Czech Technical University in Prague

Faculty of Electrical Engineering

Habilitation Thesis

January 2013 *Jiří Vokřínek*

Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Computer Science and Engineering

Contracting in Multi-Agent Systems

Habilitation Thesis

Jiří Vokřínek

Prague, January 2013

Dedicated to my beloved wife Petra,
and my amazing sons Michal and Aleš . . .

Acknowledgments

This thesis would not be possible without the strong support of all my colleagues that I have a pleasure to worked with in the last 10 years. I would like to thank all the to co-authors of included papers, mainly to Antonín Komenda, Michal Pěchouček, Jiří Bíba, Jiří Hodík, Jan Vybíhal and Jan Doubek. My special thank goes to all the partners of the projects that I have participated for providing a highly creative environment.

Abstract

Contracting in multi-agent systems is the key component for enabling agents cooperation in various environments. Standard means of contracting, i.e. task allocation, is supported by well-defined contracting protocols and agent strategies. In open heterogeneous environments, where the agents follow their private constraints and goals, there is a need for more complex contracting schemes. Such schemes are based on complex interaction protocols, private knowledge protection and commitments security.

The design of the contracting mechanism composes of (i) definition of the inter-agent interaction protocol and (ii) internal agent decision making design. Separation of these two concepts enables high flexibility in the multi-agent system design. A well-defined interaction protocol ensures feasibility and fairness of the contracts, while keeping privacy of the agent decision making process. Local agent decision making comprises of application dependent mechanisms, such as local optimization, planning or scheduling. The global result of the multi-agent contracting process can be interpreted as a distributed plan, a solution of an allocation or a scheduling problem. More generally, it can be interpreted as a set of social commitments that individual agents made. The commitments representation supports contract execution, monitoring and corrections in the case of an execution failure. In a cooperative environment, where the agents follow some shared agreed goals, the contracting problem is often reduced to distributed problem solving. In a competitive environment, the contracting problem is more oriented to identification of suitable partners and contract negotiation. The protocols used in competitive environment have to deal with execution failures, commitment corrections and potential decommitments with penalties as well.

The application scenarios of multi-agent contracting vary from production planning and scheduling, cooperative problem solving in logistics domains to formation of virtual organizations. In all these mentioned domains strong aspects of agent-based interpretation play an important role. Examples of such features are decentralized solution enabling distributed execution of agents, privacy of individual agents, openness of the system, and heterogeneous agent strategies and their internal goals and metrics.

This work summarizes advances in task-oriented problem solving and applicability in manufacturing and production planning. It is followed by an introduction of a contracting protocol oriented to the domain of virtual organizations and a contracting model minimizing private knowledge disclosure. The plan representation based on the social commitments and presentation of study of commitments stability during plan execution concludes the thesis.

Contents

Chapter 1

Introduction

A cooperation between agents is defined as a provider-customer relationship with defined conditions – i.e. the provided service (of proper quality), the price, the due-date, penalties, etc. An established cooperation is confirmed by a contract concluded by both the sides. The utility gained by each participant in the contract is given by the conditions of the cooperation and each participant's current state.

Agents share their knowledge and dynamically form teams to achieve their goals [17]. Such goals do not need to be necessarily related or compatible [2]. An agent also keeps knowledge about other agents as well as the level of confidence in that knowledge. In the area of multi-agent systems the concept of clustering individuals into cooperating groups is often used, for example an alliance and a coalition [43]. The alliance is a collection of units that share information about their resources and all agree to form possible coalitions. The alliance is regarded as a long-term cooperation agreement among the units. The coalition is a set of units agreed to fulfill a single, well-specified goal. A coalition, unlike an alliance, is thus usually regarded as a short-term agreement between collaborative agents.

The modern cooperation concepts inspired by industrial domain go from the subcontracting, through the supply chains to Virtual Organizations (VO) [38]. Collaboration between companies is needed for businesses that cannot be executed by individual companies on their own [9, 59]. In business-to-business e-commerce, a group of collaborating partners may act as a single company and thus create a more competitive whole [12]. The commonly referred work of Davidow and Malone from early 90's suggests Virtual Corporations as an industrial strategy for the twenty-first century [13]. Though innovative, it introduces a logical continuation of existing collaboration strategies [60].

The agent-based representation of collaborating entities enable to model cooperation processes from a global point of view while the privacy and autonomy of the actors are secured. The problems like team formation, alliance creation, task allocation or distributed planning can be modeled as contracting problem in multi-agent system. There exist various methods of negotiating and coordinating of agents' actions. Lomuscio et al. defines negotiation as "...the process by which a group of agents communicate with one another to try to reach agreement on some matter of common interest." [34]. They define two components of the negotiation mechanism: the negotiation strategies and the negotiation protocol. The former one defines lists of actions of individual agents that they have planned to reach their desires. The latter one (the protocol) defines rules for messages that are allowed in the message sequence. To enable the cooperation using particular contracting schema or protocol the agents still have to challenge the issues of *(i)* identifying potential partners for collaboration and *(ii)* information sharing between potential partners while keeping maximal possible privacy of their internal data, intentions and goals [27].

An overview of the contracting problem in multi-agent systems is described in Section 1.1 followed by an introduction to Virtual Organizations concepts in Section 1.2. An introduction of the most important contracting protocols is given in Section 1.3. The issues of information sharing are addressed in Section 1.4. This chapter is concluded by the case studies documenting the applicability of the introduced research advances in Section 1.5. Sections 1.2 and 1.3 are mainly based on [52], which is also included in Appendices page 55. Section 1.4 is based based on [25] and [14], which is also included in Appendices page 83.

1.1 Contracting in Multi-agent Systems

The problem of controlling entities in a heterogeneous distributed environment is crucial for many domains [16]. Classical centralized methods depend on one central planning system. Such a system gathers all required input data before the planning process takes place. Then the plan (a set of plans respectively) is generated using these data. This approach faces various problems. One problem is the need for private local knowledge of the actors. The other problem is the need for real-time replanning based on environments and conditions changing dynamically over time. On the other hand, in distributed methods of planning, each entity plans its own plan. Cooperation and heading towards common goals is done by various methods of negotiation. Distributed planning has been viewed as either (*i*) planning for activities and resources allocated among distributed agents, (*ii*) distributed (parallel) computation aimed at plan construction or (*iii*) plan merging activity.

A multi-agent system design is composed of two components: *(i)* local agent algorithms design that respects autonomy, individual constraints, goals and resources of agents and *(ii)* inter-agent interaction schemes that provide social aspects of the whole system and support (emergent) macro behavior of the multi-agent system. Agent interactions are motivated by cooperative solving of a given problem (even in the competitive environment). Similarly to social aspects referred to as comparative advantage in economy [37], where a group of individuals cooperates on the delivery of a service or goods at a lower opportunity cost than other groups, the agent community tries to find a solution maximizing social welfare [4]. Agent theory recognizes various types of agent environments based on various points of view. Basically, the nature of the agent environment affects (or defines) the behaviour of members of an agent community operating within the environment. A cooperative problem solving has been formalized by Wooldridge and Jennings by means of social commitments in [61]. One of significant points of view along which agent environments may be distinguished is a *primary motivation* of the agents to engage a cooperative problem solving (i.e. a mutual cooperation) [57].

There are two most often recognized environments as the differences between them are significant in many ways: *collaborative* and *competitive*. Agents in a *collaborative* environment are motivated primarily by a common interest in maximizing their social welfare[1] while agents in a *competitive* environment maximize rather their individual utilities, no matter what the social welfare is (the agents are so called *self-interested*) [46, 43, 6]. As shown in [57], the welfare maximization can be transformed to minimization of the cost of assignments of tasks (sub-problems) to individual agents. This cost is computed locally using planning algorithms of individual agents. The global overall solution is minimized using interactions between agents – task allocation, delegation and reallocation. The allocation of a task to an agent is represented by a social commitment that an agent undertakes [31, 32]. Social commitments in both coopera-

[1]A social welfare is usually defined as a total sum of individual utilities of all members of the observed agent community.

tive and competitive environment also differ substantially as well as their possible evolution. In both cases, the commitment representation provides a powerful tool for task execution stability and performance in dynamic and/or uncertain environments. The commitment-based approach enables to change the solver-centric point of view to a more task-centric point of view and enhance the solver operations for heterogenous commitments in dynamic environments. Most of the multi-agent systems designed and implemented for planning support operates mainly in cooperative environment. In such setting the contracting algorithms follows well defined protocols and the objective function of the algorithms are mostly based on cost minimization [57, 54].

