease with increasing distance. This function, together with the metric, is a part of the domain description. For example, in our simple experiment presented in Section 3.1 we use a simple form of weight function defined as $w_i = e^{-d(c_d,r_i)}$.

Generally speaking, we integrate the new observation $\tau_A(X|c_o)$ into the apriori trustfulness evaluation $\Theta_A^p(X|r_i)$ (where p is the number of previous observations, with aggregate weight $\sum_{j<=p} w_i^p$) for each r_i (where the corresponding w_i is non-zero) using the weighted aggregation formula:

$$\Theta_A^{p+1}(X|r_i) = \tag{11}$$
$$WeiAggr((\Theta_A^p(X|r_i), \sum_{j<=p} w_i^p), (\tau_A(X|c_o), w_i^{p+1}))$$

Exact form of the $WeiAggr()$ operator depends entirely on the model used to represent $\Theta_A^{p+1}(X|r_i)$. Assuming for the moment that the $\Theta_A(X|r_i)$ is just a w_i weighted average of all p previous observations, we obtain the update relationship:

$$\Theta_A^{p+1}(X|r_i) = \frac{(\sum_{j<=p} w_i^j)\Theta_A^p(X|r_i) + w_i^{p+1}\tau_A(X|c_o)}{(\sum_{j<=p} w_i^j) + w_i^{p+1}} \tag{12}$$

In the decision time, when we take a trusting decision, current context is determined and the trustfulness is deduced as a weighted combination of trustworthiness of reference contexts.

$$\Theta_A(X|c_d) = WeiAggr_{r_i \in \mathcal{R}}((\Theta_A(X|r_i, w_i)) \tag{13}$$

By assuming the weighted average case again, we obtain:

$$\Theta_A(X|c_d) = \frac{\sum_{r_i \in \mathcal{R}} w_i \Theta_A(X|r_i)}{\sum_{r_i \in \mathcal{R}} w_i} \tag{14}$$

The general update approach as presented in Eq. 11 ensures that several trustfulness relative to different contexts r_i $\Theta_A(X|r_i)$ are updated simultaneously – this is a critical feature for any trust model, as it reduces the time before the model can provide meaningful results. Similarly, the decision-making Formula 13 gathers the data from all relevant contexts, increasing the quantity of the information the trusting decision is based on. On the other hand, this model characteristic can turn in our disadvantage as it can be exploited by informed adversary – therefore, in our future work, we intend extend the Formula 11 to reflect the amount of the information we already have and to reduce the weight of distant reference contexts accordingly.

In this contribution, two different classes of approaches to reference contexts definition will be examined. In the first approach, the reference points will be placed regularly through the metric space (see Section 3), while in the second approach we will enhance the behavior by introduction of adaptive techniques, as shown in Section 4.

Note that while the context space $\mathbb{C}$ properties (e.g. metrics and dimensions) are the same for all agents modelled by agent A, the actual instances of the trustfulness and reference points r_i positions are separate for each evaluated agent – while they may coincide in the reference grid approach, this is no longer true for adaptive approaches mentioned below.

3 Regular Grid Approach

In the first approach to the problem, the reference context set $\mathcal{R} \in \mathbb{C}$ is defined as a regular grid covering the space $\mathbb{C}$ in each dimension q, where the regularity is defined by the distance metric d^q for each dimension. The density of $\mathbb{C}$ sampling is therefore defined a-priori, in design time, before we know what will be the real distribution of samples c_i.

3.1 Experimental Evaluation

To evaluate the regular grid approach, we have conducted a series of experiments using the context space defined in Section 2.1.1. In our simple scenario, there are 10 buyer agents who acquire furniture sets from 5 providers. Trustfulness of each provider is strongly dependent on context and influences the real price of the delivery; in our domain, we assume that the customer is bound to pay for the excess cost billed by the supplier.

The core of the supplier model consists essentially of the real price and bid price computation mechanism. *Real price* that is billed at the end is determined using the relation $pr_r = cost \cdot margin$, where $cost$ denotes the cost associated with the supply of goods and $margin$ is the profit coefficient of the supplier. The *bid price*, submitted by the supplier as a (part of) response to CFP, is already influenced by its real *trustworthiness* Θ through the simple relation: $pr_b = pr_r \cdot \Theta$.

Supplier trustworthiness Θ is a key parameter of our experiment. It is context dependent and is defined as a function of three parameters that define the context of the operation in the space $\mathbb{C}$: *price* (size) of the contract, *type* of the goods to provide and *supply*: the state of the market at the moment.

$$\Theta = \Theta_{type} \cdot atan'(price) \cdot atan'(supply) \qquad (15)$$

where the function $atan'(x)$ is defined as a normalized $arctan$: its range is (x_{inf}, x_{sup}) (both x_{inf}, x_{sup} are in the range set) and x coordinate of its flection point is defined by parameter x_{center}. x_{slope} determines the first derivation - speed of the growth on the domain.

$$atan'(x) = \frac{1 - x_{inf}}{\pi} \cdot arctan(\frac{x_{center} - x}{x_{slope}}) \qquad (16)$$

On the customer side, each agent A maintains its trust model. While the context space $\mathbb{C}$ properties and grid-defined $\mathcal{R}$ are identical for all suppliers, each supplier X is modelled by its own trustworthiness values $\Theta_A(X|r_i)$ in each point r_i. These values are used to discount the bids of suppliers when they answer a particular CFP (CFP content defines the point c_d in $\mathbb{C}$.) Discounting is implemented as a bid price modification – agent actually uses X's trustworthiness to estimate the real price using the relation:

$$\hat{pr}_r(X|c_d) = pr_b \cdot (\Theta_A(X|c_d))^{\kappa} \qquad (17)$$

where $\kappa \geq 1$ is a discounting coefficient that penalizes inappropriate behavior. For our experiment, we impose $\kappa = 2$. Once the discounted prices are known by A, it simply selects the agent X with lowest $\hat{pr}_r(X|c_d)$ as a supplier for a particular case. Upon delivery, agents use Formula 11 to update its trust model regarding the X's performance.

In Table 3.1, we present the results of evaluation. We have selected several significant types of contract and evaluated the ratio of cases when the customer selects the most trustworthy supplier or one of the best two or three ones. We have aggregated the data regardless of the supply to obtain less columns, but the parameter was used for estimation. The same results are presented under graphical form in Fig 1, Fig 2 and Fig 3.

Note that we have intentionally set our model to simulate "greedy" and optimistic customers – the emphasis on the bid price is major and quite often, the cheapest bid wins, making the overall values quite low. Overall, except for the kitchen case where only 4 suppliers were available, we may conclude that the situational trust model performs consistently better than the general trust model. On the other hand, the application of this model is not without consequences regarding the computational effort for each decision or observation, as we discuss in the suite.

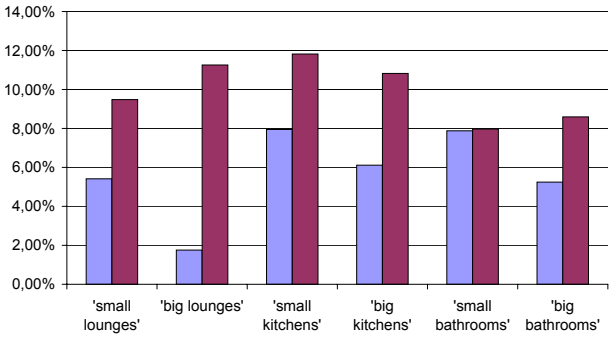


Figure 1. Percentage of cases when the best provider was actually selected for given class of problem. Situational Trust (right bars) is compared with general trust (left)

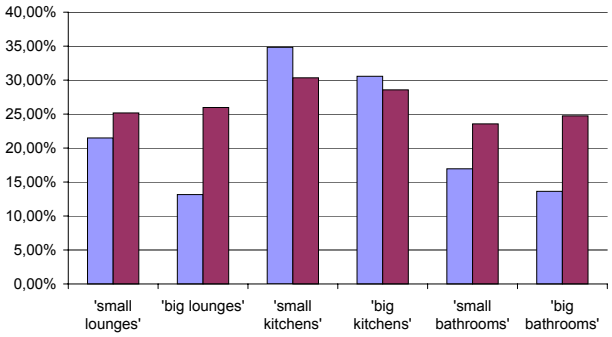


Figure 2. Percentage of cases when one of the two best providers was selected for given class of problem. Situational Trust (right)compared with general (left).

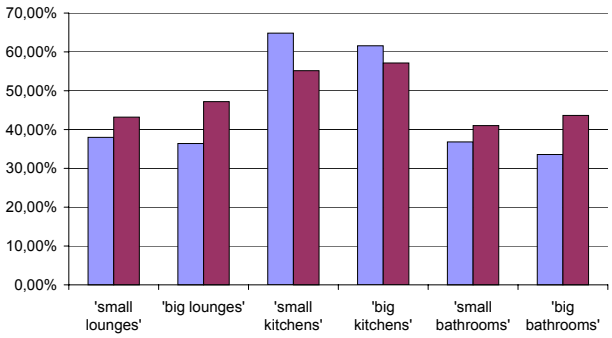


Figure 3. Percentage of cases when one of the three best providers was actually selected for given class of problem. Situational Trust (right)compared with general (left).

Market	*small lounges*	*big lounges*	*small kitchens*	*big kitchens*	*small bathrooms*	*big bathrooms*
General Trust						
Best Ag.	5.41	1.75	7.95	6.11	7.88	5.24
Best 2 Ag.	21.49	13.16	34.83	30.57	16.95	13.64
Best 3 Ag.	37.99	36.40	64.84	61.57	36.82	33.57
Situational Trust						
Best Ag.	9.48	11.26	11.82	10.82	7.97	8.59
Best 2 Ag.	25.17	25.97	30.34	28.57	23.56	24.74
Best 3 Ag.	43.20	47.19	55.16	57.14	41.02	43.64

Table 1. Results of experiments. Each value is the ratio (in %) of times when the best, or one of the 2/3 best partners was selected.