The multi-agent systems supporting automated or semi-automated cooperation and coordination of independent individuals (such as Virtual Organization) operate in rather competitive environments. A utility function of the each agent can differ and the goals do not need to be compatible. A negotiation protocol used should cover all the aspects of competitive domain, such as penalty and decommitment negotiation [52, 6].

Cooperative and Competitive Environments

Let us introduce a difference between a collaborative and a competitive multi-agent environment [3]. By a *collaborative multi-agent environment* we understand an agent community, where the agents usually share a common goal which they try to achieve cooperatively. In other cases the agents may have different goals, but their primary motivation is a maximization of their social welfare – the total sum of all the individual utilities (profits) of the collaborative agents. On the contrary, by a *competitive multi-agent environment* we understand an agent community, where the primary motivation of the agents is a maximization of their individual utilities; no matter what the social welfare of the community is (agents are so called self-interested). The agents establish a cooperation on the process of achieving a common goal only if it contributes to maximization of their individual utilities. The willingness of the agents to keep the agreed contracts also differs in both cases [6].

In a collaborative environment, the agents keep the contract as long as the social welfare is maximized. When the social welfare goes down or a better collaboration opportunity arrives, the agents either freely withdraw from the contracts or are willing to reconfigure the contract. A collaborative behavior of all agents ensures maximal social welfare after decommitment/reconfiguration. No penalty is charged in this case because both provider and customer agree with the decommitment or reconfiguration. On the other hand, in a competitive environment, the contract is secured by penalties to be paid by the agents in case of decommitments or an other breach of the contract. The contract is kept as long as the individual utilities of all parties are maximized. A feasibility of an eventual reconfiguration is then substantially conditioned by the utilities as well.

Social Commitments

An evolution of *collective commitments* has been inspected by Dunin-Keplicz [15]. The analysis of commitments and their evolution within a group of cooperating agents has been based on an assumption of persistency of their joint intentions. This assumption is valid in *collaborative* environments, but does not necessarily hold in *competitive* environments. The basic flaw consists in the fact that *self-interested* agents do not need to share the joint intention unconditionally. As the primary motivation of *self-interested* agents is a maximization of their individual utilities, any of them tends to drop the joint intention and breach the cooperation as soon as a better opportunity for a maximization of its individual utility appears. Such opportunity does not need to be recognized as the best one by all the agents involved in a cooperative problem solving as

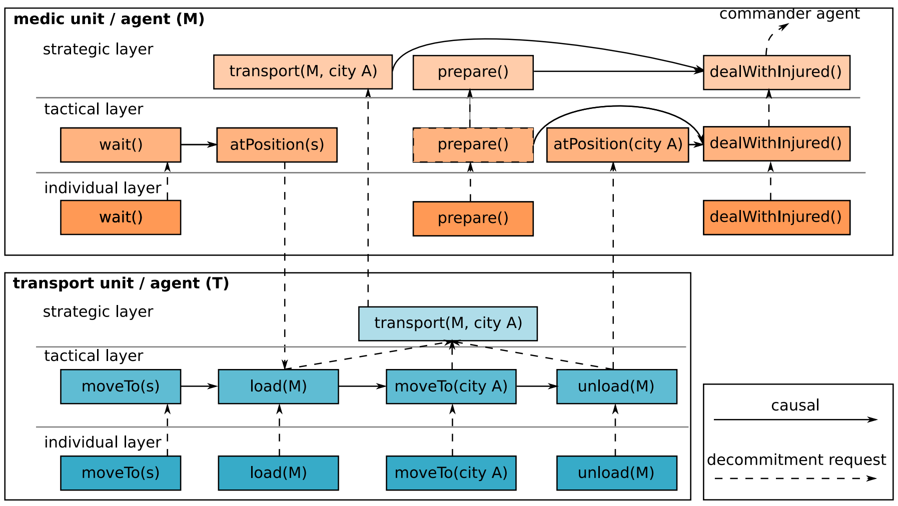


Figure 1.1: Commitment based plan representation example for two agents [32].

well as it does not need to be known to any but one agent in the cooperating group.

While social commitments in *collaborative* environments could be *blind* under a common rationality restriction (i.e. the commitments are never dropped unless achieving a commonly shared goal becomes unreasonable or mutually disadvantageous for all), Sandholm and Lesser identify such commitments (referred to as *full commitments* [46]) as unsuitable and unreasonable in *competitive* environments and introduce a new concept of *leveled commitments* [46]. Basically, the agents do not commit blindly (unconditionally), but the commitments may be dropped (decommitted) under a payment of a compensation to the party suffering from the decommitment. The level of a commitment (its seriousness) is explicitly defined between each two agents. Thus, on one hand, the *self-interested* agents are not prevented from behaving naturally (i.e. to be opportunistic), on the other hand, the compensations to be paid for a decommitment prevent the agents from a frivolous (irresponsible) behaviour [6]

The set of commitments agreed among a community of agents forms a distributed plan, where every actor is responsible for its own part [32]. Figure 1.1 shows an example of commitment-based distributed plan. Every commitment is secured by pre-agreed decommitment rules securing stability of the plan execution in dynamic, uncertain environment. Such rules can be i.e. relaxation of the commitment (postponing in time or reducing quality), delegation to another agent (while keeping the responsibility or paying penalty, etc.), or decommit (dropping of the task) [56, 55]. Causal relations between commitments captured by a commitment graph determine the influence of the decommitment rule execution to the following commitments in the plan.

Service Level Agreement

A Service Level Agreement (SLA) introduces a formalization of a business relationship (or a part of business relationship) between two parties (most often between a provider and a customer) that is a key concept for service management [49, 10]. Most often it specifies a delivery of products or services for certain price, meeting specified deadlines, quality requirements together with financial guarantees and other contract terms. It may concern continuous, discrete or one-shot service/goods deliveries. For example for Virtual Organizations it represents a description of workflows, schedules, resource allocations, participant roles, prices, sanctions, guarantees, legacy-related and other contract management and coordination issues. The SLA introduces an consistent (possibly reduced) electronic form of the contract signed by contract parties as a paper document (the reduction may concern mainly non-technical/financial parts), expressed in a machine-readable language (most often in XML that is nowadays considered as an interoperable business information exchange format).

Service Level Agreement can be seen as a multi-attribute document that may contain a relatively great number of attributes with explicit or implicit mutual dependencies. It is possible to change several of the attributes in one round of a multi-attribute negotiation, but the more attributes is changes, the less human-readable and comprehensive the negotiation may become. The scaling of a negotiation may proceed basically in two aspects: *(i)* number of attributes (complexity o the negotiated SLA) and *(ii)* the number or parties involved in the negotiation. Both these aspects may be more-or-less interlinked. As the contract may require a participation of more parties equipped with several different competencies (for competency management see e.g. [**?**]), the negotiation complexity may grow substantially. The first-type complexity can be tackled by means of variable granularity of the SLA (e.g. there is no use of negotiating for a deadline of a specific task unless the particular partner agrees on taking responsibility for the task).

The second-type complexity can be tackled by means of a pre-negotiation picking a reasonably small number of participants for a detailed negotiation on "high-granularity" terms of the SLA while the rest of participants remain wait for their chance. Both these approaches reducing the negotiation complexity are crucial when deploying agent systems as business facilitation means that include human iteration with the system or its control as it reduces the practical manageability of the negotiation for humans (e.g. a manager does not need to negotiate with all the 50 of the addressed participants who of only five are going to participate on the contract, but may reduce the group to 10 most promising partners and carry out the negotiation in a more human manageable manner). Moreover, it reduces the number of messages exchanged and thus the amount of data exchanged that may be important e.g. from a internet connection dimensioning of the company line.

1.2 Virtual Organizations

During the last years, the concept of Virtual Corporations has evolved into various cooperation models. In our work we focus on the original concept of Virtual Organizations (VO) although the results may be applied to the other cooperation models as well (e.g. Extended Enterprise). Gruber specifies the key features of VO also defined by most of another definitions as: *an extensive use of information technology for a coordination of the partners*, *sharing risk and knowledge with partners*, and *focus on core competencies* [24]. Defining a *competence* Neubert refers to *"the cognitive, conative and expressive abilities of humans to organise their activities in order to produce certain results"* [36]. Neubert suppose the competence to be a necessary prerequisite *"realizing a business process to create valuable results"* [36] for each VO member. Fischer describes a *core competence* of an enterprise as a set of skills, technologies, and know-how crucial for the added value provided by the enterprise [19].

Classic subcontracting: Production of one partner (producer) is an input for the other one (consumer).