3.2 Limitations of the Approach

Even if the grid approach suggested above satisfies the requirements for the representation of multi-dimensional context in trusting situation, it suffers of scalability concerns.

The main limitation is its granularity: the grid is defined by system designer and is difficult to change once the code is deployed, especially in case of situations when the decision must be taken rapidly or when we have to address hard system constraints. As in the majority of deployment cases the points will be not be spread uniformly, but rather concentrated in several regions of $\mathbb{C}$, most of the reference contexts will be almost useless. On the other hand, there will be relatively few reference contexts in the regions with high concentration of trust situations. Efficiency of the trust update (Eq. 11) decreases significantly with growing density of the reference context grid that each agent maintains for each partner. Therefore, designers face tough decisions: performance optimization can seriously affect the quality of the trust model as it imposes the reduction of grid density. In the remainder of the paper, we are going to examine possible solutions of this problem.

4 Towards Adaptive Approach

The first alternative approach is the introduction of grid with adaptive density using the octant tree-like approach [3]: it shall automatically add new reference contexts in the areas with high density of *diverse* observations. Such regions are easy to detect if we apply trust models that explicitly represent the uncertainty of the information (As most of the current models does – see [6]). Once we detect a that some area features a high number of diverse observations, we split the grid cells (multidimensional) into $2^{dim(\mathbb{C})}$ subcells and introduce new reference contexts into the set $\mathcal{R}$. To compute the initial trustfulness values $\Theta_A(X|r_i$ for the new reference point r_i, we apply the Formula 14 – the same one used for evaluation of trustfulness in the decision-making phase. Note that while the initial set $\mathcal{R}$ used by trusting agent A for all partners is equivalent, use of any adaptive method will modify the sets used to model the trustworthiness of other agents individually, as the variability of behavior in different contexts can vary from agent to agent, as well as typical cooperation contexts. Along similar lines, we may reduce the number of reference contexts in the areas with sparse interaction or similar behavior.

A complete extension of this principle is a situation when we initially model each agent with a single reference context r_0; this is equivalent to general trust[2]. Once we detect that the model quality doesn't satisfy our needs, we split the space $\mathbb{C}$ into $2^{dim(\mathbb{C})}$ subspaces and continue with the above algorithm until the results are satisfactory.

Another alternative approach we propose addresses the limitations of the regular grid approach by leveraging classic classification techniques. We propose use a fuzzy c-means (k-means) clustering algorithm [2] to define the set $\mathcal{R}$ – the center of each fuzzy cluster will define a reference point and the corresponding trustfulness will be derived from the observations forming the cluster. The advantage of using the fuzzy variant of this unsupervised-learning algorithm is the fact that each observation contributes its information to several reference contexts: learning is faster, even if arguably more precise. We shall also note that in our particular case, we don't optimize to perfectly separate the clusters in the space $\mathbb{C}$, but rather to define a representative set $\mathcal{R}$ to hold the trustworthiness information.

Fuzzy c-means minimizes following objective function:

$$J_m = \sum_{i=1}^{p} \sum_{j=1}^{|\mathcal{R}|} \mu_{ij}^m d(c_i, r_j) \tag{18}$$

, where μ_{ij} is a membership of the sample context c_i in the cluster j, defined around the reference context r_j. p is the number of available observations.

[2]With the assumption that the function f used to determine weights is initially constant: $w_i = 1$.

Position of each reference context is defined by the center of the corresponding cluster – it may therefore move as new observations are classified:

$$r_j = \frac{\sum_{i=1}^{p} c_i \mu_{ij}^m}{\sum_{i=1}^{p} \mu_{ij}^m} \quad (19)$$

the membership coefficient is determined as follows:

$$\mu_{ij} = ((\frac{d(c_i, r_j)^{\frac{2}{m-1}}}{d_{\mathcal{R}}}))^{-1} \quad (20)$$

where the $d_{\mathcal{R}}$ is defined by the relation $d_{\mathcal{R}} = \sum_{k=1}^{|\mathcal{R}|} (d(c_i, r_k))^{\frac{-2}{m-1}}$.

The main problem with the use of fuzzy k-means is the initial phase, when we have o determine how many clusters to create and with what initial center positions. It shall be noted that our situation is somewhat different from the classic clustering – we don't have all the points available from the beginning, but we obtain them one after another, depending on the observation rate and time.

5 Conclusion

In this contribution, we have addressed a very specific problem relevant to trusting decisions in complex environments. While the simple, specialized agents can successfully rely on general trust models [8], once we want to consider the tradeoffs of particular decision or use the trustfulness estimates to draft and evaluate several alternative coalition plans for the same goal, these models can typically no longer provide relevant outputs to support such decisions. We have detailed the general model of *context space* and *reference contexts* where we represent important features of the situation. We have also specified a generic trust update method and trustfulness aggregation method using the general set of reference contexts. This method is independent on the exact form of the set $\mathcal{R}$ and the trustfulness representation formalism used in individual reference contexts.

In the remainder of the article, we have introduced three distinct forms of the set R: regular grid, adaptive octant tree and unsupervised fuzzy k-means clustering. We have performed experiments using the regular rid and determined that it compares favorably when compared with the general trust model that uses the same formalism.

In our future research, we plan to correctly validate and benchmark all above mentioned approaches of reference set representation and determine their mapping to various types of trusting problems and computing environments - we intend not only to evaluate them with respect to trust model quality, but also to describe their computational complexity and other relevant properties. Integration of this method with advanced decision-making algorithms is also a promising area of research.

Acknowledgment

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-04-1-3044. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

References

[1] C. Castelfranchi and R. Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, page 72. IEEE Computer Society, 1998.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, NY, USA, 2nd edition, 2001.

[3] S. Frisken and R. Perry. Simple and efficient traversal methods for quadtrees and octrees. *Journal of Graphics Tools*, 7(3), 2002.

[4] D. Huynh, N. R. Jennings, and N. R. Shadbolt. Developing an integrated trust and reputation model for open multi-agent systems. In *Proc. 7th Int Workshop on Trust in Agent Societies*, pages 65–74, 2004.

[5] S. Marsh. Formalising trust as a computational concept, 1994.

[6] S. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multiagent systems. *The Knowledge Engineering Review*, 19(1), 2004.

[7] S. Ramchurn, N. Jennings, C. Sierra, and L. Godo. Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence*, 18(9-10):833 – 852, 2004.

[8] M. Rehák, Lukáš Foltýn, M. Pěchouček, and P. Benda. Trust model for open ubiquitous agent systems. In *Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference*, number PR2416 in IEEE, 2005.

[9] J. Sabater and C. Sierra. Regret: reputation in gregarious societies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 194–195. ACM Press, 2001.

The Case for a Reference Model for Agent-Based Systems

Pragnesh Jay Modi, William Regli
Department of Computer Science
Drexel University
Philadelphia, PA 19104

Israel Mayk
US Army CERDEC C2D
AMSEL-RD-C2-SS-T
Fort Monmouth, NJ 07703

Abstract

The current state of the art in agent technology sees that several implementations of agent frameworks exist. However, there is little agreement on the terms and concepts used to describe such systems, which is a significant barrier towards adoption of these technologies by industry, military and commercial entities. A clear definition of terms and concepts at an appropriate level of abstraction is needed to facilitate discussion, evaluation and adoption of these emerging agent technologies. In this paper, we argue that a reference model for agent-based systems can fill this need. We discuss what a reference model is and why one is needed for agent-based systems. While the complete model is a work in progress, we present a preliminary version to motivate further discussion from the agents community at large. It is our hope that ultimately a wider community of practice will assume responsibility for the standardization similar to the way that the well-known seven-layer Open Systems Interconnection (OSI) reference model was a driving force underlying communications standards.

1 Introduction

The ultimate goal of the Agent-Based Systems Reference Model (ABSRM) is to provide a technical recommendation for a reference model for those who develop and deploy systems based on agent technology. The ABSRM should allow for existing and future agent frameworks to be compared and contrasted as well as to provide a basis for identifying areas that require standardization within the agents community. As such, the aim of the ABSRM is to

- establish a taxonomy of terms, concepts and definitions needed to compare agent-based systems;
- identify functional elements that are common in agent-based systems;
- capture data flow and dependencies among the functional elements in agent-based systems;
- specify assumptions and requirements regarding the dependencies among these elements.

As a reference model, the ABSRM will make no prescriptive recommendations about how to best implement an agent-based system; nor is the objective to advocate for any particular approach, architecture or framework. In its broadest sense, an agent-based system for the purposes of the ABSRM simply describes a software platform for building agents and supporting their communications and collaboration. An agent-based system may consist of many different kinds of agents operating across a heterogeneous set of platforms and hosts.

One novel aspect of our approach is to create the reference model based on a forensic analysis of existing agent-based systems. The reference model developed in this document is based on static and dynamic software analysis of existing agent frameworks. Examined frameworks include Cougaar, JADE, RETSINA, and others. Anyone building an agent framework would have to recreate or reproduce some portion of the functionality or components in these existing frameworks (i.e., to enable communications, to enable agent startup and shutdown, etc). By analyzing existing frameworks and the agent-based systems they can be used to build, we can avoid the debate concerning "what is an agent" and simply document the existing state-of-the-art systems that are called agent-based. The model aims to document a superset of the features and functional concepts in the set of existing agent frameworks. Given that there is significant variation between existing frameworks and the functions they may provide, the reference model should describe at an abstract layer the complete set of functional components across all examined agent frameworks.

It is important to note however that the reference model is not confined to being a description of capabilities of existing systems —it serves as a basis for situating the complete set of functions that anyone may want or need to have in an agent-based system. For example, security for mobile agent code is currently a vastly challenging problem that lacks a satisfactory solution. However, the lack of any established, uniform and generally accepted security system for mobile

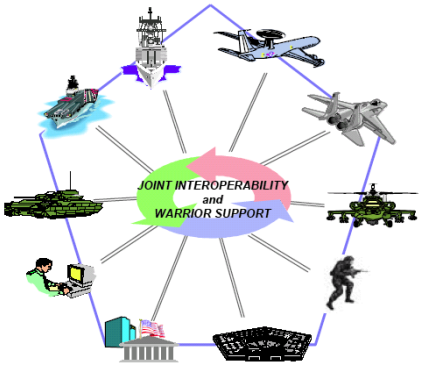


Figure 1. A Joint Service Battle Command is an intended future application for intelligent agent systems.

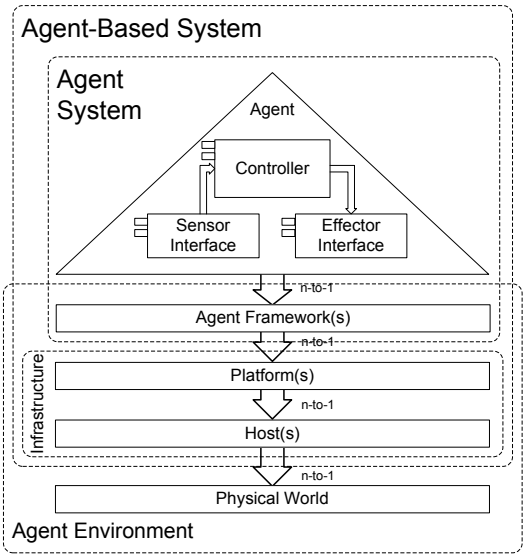


Figure 2. An Agent-Based System is made up of layers.

agents does not prevent the reference model for including a description of the security functions and facilities that an agent-based system may provide.

2 What is a Reference Model?

A *reference model* provides appropriate abstractions for facilitating adoption, adaptation and integration of evolving technologies [6]. Any generic model that has specific examples can be considered to be a reference model. Reference models are known to play a key role in understanding a given domain, establishing the domain as a scientific discipline, facilitating collaboration and promoting competition towards maturing technology relevant to the domain. Reference models emerged since the 1980s as a result of the success of the Open Systems Interconnection seven-layer reference model (ISO/IEC 7498-1:1994: Open Systems Interconnection Basic Reference Model) [3] [4] [5] that revolutionized the way communications systems developed. This model was developed as an International Standards Organization (ISO) effort. Another successful example of a reference model that has been developed by the ISO is one for archiving systems and is known as the Reference Model for an Open Archival Information Systems (OAIS) (ISO 14721:2003). In motivating and developing a reference model for agent systems, we draw heavily on these examples of existing successful reference models.

As these existing reference models demonstrate, reference models do not prescribe how functions and systems should be implemented. Instead, reference models provide the patterns of the solution for transforming vague notions into real-world implementation. Reference models simplify problem solving, to enable others to practice their discipline with a solid foundation. Software professionals, in particular use reference models to better understand abstractions and their potential for reuse.

3 Why a Reference Model?

Reference models are a necessity in a confusing, rapidly changing technology environment. As noted in [6], reference models are becoming more commonplace in fields of various human endeavors. The power of compelling reference models of knowledge can not be underestimated as a tool for technical leadership facilitating conclusions from a sound understanding of the problem space and solution domain. Thus, we argue that a reference model would be very useful tool to identify, assess and facilitate R&D and acquisition of agent technology for a wide variety of applications.

One particular application of agent-based systems is in military domains. Agent-based systems are being proposed to enhance and automate applications for collecting, presenting, storing, producing and sharing domain information. A future force is envisioned to be highly autonomous, modular, scalable, and flexible through the agentization of applications and services. This is especially true for complex system of systems (SoS). SoSs are large-scale, net-centric and include a variable mix of multi-department heterogeneous intelligent agents, humans-in-the-loop, and unmanned autonomous components and subsystems. Examples of complex SoSs in the US Army are the Army Battle Command System of the Current Force [1] and the Future Combat System of the Future Force [2]. Systems that would be found in a Joint Service (Army, Navy, Marine and Air

Force) SoS are depicted in Figure 1.

Heterogeneous intelligent agents promise to enable conflict resolution between and among applications and services engaged in competition, negotiation, mediation and arbitration, to assist humans-in-the-loop, and to control unmanned autonomous systems and robots. Agents are anticipated to play an important role in realizing dynamically varying mixed initiative capabilities to command and control manned as well as unmanned assets. As systems become increasingly complex, modularity as promoted by agent technology will become the key to reuse, scalability, and an open architecture. In addition, these design goals are a key to a manageable and affordable transformation from current to future force capabilities.

A reference model for intelligent agents would motivate the benefits of the technology by formalizing key concepts of both behavior and structure essential to enable agent technologies to reduce information collection, storage and sharing latency, workload, presentation overload and clutter for Battle Commanders and their staff. The recognition of the need and the investment to develop a unifying ontology for intelligent agent software across the domains of the systems in an enterprise-wide System of Systems (SoS) are shown to be crucial if not pivotal to the success of such SoS engineering efforts which are inherently multi-disciplinary and collaborative.

4 Towards A Reference Model

We describe a preliminary, partial reference model for agent-based systems. The portion of the model we present here is a high-level view organized as a set of layers. A more complete description of the current reference model is given in [7].

An **Agent-Based System** is comprised of agents and their supporting framework and infrastructure which provide fundamental services and operating context to the agents. Our model defines framework, platform and host layers, which mediate between the agents and the external environment. This layered model can be organized vertically as shown in Figure 2. Each layer is described as follows:

- The **Agents** layer consists of agents that perform computation, share knowledge, interact and generally execute behaviors in order to achieve application level functionality. We make few assumptions about the Agent layer except to state that agents are *situated* computational processes—instantiated programs that sense and effect an environment in which they exist. We make no assumptions about the internal processing structures of an agent. An agent could be built with a complex cognitive model or it could be a simple rule-based system. Given the vast array of tasks envisioned for agent systems, it is not the role of a reference model to limit or define what an agent is.

- The **Framework** layer provides standardized functionality specific to supporting agents. A framework typically provides support services such as conflict management, directory and naming services, security, and agent administration services such as monitoring and allocating resources to the executing agents. Figure 3 shows some examples. The major benefit of employing an agent framework is to provide standardization of services and functionality to agents that exist within the framework. The end result is that agents written within a particular framework are easily interoperable with one another. In other agent-based systems, the framework may be trivial or merely conceptual, for example if the services are merely a collection of system calls or are compiled into the agents themselves.

- The **Platform** layer provides more generic computing infrastructure. The platform contains the software components that are available to the agent framework, but are not packaged along with it. Some examples are shown in Figure 3. Elements such as operating systems, user interface libraries, database software, device drivers, and message transport or socket libraries are in this layer. These services are often provided by third parties and it is unlikely that an agent-based system will provide its own implementation of the platform functions.

- The **Host** layer contains the hardware devices on which the above layers operate. This layer includes not only the physical computing devices such as a desktop computer or hand-held device, but also the hardware that provides interaction with the environment such as robot sensors and effectors, cameras, displays, GPS receivers, etc. Some examples are shown in Figure 3

- The **Physical world** layer encapsulates the physical environment in which the agent-based system exists and operates.

Each layer can support many entities from the layers above it—many agents may execute on a single framework, many frameworks may execute on a single platform, and so on.

Figure 4 depicts an alternative view of an agent-based system. The figure organizes the interactions and communications between entities (agents, frameworks, hosts) at varying levels of abstraction. At the lowest level (physical network), hosts can communicate over a physical transport medium which can be either wired or wireless. A wireless communication medium is shown in our figure as an example. Going one level higher, connectivity between hosts

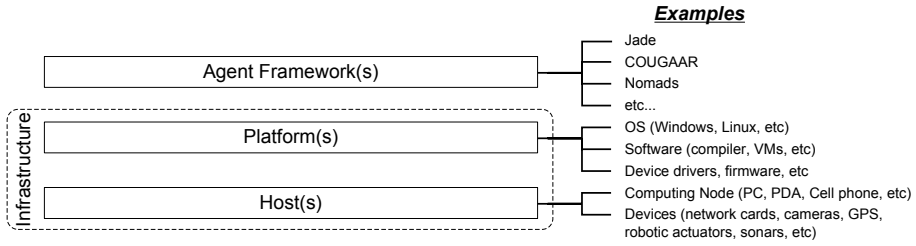


Figure 3. Framework, Platform and Host layers with examples.

is depicted at the network layer. Connectivity at this layer is determined by routing, naming/addressing or other network services. At the third higher level, we show the connectivity between agent framework instantiations on different hosts. Each host is now represented by a larger dashed oval and each framework instantiation is represented by a smaller solid oval. By showing framework ovals inside a host oval, we depict multiple frameworks running on a given host. Lines connecting the frameworks instantiations show that they can communicate to share information and services. Finally, at the highest level we show the communications that occur between framework instantiations of the same type (e.g., Cougaar, JADE, etc). In this way, we can depict that agents with common application goals can form societies that communicate in order to provide specific application functionality.

5 On-going Work

Our goal is to continue refinement of the reference model in consultation with the broader agents community. We will be making available an on-line interactive discussion forum, similar to a Wiki, for facilitating further development of the ABSRM.

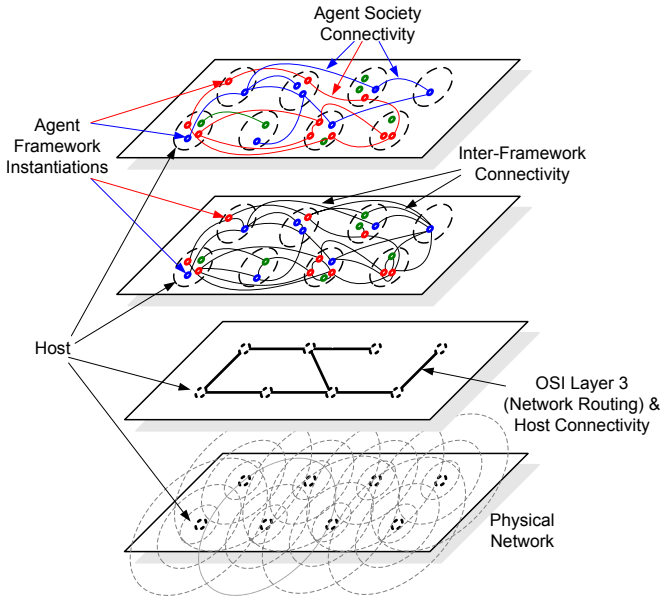


Figure 4. A Communications-oriented view of an Agent-Based System

Acknowledgments

We thank Christopher Dugan, Moshe Kam, Joseph Kopena, Robert Lass, Spiros Mancoridis, William Mongan, Jeff Salvage, Evan Sultanik, for significant contributions to this work. We also thank Tedd Gimber, Bernard Goren, Michael Huhns, James Odell, Randy Reitmeyer and Todd Urness for useful discussions.