Technology-driven subcontracting: One partner processes a task, but lacks for a competency for some of its part. Therefore, for this part of task a suitable supplier is subcontracted.

Capacity-driven subcontracting: Similar to the previous one but the partner responsible for the task lacks for a capacity. The missing capacity is outsourced.

The VOs naturally operate in a competitive environment [6]. Every partner follows its own goals and maximizes its utility. Each of individual utility functions may use different metrics and they are usually hidden to the others. Standardized protocols for contracting are often insufficient for bargaining over contracts in such environment as the related negotiation mechanisms do not account for it [53]. In the agent system each participating partner company is represented by its agent that is able to undertake predefined automated decision making support on behalf of the partner company or it enables a user to interact with the system on behalf of the company. Another possible roles played by agents in VOs are defined e.g. in [28].

Formation of a Virtual Organization (VO) is based on a negotiation between independent partners willing to cooperate [51, 30, 12]. Individual partners (mostly SMEs) are motivated to join the Virtual Organization to increase their business opportunities and to participate on larger scale jobs.

The concept of Request-based Virtual Organizations (RBVO) defined by Roberts comprises a cluster of partnering organizations that can get along without a hierarchical ordering into a monolithic organization [44]. The RBVOs are short-living entities that are formed to respond to business opportunities offered in electronic commerce. RBVOs operations are based on predefined Service Level Agreements (SLAs). The organization and functioning of RBVOs activities is ensured by a community of intelligent agents that automate procedures and operations of RBVOs. In the RBVO defined by Roberts, the agents serve as assistants for human decision makers.

VO Lifecycle

The VO lifecycle and its phases have been described many times in previous works (e.g. [19]). The basic phases, which are included or extended in most of definitions, are (see Figure 1.2):

Creation phase, which is the first phase after discovering a business opportunity. During this phase the VO is created: The VO task is defined, VO team is formed, and then the VO is initiated.

Operation phase, which contains all the value-adding processes of the VO. In some cases there is a need for an evolution (also called adaptation) of the VO during this phase, e.g. in case of initiation of new VO members.

Dissolution phase, which finalizes and evaluates the VO operation and potentially opens future cooperation. When the task of the VO is accomplished the VO operation may be evaluated.

Targeting any of these lifecycle phases various authors extend them by the other ones. The first main phase is the creation. Fischer [19] distinguishes two phases of a creation process. In the first phase the product is defined and related business process is separated to the partial processes. In the second one the team of VO members is negotiated and formed. Aiming to the latter phase (i.e. negotiation and formation, and supporting it by multi-agent technology). Fischer distinguishes four sub-phases: *identification of potential partners*, *generation of alternative mappings from partners to partial processes*, *evaluation of strategic interests and risk*, and *finalization of partners and mapping to partial processes* [19].

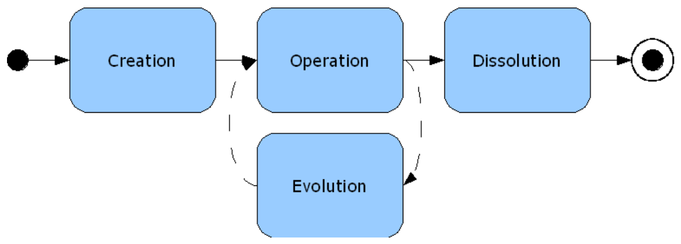


Figure 1.2: VO lifecycle – the three basic phases and optional evolution/modification phase [52].

Virtual Organizations Creation

The VO establishment is based on an agreement on a cooperation of individual partners. The concept of social commitments was introduced by Wooldridge and Jennings in [61]. This concept may be applicable in some VO domains, but it does not address the problem of unilaterally advantageous dropping of commitments. In most of VO domains an explicit employment of rewards and penalties is needed as a clear qualification of utilities that the party gains or looses. A concept of such explicit utility evaluation is then a part of commitments; the party providing a service commits not only to perform appropriate actions (in order to gain the promised utility which introduces its motivation), but also to provide a compensation in case of failing (e.g. a compensation of the profit lost to the other party). The most complete approach to the commitments in the competitive environment has been presented by Sandholm and Lesser [46] as *leveled commitment contracts* (LCC) which include an explicit utility evaluation in a form of a contract price and penalties. In order to provide a complete decision making mechanism, the authors applied several significant restrictions (e.g. the utility function needs to be identical for all participants, opportunity-cost business probability function for every agent is a common knowledge, etc.). These assumptions are limiting [6] and basically prevent a direct deployment of LCC in a real application. Nevertheless, LCC introduces a basis for notion of commitments in competitive environments.

1.3 Interaction Protocols

The scheme of interactions between agents is captured by so called interaction protocols. The protocols focused on contracting enable the agents agreed on the shared goal, pair-wise commitments or contracts. The substantial assumptions enabling such negotiation are *(i)* renegotiability of the involved commitments and *(ii)* existence of suitable negotiation means for such renegotiation. As the former assumption concerns properties of the commitments as well as attitudes of the committed agents, the latter concerns a *negotiation mechanism* comprising of both the *negotiation strategy* and *interaction protocol* [34]. FIPA [18] has standardized the most commonly used agent interaction protocols as *Request*, *Query* and *Subscribe Interaction Protocols*, several auction and brokering protocols as well as the *Contract Net Protocol* (*CNP*) proposed by Smith [48] (FIPA CNP definition is shown on Figure 1.3). by Most of the protocols mainly deal with the commitment conclusion negotiations, however, an execution of commitments or their evolution as well as a termination of the cooperative problem solving are mostly underestimated or completely disregarded. None of the FIPA interaction protocols includes proper means for an eventual renegotiation or dropping of formerly concluded commitments (i.e. means of replanning and reconfiguration). FIPA *CNP* allows for an eventual unilateral decommitment by employing a *Cancel Interaction Protocol*. However, this approach principally violates the *CNP* which cannot

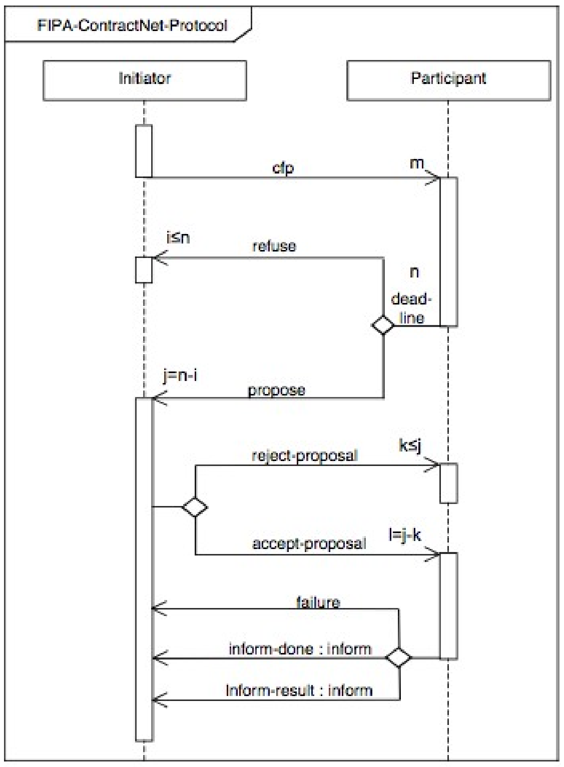


Figure 1.3: FIPA Contract Net Interaction Protocol Specification [21].

be finished correctly ever since the *Cancel Interaction Protocol* used (the mutual commitments are unilatelarly breached). It appears that FIPA interaction protocols were designed to support *full commitments* which can be reasonably used in *collaborative* environments only. Moreover, coverage of a contract life cycle as whole appears to be incomplete or unsatisfactory. There have been proposed negotiation protocols for a flexible contracting in *competitive* environments by Sandholm and Lesser [45] or Bergenti *et Al.* [5], etc. which allow for *leveled commitments.*

Contract Net Protocol (CNP)

Contract Net Protocol is one of the most popular negotiation protocol ever used in MAS's. It comes from economics and it is used in communities of altruistic as well as self-interested agents. CNP was described by Smith, who described single-shot protocol for requesting and selection of provider of product or service in group of one coordinator (who requests) and one or more participants (who may provide) [48]. In the beginning of the session the coordinator requests participants for offers and the interested ones reply their offers. Coordinator evaluates received offers and chooses the most suitable participant(s) or dissolves the session. Finally, if one or more offers are chosen, coordinator requests for them.