References

[1] Army battle command systems (abcs). http://www.fas.org/man/dod-101/sys/land/abcs.htm.

[2] Future combat system (fcs). http://www.fas.org/man/dod-101/sys/land/fcs.htm.

[3] *ISO 7498(Draft): OSI Basic Reference Model*. 1984. Information Processing Systems.

[4] Uyless Black. *OSI: A Model for Computer Communication Standards*. Prentice Hall, 1991.

[5] B. N. Jain and A.K. Agrawala. *Open Systems Interconnection: Its Architecture and Protocols*. McGraw-Hill, 1993.

[6] Raphael Malveau and Thomas J. Mowbray. *Software Architect Bootcamp*. Prentice Hall, 2001.

[7] P.J. Modi, S. Mancoridis, W. Mongan, W. Regli, and I. Mayk. Towards a Reference Model for Agent-Based Systems. In *Proceedings of Autonomous Agents and Multiagent Systems (Industry Track)*, 2006.

Market-Based Collaborations for Unmanned Air Vehicles Operations

Yi-Liang Chen, Brian Gregory, Matt Easley, Mark Peot, Joseph Lee, Thomas Altshuler
Rockwell Scientific Co., 1049 Camino Dos Rios, Thousand Oaks, CA 91360, USA
{ylchen, bgregory, measley, mpeot, jlee, taltshuler}@rwsc.com

Abstract

UAVs are a key element of the U. S. Army's vision for Force Transformation. UAVs will be employed in large numbers per Future Combat System (FCS) Unit of Action (UoA). This necessitates a multi-UAV level of autonomous collaboration behavior capability that meets RSTA and other mission needs of the FCS UoAs. The Autonomous Collaborative Mission Systems (ACMS) is an extensible architecture and behavioral planning / collaborative approach to achieve this level of capability. We present a market-based approach that we developed as the main mechanism for autonomous collaboration in the ACMS. To enable flexible collaboration among a variety of heterogeneous unmanned vehicles for a broad range of missions, this market-based collaboration approach adopts a two-stage task specification and negotiation process that can accommodate different mission planning and task allocation strategies. We describe our market-based approach, its main features, and the collaboration protocol in this article.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are a key element of the U. S. Army's vision for Force Transformation, and will be employed in large numbers per Future Combat Systems (FCS) Unit of Action (UoA). The large-scale deployment of various types of UAVs necessitates a multi-UAV level of autonomous collaboration behavior capability that meets RSTA and other mission needs of FCS UoAs.

Sponsored by U.S. Army Aviation Applied Technology Directorate (AATD) under the Unmanned Autonomous Collaborative Operations (UACO) program, Rockwell Scientific Company (RSC) leads a team of academic, government, and industrial partners to develop and demonstrate autonomous collaborative behaviors capabilities for multi-UAVs.

The objective of RSC's ACO project is to develop and demonstrate autonomous collaborative behaviors among a team of networked UAVs to achieve various missions. Our ultimate goal is to enable groups of UAVs to achieve mission level objectives with minimal human intervention by applying leading edge autonomous and collaboration technologies.

We present a market-based approach to enable flexible and effective autonomous collaboration among teams of heterogeneous UAVs to achieve a variety of behaviors and missions. This market-based collaboration mechanism is a key element in our software architecture for autonomous collaborative behaviors for UAVs, named Autonomous Collaborative Mission System (ACMS) [3, 2].

We adopted an *intentional* cooperation [10] strategy for collaboration, similar to other approaches used in multi-robot/multi-agent cooperation such as [7, 5, 12]. The emergent cooperation (i.e. swarming) approach was not selected because it excels at controlling large groups of homogenous agents working toward a common goal. UACO involves controlling groups of heterogeneous agents (UAVs) in a range of mission types. [6, 11]

This market-based approach for *intentional* cooperation enables flexible collaboration between UAVs in a range of mission types. The approach uses a two-stage task specification and negotiation process that allows high-level description of the mission/task parameters during the bid solicitation stage, and further refinement of the tasking details during the task allocation and confirmation stage. Contrary to most market-based collaboration approaches (e.g., [7, 12]), our approach does not tie to a specific planning or task allocation methodology. This allows dynamic selection of either a general planner or a "subject matter expert" planning and task allocation component optimized for a particular mission type during the execution of the ACMS to accommodate a wide variety of missions and behaviors.

This article is organized as follows. In the next section, we discuss the key features of the market-based collaboration approach that support its flexibility and effectiveness. We present an overview of the ACMS architecture and the components that participate in the market-based collaboration in Section 3.1. We describe the collaboration protocol and the task specification hierarchy in Sections 4 and 5, respectively. Section 6 presents some results of the collaboration approach from the software-only and hardware-in-the-loop simulations. We conclude in Section 7 with remarks on how to incorporate new missions and behaviors in our approach.

2. Market-Based Collaboration: Features of Our Approach

In this section, we discuss some unique features that empower our market-based approach with flexibility to effectively handle autonomous collaboration of heterogeneous UAVs for a variety of missions and behaviors.

First, we adopted a **"mission-centric" view** in implementing the market-based mechanism. Each mission request is handled by a dedicated component serving as the market facilitator/maker (i.e., the Mission Manager/Planner, see Section 3.2). This component serves as the focal point for the corresponding mission in performing duties such as bid solicitation, task allocation and refinement, and mission monitoring and it is independent from other missions. This one-market-for-one-mission approach is more flexible and robust than the fully centralized approaches (in which there is only one central market maker for all the missions) and yet is more efficient than the fully distributed approaches [7, 12] (where every participant can solicit bids and be the market maker).

We developed a **two-stage process for task specification and negotiation**. In the first stage of this process, a set of high-level task specifications (i.e, the Roles, see Section 5) that can achieve the mission objectives are developed for bid solicitation purpose. Participating autonomous entities (UAVs) then evaluate these high-level task specifications (Roles) and generate bids that contain specific information related to further refinement of the high-level Roles such as area entry points and scanning rates in addition to the costs. In the second stage, the Mission Planner evaluates all the submitted bids and selects the winning bids for the missions. It then further refines the task specifications for these winning bids and sends them to corresponding bidders to confirm the awards.

The two-stage process alleviates us from the restrictions and inflexibility of taking either a decompose-then-allocate or an allocate-then-decompose approach for mission planning and task allocation that many market-based collaboration schemes have suffered [12]. The Mission Planner does not need to know the capabilities and status of the potential bidders prior to sending bid solicitations. The planner may also optimize the task allocation and/or refine the tasks, based on bidder feedbacks.

This approach provides **great flexibility in instituting mission planning and task allocation techniques**. It is partially derived from the mission-centric implementation and two-stage task specification and negotiation process. It does not tie to a generic mission planning methodology because there is a dedicated and independent market facilitator for each mission request. Meanwhile, the two-stage task negotiation process allows detailed task specifications and planning to be delayed until mission-related information is received from bidders. Hence, for each mission, we can select a mission planning and task allocation methodology that is most appropriate for a particular mission type.

3. ACMS Components for Collaboration

In this section, we provide a brief summary of the Autonomous Collaborative Mission System (ACMS) architecture and then describe the components that participate in the market-based collaboration. For detailed discussions on the ACMS architecture design and its main feature, please refer to [3, 2].

3.1. ACMS Architecture Overview

The objective of the Autonomous Collaborative Mission Systems (ACMS) architecture design is to create a modular and expandable architecture that will ensure flexible and scalable autonomous collaborative operations for a fleet of networked heterogeneous autonomous platforms such as Fixed Wing UAVs and VTOL UAVs across a wide range of missions. Figure 1 illustrates the high-level functional system architecture diagram, and shows the main modules of ACMS: a (group) mission management module; a (single) autonomous entity management module; a (single) autonomous entity executive; and a world model. The white inner boxes are the primary components of a module. The roles and functionalities of these ACMS modules are summarized as follows.

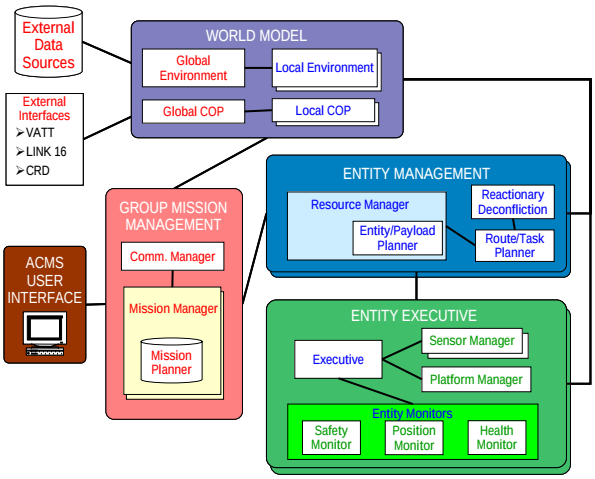


Figure 1: (Functional) Architecture for ACMS

(Group) Mission Management Module — provides mission planning, management, and monitoring for a group of entities and interacts with the ACMS user interface (if the group is the highest level entity) or the higher-level entity management module (refer to [2] for discussion on the hierarchical expansion of the ACMS).

Entity Management Module — (one module for each autonomous entity) participates in the market-based collaboration mechanism. It also plans and manages the behaviors and the resources of the corresponding entity.

Entity Executive Module — (one module for each autonomous entity) provides supervisory control and monitoring of task execution and the low-level control mechanisms for actual autonomous platform navigation and sensor control (if the entity is a platform).

World Model Module — provides information on the environment, ACMS entities, and other entities to ACMS components and other external components.

Interactions and interfacing among the components in the ACMS are through asynchronous communications of a set of XML-based messages that we extended from the Joint Architecture for Unmanned Systems (JAUS) [8] messages.

3.2. Components for Market-Based Collaboration

All the components in the ACMS are involved in market-based collaboration related activities to varying degrees. Three components that play the main roles in the market-based collaboration approach are the Mission Manager, the Mission Planner (both in the Mission Management Module), and the Resource Manager (in the Entity Management Module).

Adopting a "mission-centric" viewpoint (see discussion in Section 2), a new **Mission Manager** is instantiated when a new mission request is received in the ACMS to facilitate the collaboration, planning, and monitoring of the mission throughout the life span of the mission. The Mission Manager's role in our market-based collaboration approach is to implement the collaboration protocol (discussed later in Section 4) that facilitates the two-stage mission specification and negotiation between its Mission Planner sub-component and the Resource Managers of all the participating autonomous entities. It may also filter the potential candidate autonomous entities (e.g., UAVs) from which the bid is solicited based on the specifications provided by the Mission Planner.

The **Mission Planner** is a subcomponent of the Mission Manager. The main responsibility of the Mission Planner is to define high-level task specifications (i.e., the Roles, see Section 5) for bid solicitations given the specifications of the mission request. When the bids offered by the Resource Managers of participating UAVs are collected and forwarded by the Mission Manager, the Mission Planner then performs further analysis to select the winning set of the offers and further refine the tasks for the winner to fulfill the objectives of the mission. As explained earlier, our approach does not tie to a specific planning or task allocation methodology. Hence, the Mission Manager may select a particular Mission Planner implementation from a library of different implementations to instantiate the one that is the most appropriate (and may be customized/optimized) for the given mission request. The Mission Planner may be triggered by the Mission Manager to re-plan/re-task for the mission, if the circumstance warrants so during the execution of the mission.

The **Resource Manager** of each autonomous entity (e.g. UAV) participates in the market-based collaboration mechanism by evaluating bid solicitations from Mission Managers. It formulates the bids and their associated costs by checking its schedule and resource status for availability. The Resource Manager then consults with both the Route Planner

and the Entity/Payload Planner to determine route feasibility. If the Mission Manager awards a bid, it will then re-evaluate the refined tasks provided by the Mission Manager/Planner prior to confirming the award of the tasks for execution. We note that, since the Resource Manager can re-evaluate the awarded bid prior to acceptance for execution, the Resource Manager can take either a conservative (i.e., reserve the required resources) or aggressive (i.e., only check for the availability but not reserve the resources) stance in offering the bids to different missions.

4. Collaboration Protocol

In this section, we describe the collaboration protocol among the main components of our market-based approach described in Section 3.2. The collaboration protocol is based on the general steps of the first-price-one-round auctions [9] with extensions to incorporate the two-stage task negotiation process. Interactions outlined in the protocol are realized, similar to all other interactions in the ACMS, by a set of XML-based asynchronous messages. The steps in the collaboration protocol are as follows:

Stage 1:

Bid Solicitation: given the mission request and its specifications, the Mission Planner generates a set of general/high-level specifications of Roles (see Section 5) required for the mission. The Roles are sent by the Mission Manager to the Resource Managers of the (selected set of, if filters are applied) autonomous entities to solicit bids. T

Bid Response: the Resource Manager evaluates the bid solicitation received and formulates the bids in response to the solicited high-level role. We adopt a multi-aspect cost breakdown in the bid, where costs may be specified per specific action, time consumed, and resource consumed. Compare to most of the other approaches that adopt a fixed (single) cost breakdown [7, 12], this allows the bids to be evaluated by a variety of criteria, possibly optimized by the Mission Planner for the specific mission. The bids may also contain specific information related to further refinement of the high-level Roles needed for the Mission Planner. Multiple bids may be proposed by the Resource Manager in response to the same solicitation.

Stage 2:

Task Reservation: the Mission Planner evaluates all the bids received using the multi-aspect costs proposed and other information provided in the bids and select a set of bids that can satisfy the mission specification. It then refines the task descriptions to the entities to be participated. The detailed task description will then be sent to the Resource Managers of these winning entities for task reservation. Upon receiving the task reservation request, the Resource Manager re-evaluates the refined tasks provided by the Mission Manager/ Planner for feasibility of execution and reserves the schedule and resource for execution.

Task Commitment: If all the tasks required for the mission are reserved successfully. The Mission Manager will signal to the Resource Managers of the tasked entities for execution of the tasks for the mission.

Bid/Task Cancellation: the Mission Manager may signal to the Resource Manager to cancel the outstanding bids if they are not selected for the missions. It may also cancel the reserved tasks, if re-planning of the mission or re-allocation of tasks is needed.

5. Task Specification Hierarchy

In this section, we describe the breakdowns of the task specifications that are used in the two-stage task negotiation process.

As illustrated in Figure 2, we specify the tasks for the mission in four different levels of abstraction. At the highest (coarsest) level, the Mission contains the objectives, constraints, and other parameters specified in the mission request received from the human users. At the second level, a high-level plan generated by the Mission Planner consists of a set of Roles that are required to achieve the mission objectives. The Roles are used in the first stage (i.e. bid solicitation and response) task negotiation process. During the second stage, Roles are further refined by the Mission Planner into a set of detailed Tasks for task reservation and confirmation. Finally, the Resource Manager, upon task commitment, will further refine Tasks into Events and Activities for execution. In the following, we discuss Roles and Tasks that are key elements to market-based collaboration.

Taking analogy from the robotic soccer paradigm [1], the Roles are high-level descriptions of the “types” of the “players” required to achieve the mission. The number of the Roles required may vary from mission to mission and may not always correlate well with the

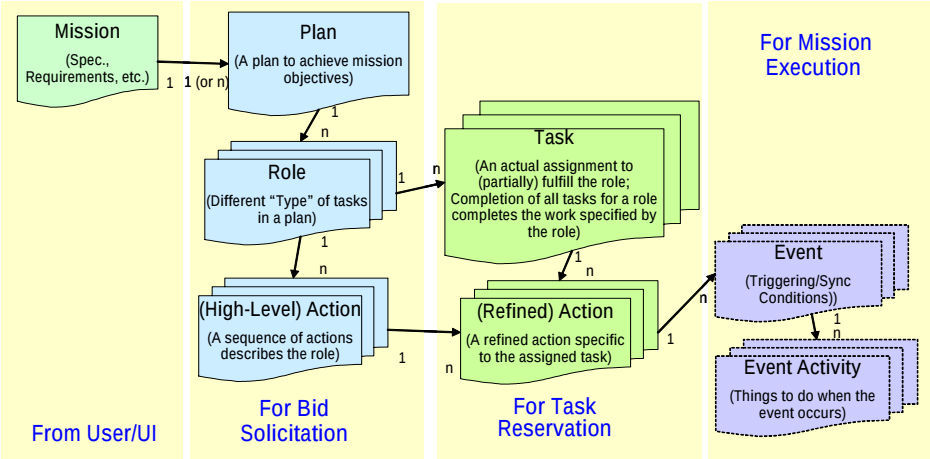


Figure 2: Task Specification Hierarchy

mission complexity. For example, an area surveillance mission may require only one general Role (i.e., "scanners") while a SEAD (Suppression of Enemy Air Defense) mission may have several Roles (e.g., "Wild Weasels", "sensors", "shooters", etc.). Each Role is defined by a sequence of high-level Actions. The high-level Action describes the jobs, timing, and resource required for each phase of the Role.

A Task is a further refinement of a Role generated by the Mission Planner to be assigned to an autonomous entity. We note that, the two-stage negotiation process enables the Mission Planner to prescribe one or more Tasks to be assigned to fulfill a Role, depending on the given mission and the bids received, For example, in an area surveillance mission, a Mission Planner may either assign two small UAVs or one more-capable UAV for the scanner Role based on the mission constraints and the bids received. Similar to a Role, each Task is defined by a sequence of detailed Actions. These detailed Actions are refined from the high-level Actions of the Roles with specific parameters and constraints customized for the assigned tasks and autonomous entities.

6. Simulation Results

The ACMS and the market-based collaboration approach are being implemented through a spiral development process for gradual and systematic build-up of the collaborative behaviors. Behaviors in each spiral are experimented with and verified through a series of software-only and hardware-in-the-loop simulations.

Figure 3 shows the hardware-in-the-loop simulation results of the collaborative reconnaissance behavior. In this behavior, an area surveillance mission request was sent to the ACMS. The Mission Planner created a plan consisting of a single high-level Role for area scanning. The Roles were sent out by the Mission Manager to a team of three UAVs to solicit bids. The Resource Managers of the UAVs submitted the bids. The Mission Planner evaluated the bids and, in this example, selected the bids from two UAVs (Figure 3) for awards. The Mission Planner refined the Tasks for the two UAVs, including their assigned areas and timing constraints and sent the Tasks to the Resource Managers for reservation. The Resource Managers reserved the Tasks and generated the flight plans for the UAV via the help of its Entity/Payload Planner and Route Planner. The third UAV (in the center-lower half of the figure) was not tasked for the mission. The Mission Manager canceled its bid. The resulting flight routes (including the idling UAV) are shown in Figure 3.

For more results of the ACMS and our market-based collaboration approach, see [4].

7. Concluding Remarks

In this article, we presented a market-based approach, developed in the ACO program, to enable flexible and effective autonomous collaborations among teams of heterogeneous UAVs to achieve a wide variety of behaviors and missions. We discussed the key features of the approach and provided an overview of the ACMS components that participate in the approach. We presented the collaboration protocol

and task specification hierarchy that incorporate the flexible two-stage task negotiation process. We presented some simulation results on this ACMS and the market-based approach. We plan to present further analysis of our approach in the subsequent reports.