The CNP in its basic form provides a lot of freedom in each step of the interactions and the obligation to fulfil the contract defined in the call is not required in the basic CNP (e.g. proposals acceptance depends on the proposals themselves and the actual state of the coordinator at the moment of the proposals evaluation). The CNP fits well in collaborative environments where there are one subject evaluating possibilities and the others that are providing with for the call the most suitable offers. In the environments with some type of preferences expressions (e.g. by money) and/or environments with competitive participants the CNP must be extended by rules and features e.g. known from the auctions.

Auctions

An auction a method of optimal reallocation of resources according to the actual demand and supply, which are usually measured by monetary unit. Many types of auctions exist; they vary in features like bid adaptation possibility, number of sellers and buyers, discrete or continuous evaluating of bids, number of criteria for evaluating a bid, and others. The definitions of basic auctions are usually provided for a negotiation about a single issue with invariable features. Basically two auction mechanisms are possible: the one-shot and iterative.

In case of the former one, there is only one round of a negotiation. It means that the negotiation coordinator announces proposal to that the participants respond by obligatory offers. Then the coordinator evaluates received offers and announces winning offer(s). There are two basic types of the one shot auctions; they differ in price that the winner has to pay: *(i)* in the First-price-sealed-bid auction the winner pay price that she proposed, *(ii)* in the Second-price-sealed-bid auction the price to be paid is defined by the second best proposal. The Second-price-sealed-bid is usually called the Vickrey auction. Although it is an application of the Vickrey auction to the single-item single-unit domain it is not the only Vickrey auction. The Vickrey auction is naturally single-item multi-unit. For one kind of a commodity (single-item) it provides its redistribution of the commodity according to the match of the curves of supply and demand.

The iterative auctions are *(i)* the English auction, where the price of the auctioned issue is being increased until only one participant is paying for it (for the reverse auction the price is being decreased until only one interested provider remains), and the *(ii)* Dutch auction in that the price is too high (low in reversion auction) at the beginning of the negotiation and then it is being decreased (increased in reversion auction) until any participant accepts it. When the English and the reverse English auctions are combined together the Double auction is created. In that auction there are groups both of participants interested in selling and buying. The sellers overbid themselves by decreasing the required price, while the buyers increase it. When some selling and buying bids match the auction is successfully finished. The very special group of iterative auctions is a group of the Continuous auctions in which the bids are evaluated on-line and when some of them are matching the exchange is executed. Independently on an identified match the auction continues in identifying another match of bids.

The iterative auctions are more complicated then the single-shot auctions especially in the case of multi-criteria description of the proposals. In case of one-criteria auction, where each proposal may by described by one number, e.g. price, (actually, there is only request for comparability of each two from possible values and transitivity of the comparability), the solution is clear: the one with the highest (not dominated) offer is winner and the individual offers depend on the type of auction and preferences of the participants. The case of iterative multi-criteria auction is the most complicated one.

Legal Agreement Protocol (LAP)

Legal Agreement Protocol [42] is based on Australian contract law. The protocol allows an M:N negotiation which is split into several phases. The first phase allows a not-binding negotiation (the agreed conditions do not imply any commitment for any of the parties) which enables the parties to reach a mutually advantageous compromise. The next phase consists in a binding negotiation over a binding offer (which can be accepted or rejected). Once a contract is established, it is possible to terminate it in several ways – by fulfilling the contract (does not require communication), unilateral decommitment under agreed penalties given by the agreement, mutual agreement about cancelling the contracts without penalties, and contract breach (not solved by the protocol – to be resolved per curriam). One instance of the protocol is started for each task (a single-task negotiation) and multiple tasks are negotiated independently in concurrent protocol instances (i.e. multiple contracts). The LAP allows flexible negotiations including backtracking, withdrawing offers, temporary rejections etc. (it is possible to implement various search algorithms like depth-first search, A*, etc.). Decommitments are not negotiated upon, but are carried out unilatelarly by informing the other party about the decommitment. The protocol does not directly support contract renegotiation – it is covered by cancelling the contract or decommitting while new contract conditions are negotiated in a new contract. The protocol assumes safe message delivery and an absence of communication is involved as an interaction option of the protocol (the protocol considers timeouts explicitly). The authors of the LAP have proved various properties of the protocol like the protocol is free of a communication deadlock (communication is always terminated, though, the matter of mutual commitments and their status is not considered), etc [42].

Extended Multi-agent Negotiation Protocol

Extended Multi-agent Negotiation Protocol [1] is inspired by Extended Contract Net Protocol (ECNP) [20]. It has been proposed for use in both collaborative and competitive environments. The performatives used by ECNP either more-or-less correspond to some FIPA performatives or introduce new speech acts: *Announce*, *PreBid*, *PreAccept*, *PreReject*, *DefinitiveBid*, *DefinitiveAccept* and *DefinitiveReject*. The protocol allows a multi-round barganing with more-or-less commiting negotiation actions that optimize resource allocations (overbooking) and prevent the agents from being forced to sequence their negotiation messages when involved in more parallel bargaining (more ECNP negotiations running in parallel are enabled due to preliminary accept/reject actions allowing more flexibility). The content of messages is not restricted in any way, though, it seems to be assumed task-plan/schedule/dependance oriented rather than whole-contract oriented. Decommitments are intended to be completely prevented by a proper contracting during the contract conclusion phase (the execution phase is rather disregarded). It appears that the ECNP accounts for full commitments rather than for levelled commitments [46] – any means for reflecting changes in attitudes of the participants once committed are not provided. Thus, the rest of the contract lifecycle (execution and termination) remains rather inflexible.

Competitive Contract Net Protocol

Competitive Contract Net Protocol (C-CNP) [58] is a FIPA-like protocol designed for flexible contracting in a competitive environment (e.g. E-commerce and VOs) and aims at covering the whole contract lifecycle, specifically: *(i)* contract conclusion phase, *(ii)* optional decommitment phase, and *(iii)* contract termination phase. Not all the parties involved in a multi-round negotiation of commitments need to be addressed by call-for-proposals (CFP) messages. The protocol allows participants to impose their proposals (based on third-party information) into an already running negotiation. The 1:N negotiation is held in a pairwise manner. During the execution phase any of the parties involved in pairwise commitments may attempt to decommit from the contract. The multi-round decommitment negotiation on conditions of dissolving the cooperation may end up either by backing off by the decommitting party (the contract returns back to normal) or by dropping the commitments under a payment of a penalty (the penalty may be fixed during the contract-conclusion negotiation or may remain opened and be adjusted in time). Finally, in the termination phase the results are evaluated with respect to the agreed commitments. Eventually, penalties for non-compliance with commitments are negotiated. The message content is assumed to describe the contract as a whole, i.e. full and explicit descriptions of commitments (i.e. not only a mere task assignment, but also resource allocation, quality of service, schedules, etc.), rewards and sanctions are provided (such message content may be e.g. an SLA). Thus, the negotiation is also assumed to be multi-attribute rather than single-attribute. The multi-round manner of the protocol allows multiple simultaneously running negotiations and as well as multi-level ordering of subsequent protocols (i.e. a participant of a C-CNP may become a coordinator of another subsequent C-CNP negotiation, e.g. for outsourcing).

Renegotiable Competitive Contract Net Protocol

Renegotiable Competitive Contract Net Protocol (RC-CNP) [7] extends C-CNP by renegotiation phases and provides means for fully flexible contracting in competitive environments and enables a consistent evolution of commitments starting at their creation and terminating by their fulfilment, adaptation or breach (even partial breach) under punishment (payment of a penalty) – i.e. it covers a complete commitment lifecycle within a group of mutually committed agents (commitments are pairwise between coordinator and participants). The protocol allows M:N multi-attribute negotiations, it can be extended by not-binding phases and it leverages a possible temporary communication inaccessibility by definition of timeouts and related default transitions (as well as a possible synchronization backtracking).

RBVO Formation Protocol

RBVO Formation Protocol [52] has been designed to support a flexible formation of Request-based Virtual Organizations (RBVO) with an emphasis on reflecting the conditions of real competitive environments. It supports automated or semi-automated negotiations mainly in the creation part of a Virtual Organization lifecycle and it accounts for a use of Service Level Agreements (SLA). The protocol consists of three phases: *(i)* potential partner search, *(ii)* negotiation of SLAs and RBVO establishment, and *(iii)* RBVO (execution and) dissolution.

1.4 Information Sharing

A cooperation of agents in competitive environments is more complicated than in collaborative ones. Both the replanning and reconfiguration play the crucial role in the cooperation and introduce a means for an implementation of a system flexibility. The concepts of commitments, decommitments with the penalties and subcontractions may facilitate effective reconfiguration and replanning. Complex task decomposition problems in large agent communities operating in highly distributed heterogeneous environment is a problem that requires a big amount of communication traffic and complex optimization processes.

The information support needed for contracting in multi-agent system comprises of *(i)* potential partners search and *(ii)* information sharing between potential partners while keeping maximal possible privacy of their internal data, intentions and goals.

A common solution used for information management in teams consists in a central point maintaining the shared documents and enabling an access for all team members. Friese compares features of centralized (client-server architecture) and distributed technologies for information management. Without a central server, bottlenecks and single points of failure are avoided and individual peers keep their independence. On the other hand, network administration lacks any central control, which is required in some domains to ensure consistency and verity of data within the network [22]. Jun Yan *et Al.* also discuss weaknesses (mainly architectural limitations) of conventional workflow management systems: poor performance, lack of reliability, limited scalability, user restriction, and unsatisfactory system openness [62]. Examples of distributed architectures for information sharing during the VO creation process and further workflow management tasks are: peer-to-peer networks, grids, and multi-agent systems. Such information support in multi-agent systems can be secured by the use of acquaintance models where the agents stores the information about possible collaborators [39]. The acquaintance models helps significantly reduce the number of exchanged messages and danger of private knowledge disclosure.

Profiles and Competencies

Sharing information about agent' profiles, products, services, and available capacities within the group of collaborating independent agents helps to establish a cooperation covering the opportunities (complex tasks or a business opportunity for Virtual Organization) that cannot be carried out by separate agents.

Before VO formation the enterprises need information about potential partners to choose the right ones for the negotiation about the future collaboration. Support for the VO formation is added value of collaborative network, which consequently allows its members to concentrate to their profitable businesses. Specialized tool can simplify sharing information about agents' profiles and competencies by rules and tools for information maintenance and processing.

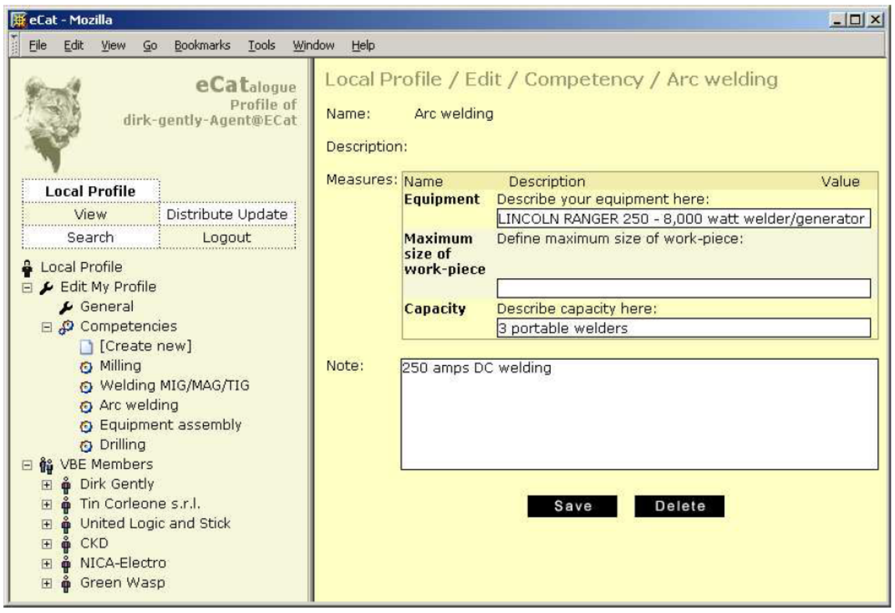
Usually only part of internal information an enterprise desires to presented to the partners. This task is solved in multi-agent systems by the Social Knowledge (described e.g. in [43]). Information that the agent never says to others (e.g. real costs) is his *private knowledge*; information given to the selected agents (e.g. during market negotiation) is *semiprivate knowledge*; and information accessible to everyone (e.g. agent's communication address) is *public knowledge.* To get information about others, one can ask them directly (or use subscribe/advertise protocol), or ask an information provider.

Generally, the problem of information sharing in distributed systems is addressed in research of distributed databases [47] and P2P networks [23], partially centralized approaches uses specialized agents [35] like mediator, broker, matchmaker, or facilitator.

The tools for sharing and managing information in collaborative networks work with *partner's profile.* This profile contains basic information about partner (such as name, address and size) and competencies. The works [50] and [8] are focused to human resources management but they are also easily applicable to the other domains. Information shared in the aim of the future collaboration is mainly based on the profiles including competencies of individual partners. Part of such information is public knowledge and part of it is provided only to selected partners. Such information is semiprivate knowledge as defined above.

An example of such tool is e-Cat system [28, 25], which is focused on the storing, maintaining and sharing business information of partners' profiles for production oriented Virtual Organizations creation support. The designed e-catalogue combines peer-to-peer approach together with centralized architecture. Used architecture is open and easily extensible. It is possible to add any component by desired functionality (e.g. web-services, proprietary GUIs, databases, etc.) or 'plug-in' any new modules (general ontology support, trust mechanisms, advanced search algorithms, etc.). The system can be also extended by modules for knowledge sharing in the domain of VO formation, negotiation and management. An example of e-Cat screenshot is given in Figure 1.4.

The catalogue hybrid solution enables effective cooperation in naturally distributed environment based on the agent negotiation mechanisms as well as standard centralized web-based



Local Profile / Edit / Competency / Arc welding

Name: Arc welding

Description:

Measures:

Name	Description	Value
Equipment	Describe your equipment here: LINCOLN RANGER 250 - 8,000 watt welder/generator	
Maximum size of work-piece	Define maximum size of work-piece:	
Capacity	Describe capacity here: 3 portable welders	

Note: 250 amps DC welding

Figure 1.4: An example of e-Cat web interface (profile editing).

approach. Distributed elements ensure maximal independence between individual partners and facilitate storing sensitive information on their local servers. Local copy of data allows each member to use the system, even if it is totally disconnected from the rest of the word. On the other hand, the “master copy” of data, that are not safety critical, are managed by each member so they are fully controlled by them. Centralized elements ensure common understanding of competencies in the whole system, and by maintaining identifying information about members they ensure access to the community only to authorized members.

Contracting Model

As shown in Section 1.1 the contracting in multi-agent system is based on appropriate interaction protocol and the means for negotiation, i.e. the contract content to be agreed. Once an agent identifies potential collaborators it utilizes agreed protocol to establish commitments.

The general contracting problem contains two pars: *(i)* task decomposition and *(ii)* service allocation. These parts can be separated in theory, but practically they are strongly interrelated. Service allocation strongly depends on the results from the task decomposition algorithm, (decomposition determines what services may be contracted), but the quality of decomposition depends on a possible outcome of service allocation (how the services would be eventually contracted). Besides, we consider the task decomposition to allow division by type and also by an amount.

The number of possible decompositions is exponentially growing with increasing needed amount and the number of possible delivery agent combination exponentially grows with increasing number of involved agents. Enumeration of all the decompositions consumes big amount of time and evaluation of such number of requires huge communication.

To avoid this problem, so called subscribe-advertise protocol can be used for update agent’s knowledge before the course of service allocation is started. This method significantly reduces communication flow of the allocation phase [40]. Generally, overall number of messages is not lower then with updating the model during allocation phase (in worst case, it has the same complexity). But it can be much lower in the case of stable, slowly changing environment. On the other hand, mass subscription among agents can produce huge communication during knowledge-updating phase.

Building complete model of each possible service provider is not efficient. Actually, only profitable part of the service providing distribution model is important. Unfortunately, agent can not know which part of model is profitable before complete model is built. The providing agents' behaviors models (delivery time distribution, price policy, etc.) can be used by the service requesting agent for searching the best estimation of the most suitable decomposition and allocation. Collecting the most accurate knowledge about other agents helps to make proper decisions without need of additional negotiation.

An example of acquaintance model minimizing the amount of information exchange is Incrementally Defined Acquaintance Model (IRAM) [14]. From the single agent point of view (requestor) the problem is to find the most optimal decomposition of the task service allocation within the community members. Provided that the model starts with the simplest possible decomposition (1 task decomposes to exactly 1 service) and the problem is reduced to service single allocation problem that has been previously solved by Contract Net Protocol [48] or various auctioning techniques [29, 11]. From this starting point the agent iteratively builds the partially-linear acquaintance model that is beneficial mainly in stable or slowly changing agent community, while it can be also usefully employed in dynamically changing communities.

Using models in the contracting (optimization) phase, the non-profitable partners can be eliminated and they may not be contracted. It speeds up the contracting phase and reduces private knowledge disclosure. In idealistic case, only the best partners should be contracted.