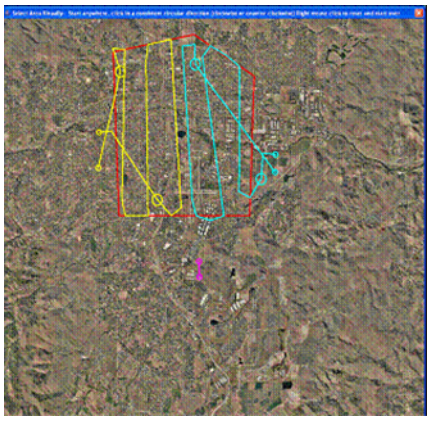


Figure 3: Market-based planning for collaborative reconnaissance behavior.

Finally, we note the steps to incorporate new missions and behaviors. First, given the new mission/behavior type, we determine whether a general (or existing) implementation of the Mission Planner could be used for planning and task allocation or a new implementation that can better optimize the mission may be introduced. Once the Mission Planner is selected, we then check to see if the high-level roles needed for the mission can be constructed using the existing library of high-level Actions. If not, new high-level Actions must be introduced and incorporated into the Mission Planner and the Resource Managers of the related autonomous entities. Finally, we check to see if the new high-level Action can be refined using the existing library of the detailed Actions. If not, we introduce the new detailed Actions and incorporate them into the corresponding Mission Planner and Resource Managers.

8. Acknowledgement

This article was supported in part by U.S Army/AATD under contract number W911W6-04-C0056.

9. References

1. J. Bruce, M. Bolwing, B. Browning, and M. Veloso. Multi-robot team response to a multi-robot opponent team. In Proc. ICRA Workshop on Multi-Robot Systems, 2002.

2. Y.-L. Chen, M. Peot, J. Lee, and T. Altshuler. An extensible architecture for collaborative networked unmanned air vehicles operations. In Proc. SPIE European Defense and Security Symposium, September, 2005.

3. Y.-L. Chen, M. Peot, J. Lee, V. Sundareswaran, and T. Altshuler. Autonomous collaborative mission systems (ACMS) for multi-UAV missions. In Proc. 2005 SPIE Defense Transformation and Network-Centric Systems, 5820:152-159, May, 2005.

4. Y.-L. Chen, M. Peot, J. Lee, V. Sundareswaran, and T. Altshuler. Autonomous collaborative behaviors for multi-UAV missions. To appear in Proc. 2006 SPIE Defense and Security Symposium, May, 2006.

5. S. Chiu, Y.-L. Chen, et. al. Distributed diagnostic and reconfiguration for shipboard chilled water system. In Proc. 13th Intl. Ship Control Systems Symposium, April, 2003.

6. J.-L. Deneubourg, G. Theraulaz, and R. Beckers. Swarm-made architectures. In Proc. European Conf. Artifical Life (ECAL), pp. 123-133, 1991.

7. B. P. Gerkey and M. J. Matarić. Sold!: auction methods for multirobot coordination. *IEEE Trans. Robotics and Automation.* 18(5):758-768, October, 2002.

8. Joint Architecture for Unmanned Systems. http://www.jauswg.org.

9. R. P. McAfee and J. McMillan. Auctions and bidding. *J. Economic Literature.* 25:669-738, June, 1987.

10. L. E. Parker. ALLIANCE: an architecture for fault-tolerant multirobot cooperation. *IEEE Trans. Robotics and Automation.* 14(2): 220-240, April, 1998.

11 H. V. D. Parunak, S. Brueckner, and J. Sauter. Digital pheromones for coordination of unmanned vehicles. In *Proc. Workshop on Environments for Multi-Agent Systems (E4MAS 2004)*, pp. 246-264, 2004.

12 R. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. In *Intl. J. of Robotic Research 25(1), Special Issue on 4th Intl Conf. on Field and Services Robotics*, January, 2006.

Negotiation-Based Approach to Unmanned Aerial Vehicles

David Šišlák, Martin Rehák, Michal Pěchouček, Dušan Pavlíček and Miroslav Uller
Gerstner Laboratory – Agent Technology Group
Department of Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic
sislakd@fel.cvut.cz, {mrehak|pechouc|pavlicd|uller}@labe.felk.cvut.cz

Abstract

We present a framework for agent based aircraft deconfliction mechanism to enable efficient airspace use by various UAVs during coalition operations. In our approach, each vehicle is autonomous, but cooperative: it actively shares its flight plan with near aircrafts so that potential collisions can be detected and resolved using norm-based system. Non-cooperative and utility-based deconfliction approaches are also discussed as they offer a possibility to achieve more efficient and robust mechanism in the future. System is validated on multi-agent simulation that uses the public online-accessible data from various information sources.

1 Introduction

In current coalition military and humanitarian relief operations, UAVs are deployed in growing numbers to provide intelligence and other services. They are attached to organizational entities in various levels of hierarchy, depending of their role, capabilities and operational requirements. Imposing a traditional, centralized approach to airspace management either requires extensive integration of heterogenous systems (at various levels of command and belonging to different coalition members), or severely restricts the agility of their deployment, inhibiting their primary advantage [2]. Therefore, we propose an alternative approach to airspace deconfliction, based on peer-to-peer negotiation between aircraft themselves.

Formally, we address the See & Avoid capability as specified in [1]. Our implementation of this capability is based on *active cooperative* approach (as defined in [1]), where the aircraft exchange basic flight data upon when approaching each other. In contrast to baseline active cooperative approach, we exchange not only the current position and bearing, but also the plans for immediate future. This operation enables efficient deconfliction using the cooperative negotiation mechanism described in Section 3. On the other hand, the requirement to share complete plans upon encounter makes interoperability more difficult and may be very difficult to implement in case of mini-UAVs. Therefore, other deconfliction strategies based on non-cooperative approaches and passive detection mechanisms shall be exploited, as we hint in Section 4.

Besides the above cited experiences [2], the current air traffic control methods based on rigidly structured airspace have shown to be inefficient even for the future coordination of the manned aircraft [9]. This is true mainly due to: (i) inefficiency of airspace utilisation that is based on fixed predefined flight corridors, (ii) increased air traffic workload given by ever increasing air traffic density and (iii) use of legacy technologies, that are in many cases 30 years old.

Due to the first two reasons of inefficiency listed above, the classical air traffic control methods are not very suitable for coordination of higher numbers of dynamically tasked aerial vehicles in densely used airspace. Therefore, our work is based on the *free flight* concept [7, 3] – an approach to autonomous routing of the aircraft and continuous flight trajectory adaptation complemented by the peer-to-peer deconfliction mechanism.

In Section 2, we present an air traffic simulator prototype developed to verify alternative approaches to airspace deconfliction.

2 Multi-Agent Flight Modelling

In order to verify the deconfliction mechanisms, we have developed a multi-agent framework for flight simulation, planning and visualization. Besides its primary verification function, as described in the remainder of this Section, this system can actually work as a centralized planner as well. Formally, this approach to planning solves the NP hard problem by application of negotiation heuristics. Our solution addresses the difficulties of integration of proposed mechanisms into the actual UAV hardware. In the central planning mode, the system executes high-level flight plans

as received from UAVs or their pilots and the agents simulating the aircraft perform the negotiation upon encounter, possibly altering their detailed flight plans. This "planning by simulation" mode can provide safe flight plans for high number of aircraft in matter of seconds. Deconflicted plans can be then used by real aircraft during their mission. In-flight replanning is painless – we fix the current aircraft positions and run the fast simulation again to obtain deconflicted plans. To address the specific needs of each mission, aircraft safety zones can be resized to allow the necessary maneuvers.

2.1 Multi-Agent System Architecture

The agent-based part of the designed prototype for the flight modelling runs on the $\mathcal{A}$-**globe** Java multi-agent platform [8]. Besides the functions common to most of agent platforms it provides *Geographical Information System* based on topic messaging. Therefore, the platform is ideally suited for testing experimental large scale, real world scenarios with fully fledged agents featuring agents' position, position dependent environment simulation and communication inaccessibility.

The multi-agent system for flight modelling consists of several components, see Figure 1.

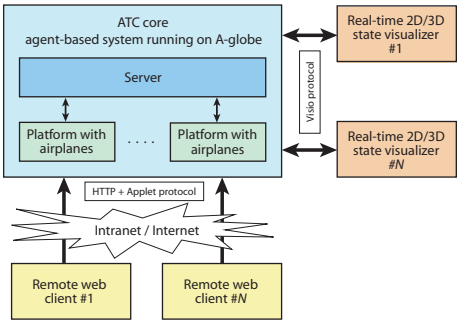


Figure 1. Multi-agent system framework

Server Component

The *server component*, figure 2, of the ATC core system is a sole central element of the system. It simulates positions of aircraft and other objects in the simulated world, aircraft hardware, weather conditions, communication ranges given by the ranges of board data transmitters, etc. It is also responsible for acquiring information about all airplanes and provides them to both types of visualizers. If the proposed distributed agent system for flying on deconflicted airways is used to control real aircraft, this *server component* could be removed from the system.