Using IRAM, the decomposition optimization and allocation phase is highly effective in the both, competitive and cooperative environment. The number of exchanged messages is much lower than without using these techniques and allows operate in the domains with poor connection and restricted communication. This approach has been used for optimization of the delivery time required to bring the aid to the needed [43]. An alternative usage of the acquaintance model based negotiation is in the supply chain management where the task corresponds to a specific manufacturing project in a single company [39]. The task is decomposed into the component list and the components that need to be purchased externally correspond to the services. In supply chain management the tasks are usually aggregated in clusters of projects. This is why purchase of components is often organized in high volumes. It is not a rare situation when there is no service provider that can provide the requested services in the amount desired. Similarly, there is a potential for deploying the acquaintance based task decomposition in the project oriented domains [52].

1.5 Case Studies

In the past 10 years the results of the research outlined in this chapter has been utilized in applications and demonstration prototypes developed in series of the research projects and industrial cooperations.

An early prototype of the multi-agent contracting protocol has been adopted by ExtraPlanT [26] multi-agent system for production planning. This system operates on two levels: *(i)* intra-enterprise level represented by a set of *planning* and *resource agents* and *(ii)* extra-enterprise level represented by *enterprise-to-enterprise* (E2E) *agents* which allow free capacity sharing among independent enterprises. The standard Contract Net Protocol has been used for intra-enterprise planning where all the agents maximize overall optimality criterion. On this

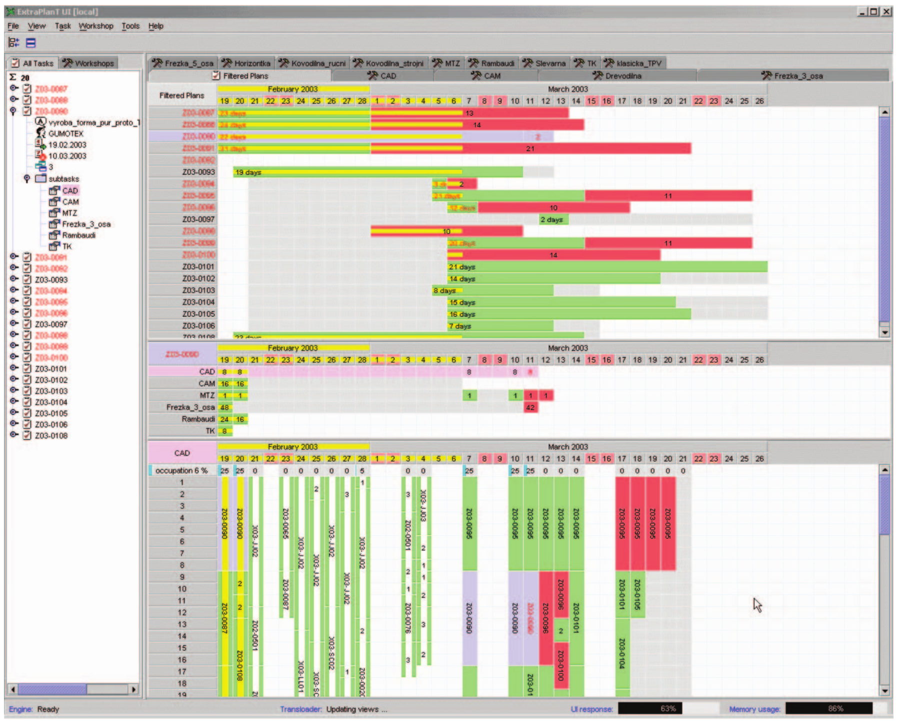
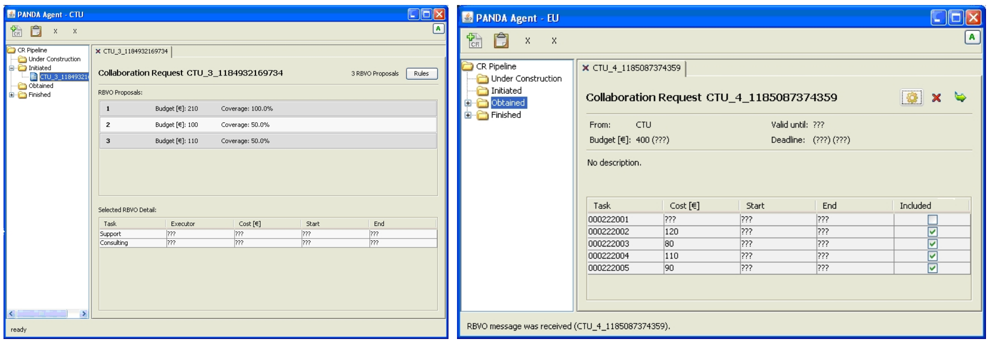


Figure 1.5: An example of ExPlanTech planning system UI interface [41].

level the contracts are not breached, but may be modified by means of a reconfiguration – no penalty is charged and no negotiation about decommitment is needed (example of the planning result is captured on Figure 1.5). On the contrary, on the extra-enterprise level both the optional decommitment and contract termination negotiations are to be taken into account. Each E2E agent represents one independent enterprise with it's own goals and other business opportunities. Here C-CNP improves collaboration and cooperation possibilities of the system.

Extra-enterprise part of the ExtraPlanT has been adapted to support a cooperation in Enterprise Resource Systems (ERP) value chains as a part of FP6 specific targeted research on innovation project PANDA. A distributed intelligent agent system employs E2E agents for *(i)* a potential collaborators search using e-Cat, *(ii)* a contract negotiation among several partners using IRAM and RBVO formation protocol, *(iii)* a cooperation monitoring, and *(iv)* replanning and reconfiguration. The E2E agents represent individual ERP vendors or dealers and support a full human control of all the information provided by agents to the system. Contract details and potential penalties are described by *service level agreement*. The successfully deployed prototype has been tested in the real industrial environment. There were about 38 Partner Agents utilizing the protocol deployed on various sites (hosted on platform central server or distributed on user partners servers or PCs in 7 different geographical locations); 12 of them were able to play coordinator role and thus initiate the RBVO formation. The Partner Agents were able to provide 5 different services with various constrains like languages, countries, industry domains, ERP module expertise, reputation, price and availability etc. In worst case, there were maximal of 24.3 millions of potential RBVOs for non constrained 5-tasks CR if all companies offer all services. The RBVO Formation Protocol empowered by IRAM algorithm and business logic captured by rules provides efficient (semi-)automated cooperation establishment and effective collaboration



Collaboration Request CTU_3_1184932169734

RBVO Proposals:

1	Budget [€]: 210	Coverage: 100.0%
2	Budget [€]: 100	Coverage: 50.0%
3	Budget [€]: 110	Coverage: 50.0%

Selected RBVO Detail:

Task	Executor	Cost [€]	Start	End
Support	???	???	???	???
Consulting	???	???	???	???

Collaboration Request CTU_4_1185087374359

From: CTU
Valid until: ???
Budget [€]: 400 (???)
Deadline: (???) (???)

No description.

Task	Cost [€]	Start	End	Included
000222001	???	???	???	☐
000222002	120	???	???	✓
000222003	80	???	???	✓
000222004	110	???	???	✓
000222005	90	???	???	✓

RBVO message was received (CTU_4_1185087374359).

Figure 1.6: An example of PANDA UI interface – RBVO proposals evaluation by coordinator (left) and call response editing by partner (right).

support. Examples of user interfaces for RBVO proposals evaluation and participation proposal editing in Partner Agent are depicted in Figure 1.6.

The complete application set of usage of the family of contracting protocols based on C-CNP comprises of *(i)* a cooperation support in ERP value chains (a distributed intelligent agent system which is a part of FP6 specific targeted research on innovation project PANDA), *(ii)* a formation of virtual organizations (a distributed decision making support system which is as a part of FP6 integrated project ECOLEAD), and *(iii)* a distributed system for modelling and validation of e-Business contracts (a part of FP6 specific targeted research project CONTRACT).

Although cooperating in a cluster, its members are self-interested in all the presented application scenarios. Thus, they may have an intention of leaving an already concluded virtual organization due to more profitable businesses. An another reason for a revision of an already concluded contract is an incapability of a Virtual Organization to respond to new circumstances that had not been known during the contract conclusion. In order to respect such features of a cooperation in virtual organizations a possibility of both the contract adaptation and dissolution must be taken into account.