It consists of several environment simulation agents. *Configurator Agent* loads initial configurations from the specified configuration files and distributes them to other

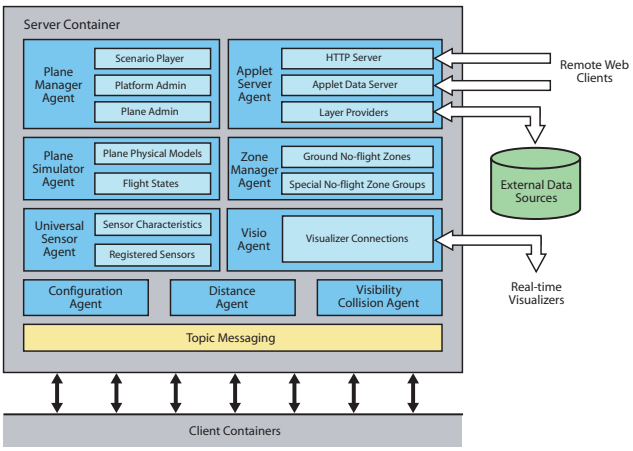


Figure 2. Server component architecture

agents. *Plane Manager Agent* administrates running aircraft containers, it spawns new airplanes and removes the existing ones, and assigns initial flight missions to the airplanes. The manager also acts as a load balancer. *Plane Simulator Agent* computes the current position of the aircraft in the simulated world. It contains all physical models for all plane types and keeps all current flight plans and states of the running aircraft. When a plane pilot agent changes some part of the flight plan, the change is propagated via the plane agent to the plane simulator agent in the form of a difference flight plan update. The agent can be asked for the current airplane position by the pilot agent. *Distance Agent* calculates Euclidian distances between each pair of existing aircraft using their current positions. *Visibility Collision Agent* prepares $\mathcal{A}$-**globe** visibility updates [8] for controlling communication restrictions between airplanes. The airplanes that have collided with any other object are uncontrollable and they fall down to the ground. Falling aircraft can endanger any airplane that flies under it. *Universal Sensor Agent* represents all radar sensors on aircraft boards. *Zone Manager Agent* transforms any defined no-flight zones including ground surface to a compressed octant tree. *Visio Agent* is an interface between the Core agent system and the real-time visualizers. *Applet Server Agent* provides a communication interface between the agent system and remote web clients.

Platform Component

The *platform component*, figure 3, is used as a registration unit for starting containers with agents simulating aircraft behavior inside running Java Virtual Machine (JVM). When flight-modelling system is used for planing/simulation of a huge number of aircraft, its architecture allows to use several host computers via platform component each running in its own JVMs. All running aircraft units are proportionally split between the registered hosts. This enables balancing of the overall load between all registered computers.

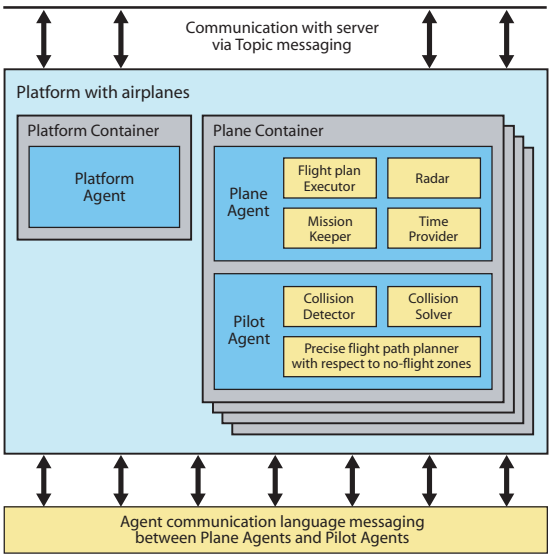


Figure 3. Platform component schema

Agents running on the *plane container* represents behavior of one simulated aircraft in the system. *Plane Agent* provides high-level airplane functions, such as flight plan execution in cooperation with the plane simulator agent, radar and detector readings, airplane configuration and time synchronization. *Pilot Agent* is the main control unit of the simulated aircraft with implemented deconfliction mechanisms.

2.2 Flight Path

The flight plan describes a trajectory which the plane follows during its flight, see Figure 4. A *waypoint* (end point of the segment) is an important navigational point used for rough definition of the flight route; it represents a certain location and also specifies the time interval when the airplane should fly through it. The waypoints serve as an input to the planner, which generates the precise flight path consisting of the segments and elements. The segments are composed of elements, which constitute the most basic parts of a flight plan with a simple geometry.

Flight path *planning process* generates a smooth and continuous path passing through all input waypoints respecting associated time constraints. The planning proceeds in three phases: the computation (composition of elements) of the actual path without regard to the time constraints, the adjustment of the flight plan to satisfy the time criteria and the replanning of the plan in the parts where it collides with no-flight zones (if defined). For the replanning there is implemented 2-phase A^* running over compressed octal tree with information about no-flight zones. First phase of replanning produces continuous subspace of the whole world

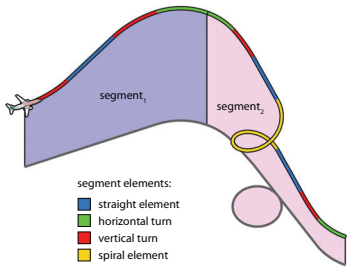


Figure 4. Flight plan, segments, elements

where second phase finds best suitable shortest path.

3 Deconfliction by Negotiation

The mechanism we propose is distributed by its nature, which is why the deconfliction technology (while developed within the multi-agent ATC model) is ready for deployment on autonomous vehicles without any central point of control.

The aircraft are modelled by agent containers hosting several agents. In this contribution we will be referring to agents representing an aircraft as auto-pilot. This agent is a self-interested entity that is in charge of (i) preparing a detailed flight plan for the airplane respecting time-specific waypoints for the airplane's mission and (ii) executing the detailed flight plan by performing the flight.

Each simulated aircraft is surrounded by a number of concentric spherical zones: **Communication**, **Alert**, **Safety** and **Collision** zones. The **communication zone** is the outermost one. It represents the communication range of the transponder and data transmitter onboard the aircraft. Using this data, the aircraft can send data packets to other aircraft that are positioned within the specified spherical zone defined by its radius. The **alert zone** defines the operation range of the radar onboard the aircraft. If another aircraft is located within the alert zone, the aircraft are periodically notified about its relative position and its flight code. The **safety zone** encapsulates the area around an aircraft that other aircraft are not allowed to enter in order to minimize protect the safe operation of each aircraft. The size of this zone is not constant, but is determined by the aircraft type, its current tasks and the environment, allowing higher degree of freedom when the environment is adversarial. Safety zone radius can change dynamically during the flight. If two aircraft do enter each other's safety range, they can still continue flying but their flight path may be influenced by e.g. turbulence. This is not the case when two or more airplanes fly together in a close formation. The **collision zone** is the innermost zone. It defines the critical contact area. When the mutual distance between two aircraft is smaller than the sum of their collision zone radiuses, the physical collision happens.

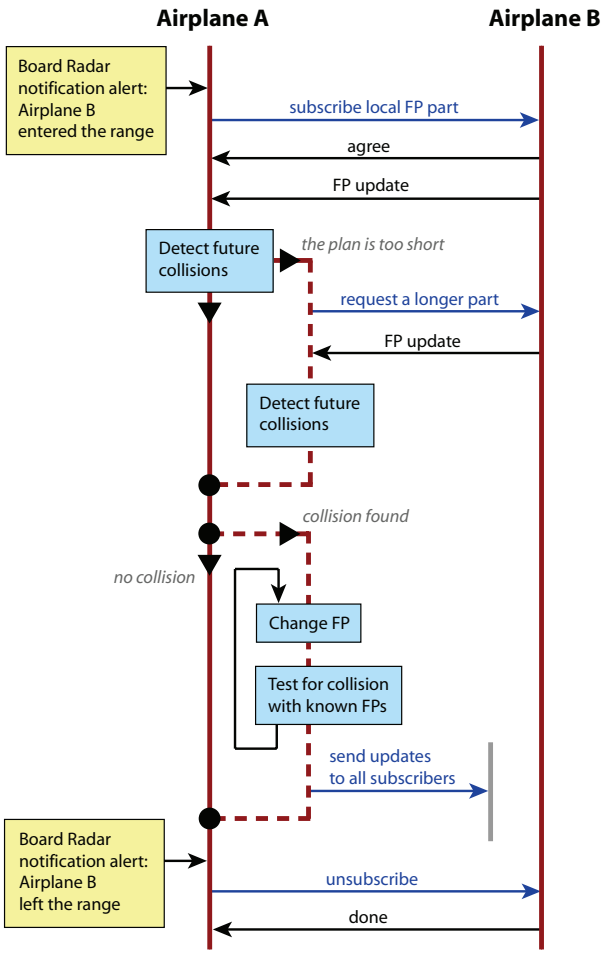


Figure 5. Negotiation protocol

Cooperative deconfliction. In the simulation phase the ATC system solves collisions *cooperatively* by negotiation between the auto-pilot agents, see Figure 5. Let us suppose an aircraft A to fly along its planned optimal flight path through its mission waypoints. An airplane B enters the *alert zone* of the airplane A. The pilot agent of the aircraft A is notified about its position and flight code by the on board radar system. The pilot agent of the aircraft A tries to establish negotiation connection with the pilot agent of B. In the case when the connection cannot be established or the communication is not trusted, the pilot agents should use *non-cooperative approach*, described later in this section. If the connection was established successfully, the pilot agent A subscribes for a local area flight plan of the aircraft B (representation of the flight plan is described in Section 2.2). The pilot agent of aircraft B sends an update to the subscriber every time it changes its own flight plan. The update contains the part of the flight plan of the aircraft for the specified amount of time depending on the flight speed. The update is also sent when the time span of the previous update was not long enough and it needs to be updated again. When the pilot agent A receives an update from the pilot agent B, it executes the **collision detection process** on its own flight plan and the received one.

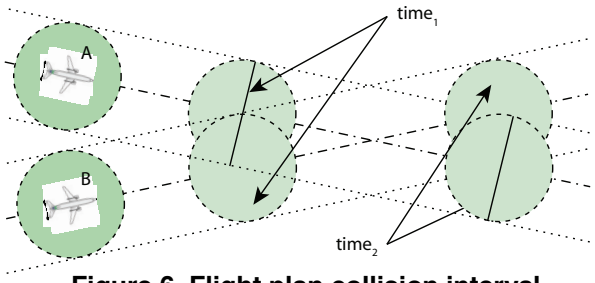


Figure 6. Flight plan collision interval

The collision detection process is an algorithm that analyzes two flight paths and tests the safety range violation. If there is a specific point in time detected when the distance between the positions of the aircraft is smaller than the maximum of the safety ranges of the aircraft (see Figure 6), the detection process returns $time_1$ and $time_2$ which represent the first and the last collision point between the two flight plans. This information is needed by the auto-pilot agent to handle the situation.