The example of application from cooperative contracting environment is i-Globe system [33] for planning in humanitarian relief scenarios using social commitments. Figure 1.7 shows an example of system user interface and commitments execution visualization. The research results applied there has been focussed on the process of setting the contract conditions. Both the contract prices and penalties affect substantially the flexibility of the cooperation in both cooperative and competitive environments and their proper setting is a crucial issue in contracting [6]. In this case, the stability and reconfigurability of the commitments in unpredictable execution environment has been studied [56, 55].

Selected publications about topics introduced in this chapter are summarized in next chapter and included in Appendices.

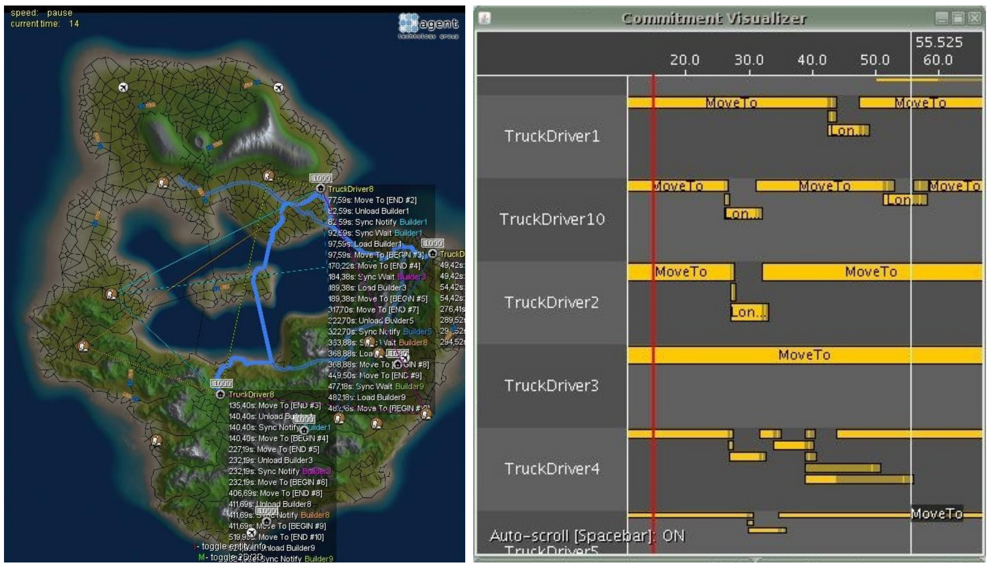


Figure 1.7: Screenshot of i-Globe simulation visualization – situation overview with selected plan of one truck agent and corresponding commitments shown on the map (left) and time-table of commitment execution (right).

Bibliography

[1] S. Aknine, S. Pinson, and M. Shakun. An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, 8(1):5–45, 2004.

[2] L. G. Alan H. Bond, editor. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA, 1988.

[3] M. Andersson and T. Sandholm. Leveled commitment contracts with myopic and strategic agents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98, 26-30 July 1998, Madison, WI, USA*, pages 38–45, Menlo Park, CA, USA, 1998. AAAI Press/MIT Press.

[4] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare (Handbooks in Economics).* North Holland, August 2002.

[5] F. Bergenti, A. Poggi, and M. Somacher. A contract decommitment protocol for automated negotiation in time variant environments. In A. Omicini and M. Viroli, editors, *WOA 2001 - Dagli oggetti agli agenti: tendenze evolutive dei sistemi software*, pages 56–61. Pitagora Editrice Bologna, 2001.

[6] J. Bíba and J. Vokřínek. Agent contracting and reconfiguration in competitive environments. In *Cybernetics and Systems 2006*, volume 2, pages 527–532. Austrian Society for Cybernetics Studies, 2006.

[7] J. Bíba, J. Vokřínek, and J. Hodík. Renegotiable competitive contract net protocol. Technical report, Agent Technology Group, Gerstner Laboratory, Czech Technical University in Prague, 2008.

[8] E. Biesalski. Knowledge management and e-human resource management. In *FGWM 2003*, Karlsruhe, 2003.

[9] I. Boughzala and M. Zacklad. Cooperation engineering for the extended enterprise. In P. Lenca, editor, *Proceedings of the Human Centered Processes Conference*, pages 119–127, Brest, France, September 1999.

[10] J. Bouman, J. Trienekens, and M. van der Zwan. Specification of service level agreements, clarifying concepts on the basis of practical research. *Software Technology and Engineering Practice, 1999. STEP '99. Proceedings*, pages 169–178, 1999.

[11] C. Boutilier, M. Goldszmidt, and B. Sabata. Sequential auctions for the allocation of resources with complementarities. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 527–523. Morgan Kaufmann Publishers Inc., 1999.

[12] J. Browne and J. Zhang. Extended and virtual enterprises - similarities and differences. *International Journal of Agile Management Systems*, 1(1):30–36, 1999.

[13] W. Davidow and M. S. Malone. *The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century.* New York: Harper Business, 1992.

[14] J. Doubek, J. Vokřínek, M. Pěchouček, and M. Rehák. Incrementally refined acquaintance model for consortia composition. In M. Klusch, Pěchouček, and A. Polleres, editors, *Cooperative Information Agents XII, 12th International Workshop, CIA 2008*, volume 5180 of *Lecture Notes in Computer Science*, pages 280–291. Springer, 2008.

[15] B. Dunin-Keplicz and R. Verbrugge. Evolution of collective commitment during teamwork. *Fundamenta Informaticae*, 56(4):563–592, August 2003.

[16] E. H. Durfee. Distributed problem solving and planning. In G. Weiß, editor, *A Modern Approach to Distributed Artificial Intelligence*, chapter 3. The MIT Press, San Francisco, CA, 1999.

[17] F. Farhoodi and I. Graham. Practical approach to designing and building intelligent software agents. In *Proceedings of PAAM'96*, pages 181–204. The Practical Application Company, 1996.

[18] FIPA. Foundation for intelligent physical agents [online]. `http://www.fipa.org`.

[19] K. Fischer, I. Heimig, and J. Müller. Intelligent agents in virtual enterprises. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology (PAAM 1996)*, 1996.

[20] K. Fischer, J. P. Muller, and M. Pischel. Schedulling an application domain for dai. *Applied Artificial Intelligence*, 10:1–33, 1996.

[21] Foundation for Intelligent Physical Agents. FIPA Contract Net Interaction Protocol Specification. `http://www.fipa.org/specs/fipa00029/`, 2001. [Online; accessed 24-January-2013].

[22] T. Friese, B. Freisleben, S. Rusitschka, and A. Southall. A framework for resource management in peer-to-peer networks. In *Proceedings of NetObjectDays 2002*, pages 4–21, 2002.

[23] T. Friese, J. Müller, M. Smith, and B. Freisleben. A robust business resource management framework based on a peer-to-peer infrastructure. In *Proceedings of the 7th International IEEE Conference on E-Commerce Technology, Munich, Germany*, pages 215–222, 2005.

[24] M. Gruber and M. Nöster. Investigating structural settings of virtual organizations. In K. Pawar, F. Weber, K. Thoben, and B.Katzy, editors, *Proc. of the Eleventh Int. Conf. on Information and Computation Economies ICE-2005*, pages 245–252. Centre for Concurrent Enterprising, 2005.

[25] Hodík, Vokřínek, and P. Becvar. Support for virtual organisation creation - partners' profiles and competency management. *International Journal of Agent-Oriented Software Engineering*, 3(2/3):230–251, 2009. INDERSCIENCE ENTERPRISES LTD, World Trade Center BLDG.

[26] J. Hodík, P. Bečvář, M. Pěchouček, J. Vokřínek, and J. Pospíšil. Explantech and extraplant: multi-agent technology for production planning, simulation and extra-enterprise collaboration. *International Journal of Computer Systems Science and Engineering*, 20(5):357–367, 2005.

[27] J. Hodík, P. Bečvář, J. Vokřínek, J. Bíba, and E. Semsch. e-Cat – VBE members profiling and competency management tool. In *Proceedings of the IPROMS 2006, the Second Virtual International Conference on Intelligent Production Machines and Systems*, 2006.

[28] J. Hodík, J. Vokřínek, J. Bíba, and P. Bečvář. Competencies and profiles management for virtual organizations creation. In *LNAI 4696 – CEEMAS 2007: Multi-Agent Systems and Applications V*, pages 656–668, Berlin, 2007. Springer-Verlag.

[29] L. Hunsberger and B. J. Grosz. A combinatorial auction for collaborative planning. In *ICMAS '00: Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, pages 151–158. IEEE Computer Society, 2000.