If the detection test is negative, both flight paths are mutually safe. If a collision is found, the airplanes A and B must modify their flight paths. The first prototype of the designed system uses a rule-based approach for modifying flight plans described in Section 3.1.

Non-cooperative deconfliction: The distributed deconfliction approach is open to extension towards *non-cooperative deconfliction*. The non-cooperative deconfliction is useful in situations when an airplane has a malfunctioning or incompatible transponder/transmitter/receiver on its board or in the situation when there is an intruder/enemy airplane with adversarial behavior [5] which intentionally sends incorrect future flight path parts to the others. The most suitable approach to the *non-cooperative deconfliction* is the game theory. In this case the pilot agent tries to change its own flight plan in a way that would guarantee a minimal collision risk for any conceivable future position of the other airplane. To determine all possible future positions of the other plane, information about its current position, direction and information about its type can be used. The monitored object's flight path is always continuous but there are also certain restrictions that depend on the airplane type – e.g. minimal/maximal flight speed, minimal radius of turning, etc. These parameters also influence the radius of the safety zone as used in the computation – the radius of the safety zone increases with the aircraft agility to allow more reaction time for replanning if it changes its direction in the future. When the pilot agent wants to identify whether

or not it should use the non-cooperative deconfliction for a particular airplane, it can integrate a special *detection module*. The detection module compares the active cooperative information [1] with the active and passive non-cooperative observation data provided by aircraft sensor and trusted aircraft and decides whether the aircraft can be trusted or not. Trust [4],as a measure of adversariality can help the aircraft to determine the appropriate safety range around its non-cooperative partner.

3.1 Rule-Based Collision Avoidance Mechanism

Upon each detected collision, its type is determined from the angle between the direction vectors of the concerned planes at $time_1$ projected to the ground plane (defined by X and Y axes), see Figure 7.

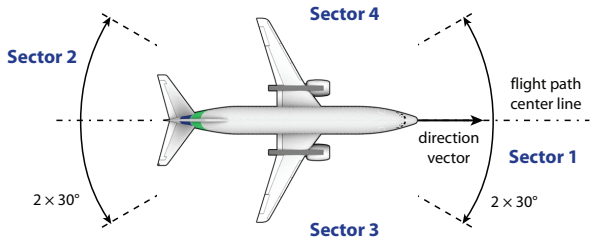


Figure 7. Identification of the collision type

Depending on the computed angle, the airplane `B` falls into one of four sectors surrounding the airplane `A`. Depending on that sector, one of the following rules is applied on the flight plan of the airplane `A` to avoid the collision:

- **Sector 1** – head-on collision, in this case the airplanes avoid each other by both of them turning to the right. The flight plan is changed as shown in Figure 8. The pilot agent shifts the plan points at $time_1$ and $time_2$ to the right, perpendicularly to the old direction vector. The length of the shift is equal to a minimum of safety ranges of both airplanes. Beyond $time_2$, the flight plan follows the shortest way to the next mission waypoint.

- **Sector 2** – rear collision, there are two subcases: i) the front aircraft is faster – airplanes do not change their current flight plans; ii) the rear airplane is faster – it has to change its flight plan so that it turns to the right and passes the front airplane without endangering it. The flight plan is similar to that in Figure 8. To achieve this, the rear airplane shifts its flight plan points at $time_1$ and $time_2$ to the right, perpendicularly to the old direction vector. The length of the shift is at least 1.1 times of the safety range.

- **Sector 3** – side collision, the airplane `B` has higher traffic priority. The aircraft `A` needs to slow down its speed so that it reaches the collision point at $time_1$ later than the airplane `B`. If this is not possible due to the minimal flight speed defined for each airplane type, the airplane `A` slows down as much as possible and shifts its flight plan point at $time_1$ to the right so that there is no collision between the two flight plans.

- **Sector 4** – side collision, the airplane `B` has lower traffic priority. The aircraft `A` changes its flight plan by increasing its flight speed so that it passes the collision point before the airplane `B`. The airplane `A` only accelerates as much as needed.

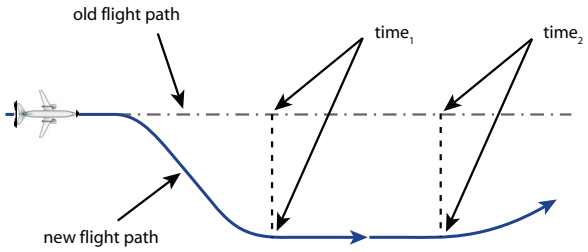


Figure 8. Change of the flight plan.

The above rule-based changes to the flight plan are carried out by both planes independently because each aircraft detects the possible collision with the other airplane from its own point of view. After applying the changes to the flight plan, the airplane sends an updated local flight plan part to all airplanes that subscribed for it. The change is also verified against all other known flight plans of all aircraft monitored by the board radar system. If another collision is detected, new collision is resolved.

The pilot agent internally uses the *flight plan wrapper* interface for manipulation with its flight plan. The requests for each plan modification are handled as a special set of solver time-constrained waypoints. A special handling algorithm takes care of the execution of each modification that overrides the previous one.

4 Conclusion

This contribution presents a multi-agent approach to airspace deconfliction mechanisms. We implement the active coopertive implementation of the See & Avoid capability, where we enrich the transponder data with flight plans to increase system robustness and minimize adverse long time effects.

To demonstrate the mechanism, we have developed a multi-agent framework described in Section 2. As in the future we aim to solve the multi-level deconfliction problems

with significant number of non-cooperative aircraft, we are currently extending the framework to include the near-real time information about the civilian air-traffic around major US hubs. As these aircraft will obviously not change their flight plans to accommodate simulated UAV traffic, UAVs will have to handle them as non-cooperative elements. Such traffic also provides a good test cases for the deconfliction algorithms – good algorithm is not only able to handle a large number of cooperative UAVs, but is able to perform real-time deconfliction in an already congested airspace.

Besides the flight data, the framework integrates several external data layers (see Figure 9): a mosaic of Landsat7 images, state boundaries, airports, populated places and powerplants used to define the no-flight zones. External data sources are imported from NASA, U.S. Geological Survey (USGS), Geographic Names Information System (GNIS) database and AirportMonitor. Positions of real air traffic is 10 minutes delayed.

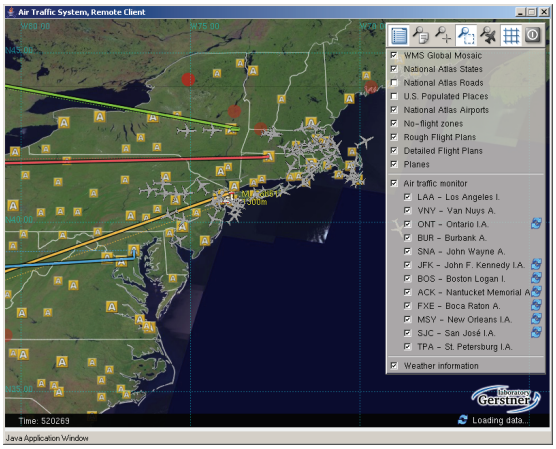


Figure 9. ATC system remote client GUI overview

The prototype currently relies on simple deconfliction rule-based mechanism for the flight plan and assumes that all airplanes use the same deconfliction rules. The system will be extended so that airplanes that detected future collisions would iterate through *monotonic concession protocol* (MCP) to find new flight plans that are collision free, obtaining better results. The monotonic concession protocol is a simple protocol developed by Zlotkin and Resenschein for automated agent to agent negotiations [10, 6]. Both airplanes prepare a set of possible flight plan changes scored by the utility function. The utility function includes pilot's own intentions including flight priority, fuel restrictions, time restrictions, etc. From all collision free combinations of flight plan pairs, the possible solution set is created. The iteration protocol results in a commonly accepted solution of the collision. Then each airplane applies the agreed flight plan changes. However, the iterativeness of solution can lead to a situation when the solution is not found fast enough. The process has to be extended with an emergency solution that is used when the iteration process does not lead to any fast solution. As the emergency solution, the game theory approach can be used.

Acknowledgement: Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-04-1-3044-P00001 extension of the FA8655-04-1-3044. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon[1].

References

[1] DOD. Unmanned aircraft systems roadmap 2005-2030, 2005.

[2] G. W. Goodman. Congested airspace. *C4ISR Journal*, (January), 2006.

[3] J. C. Hill, F. R. Johnson, J. K. Archibald, R. L. Frost, and W. C. Stirling. A cooperative multi-agent approach to free flight. In *AAMAS*, pages 1083–1090, 2005.

[4] M. Rehák, Lukáš Foltýn, M. Pěchouček, and P. Benda. Trust model for open ubiquitous agent systems. In *Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference*, number PR2416 in IEEE, 2005.

[5] M. Rehák, M. Pěchouček, and J. Tožička. Adversarial behavior in multi-agent systems. In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications IV: 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005*, number 3690 in LNCS, LNAI, 2005.

[6] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. The MIT Press, Cambridge, Massachusetts, 1994.

[7] R. Schulz, D. Shaner, and Y. Zhao. Free-flight concept. In *Proceedings of the AiAA Guidance, Navigation and Control Conference*, pages 999–903, New Orelans, LA, 1997.

[8] D. Šišlák, M. Rehák, M. Pěchouček, M. Rollo, and D. Pavlíček. $\mathcal{A}$-**globe**: Agent development platform with inaccessibility and mobility support. In R. Unland, M. Klusch, and M. Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.

[9] C. Tomlin, G. Pappas, and S. Sastry. Conflict resolution for air traffic management: A case study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, August 1997.

[10] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In N. S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 912–917, San Mateo, CA, 1989. Morgan Kaufmann.

[1]The views and conclusions contained herein are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.