[30] H. Jägers, W. Jansen, and W. Steenbakkers. Characteristics of virtual organizations. In P. Sieber and J. Griese, editors, *Organizational Virtualness, Proceedings of the VoNet – Workshop*, pages 65–76. Simowa Verlag Bern, 1998.

[31] A. Komenda, M. Pěchouček, J. Bíba, and J. Vokřínek. Planning and re-planning in multi-actors scenarios by means of social commitments. In *Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT/ABC 2008)*, volume 3, pages 39–45. IEEE, october 2008.

[32] A. Komenda, Vokřínek, and Pěchouček. Plan representation and execution in multi-actor scenarios by means of social commitments. *Web Intelligence and Agent Systems*, 9(2):123–133, march 2011.

[33] A. Komenda, J. Vokřínek, M. Pěchouček, G. Wickler, J. Dalton, and A. Tate. I-globe: Distributed planning and coordination of mixed-initiative activities. In *KSCO '09: Knowledge Systems for Coalition Operations 2009*, Chilworth Manor, Southampton, UK, Mar-Apr 2009.

[34] A. R. Lomuscio, M. Wooldridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation*, 12(1):31–56, January 2003.

[35] A. Masaud, H. Ghenniwa, and W. Shen. Brokering services in cooperative distributed systems: Privacy-based model. In *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies - EC-Web 2003*, pages 435–444. Springer, 2003.

[36] R. Neubert, O. Langer, O. Görlitz, and W. Benn. Virtual enterprises – challenges from a database perspective. In *Proceedings of the IEEE Workshop on Information Technology for Virtual Enterprises*, pages 98–106, 2001.

[37] O'Sullivan and S. M. Sheffrin. *Economics: Principles in Action.* Addison Wesley Longman, 2002.

[38] H. p. Wiendahl and S. Lutz. Production in networks. *Annals of the CIRP*, 51:1–14, 2002.

[39] 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.

[40] M. Pěchouček, V. Mařík, and O. Štěpánková. Towards reducing communication traffic in multi-agent systems. *Journal of Applied Systems*, 2(1):152–174, 2001. ISSN 1466-7738.

[41] M. Pěchouček, J. Vokřínek, and P. Bečvář. Explantech: Multiagent support for manufacturing decision making. *IEEE Intelligent Systems*, 20(1):67–74, 2005.

[42] D. Perugini, S. Jarvis, S. Reschke, and D. Gossink. Distributed deliberative planning with partial observability: Heuristic approaches. In *Proceedings of the Fourth International Conference on Knowledge Systems for Coalition Operations (KSCO-2007). In Proceedings of the IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems Modeling, Evolution and Engineering (KIMAS-2007)*, Waltham, MA, USA, May 2007.

[43] M. Pěchouček, V. Mařík, and J. Bárta. A knowledge-based approach to coalition formation. *IEEE Intelligent Systems*, 17(3):17–25, 2002.

[44] R. Roberts, A. Svirskas, and B. Matthews. Request based virtual organisations (RBVO): An implementation scenario. In *Collaborative Networks and their Breeding Environments*, volume 186 of *IFIP*, pages 17–25. Springer, 2005.

[45] T. Sandholm and V. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of First International Conference on Multiagent Systems , 12-14 June 1995 , San Francisco, CA, USA*, pages 328–35, Menlo Park, CA, USA, 1995. AAAI Press.

[46] T. Sandholm and V. Lesser. Leveled commitment contracts and strategic breach. *Games and Economic Behavior*, 35(1-2):212–70, April-May 2001.

[47] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236, 1990.

[48] R. G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. *In IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[49] J. J. M. Trienekens, J. J. Bouman, and M. van der Zwan. Specification of service level agreements: Problems, principles and practices. *Software Quality Journal*, 12(1):43–57, March 2004.

[50] HR-XML Consortium. HR-XML homepage [online]. ⟨`http://ns.hr-xml.org`⟩, 8 2004.

[51] J. van Wijk, D. Geurts, and R. Bultje. 7 steps to virtuality: understanding the virtual organisation processes before designing ict support. Objects, Components and the Virtual Enterprise '98. An interdisciplinary workshop, Proceedings, http://www.cs.tcd.ie/ Virtues/ ocve98/ proceedings/ index.html, 1998.

[52] J. Vokřínek, J. Bíba, J. Hodík, and J. Vybíhal. The rbvo formation protocol. *International Journal of Agent-Oriented Software Engineering*, 3(2/3):135–162, 2009. INDERSCIENCE ENTERPRISES LTD, World Trade Center BLDG.

[53] J. Vokřínek, J. Bíba, J. Hodík, J. Vybíhal, and P. Volf. Rbvo formation protocol. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, pages 454–457, 2007.

[54] J. Vokřínek, J. Hodík, M. Pěchouček, P. Bečvář, and J. Pospíšil. *ExtraPlanT as a Multi-Agent System for Extra-Enterprise Collaboration*, pages 593–600. Information science Reference, 2008.

[55] J. Vokřínek, A. Komenda, and M. Pěchouček. Decommitting in multi-agent execution in non-deterministic environment: experimental approach. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 2*, pages 977–984. IFAAMAS, 2009.

[56] J. Vokřínek, A. Komenda, and M. Pěchouček. Relaxation of social commitments in multi-agent dynamic environment. In *ICAART 2009 - Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 520–525. INSTICC Press, 19-21 January 2009.

[57] J. Vokřínek, A. Komenda, and M. Pěchouček. Abstract architecture for task-oriented multi-agent problem solving. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(1):31 –40, 2011.

[58] J. Vokřínek, J. Hodík, J. Bíba, J. Vybíhal, and M. Pěchouček. Competitive contract net protocol. In *Lecture Notes in Computer Science 4362 - SOFSEM 2007: Theory and Practice of Computer Science*, pages 656–668, Berlin, 2007. Springer-Verlag.

[59] A. Říha, M. Pěchouček, J. Vokřínek, and V. Mařík. From intra-enterprise towards extra-enterprise production planning. In *In Knowledge and Technology Integration in Production and Services*, pages 349–356, New York, 2002. Kluwer Academic / Plenum Publishers.

[60] D. Walters. Virtual organisations: new lamps for old. *Management Decision*, 38(6):420–436, 2000.

[61] M. Wooldridge and N. R. Jennings. The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4):563–592, August 1999.

[62] J. Yan, Y. Yang, , and G. K. Raikundalia. SwinDeW-a p2p-based decentralized work-flow management system. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 36(5):922–935, 2006.

Chapter 2

Selected Published Works

The topics introduced in previous chapter are based on more that ten years of research performed at Agent Technology Center[1] established in 1999 at the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, moved to the Department of Computer Science and Engineering at 2011. It consists of the achievements and results from series of projects focussed on multi-agent planning, contracting and cooperation (for more details see Section 1.5).

This chapter summarises selected published works and provides references for additional reading. It is divided into three section according selected topics: (i) planning and contracting in multi-agent systems, (ii) interaction protocols and (iii) social commitments. In each section the most important publications are introduced (the ones included in Appendixes) together with additional references.

All impact factors (IF) of journals are taken from ISI Web of Knowledge Journal Citation Reports as 5-Year Impact Factor by January 5, 2013. Source Normalized Impact per Paper (SNIP) for journal papers is taken from SCOPUS database by January 5, 2013 for year 2011. All publications are also supplied by the authorship ratio of the author of this thesis.

2.1 Planning and Contracting

The planning and contracting in multi-agent systems is an important issue from both theoretical and application oriented point of views. The main theoretical achievement is represented in following article (see article in Appendices page 29 or [20]).

> J. Vokřínek, A. Komenda, and M. Pěchouček. Abstract architecture for task-oriented multi-agent problem solving. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(1):31 –40, 2011. (IF 2.397, SNIP 3.835, authorship 80%).

The article focusses to problem solving and planning in decentralized environments. It presents an abstract architecture of a multi-agent solver and respective algorithm providing decomposition, task allocation, and task delegation. The article studies various features of the abstract architecture, such as computational complexity or admissibility of the underlying optimization heuristics. Additionally, the article presents an short overview of four instances of the abstract

[1] http://agents.fel.cvut.cz

architecture implementations demonstrating the applicability of the presented abstract solver. These implementations are described in more details in others articles and papers below. The contributions of the author of this thesis are a definition of the multi-agent problem, proposed solving architecture and algorithm, formal analyses of features, and multi-agent problem solver implementation.

Applications of the multi-agent contracting in the domain of production planning are illustrated by following article (see article in Appendices page 39 or [13]).

M. Pěchouček, J. Vokřínek, and P. Bečvář. Explantech: Multiagent support for manufacturing decision making. *IEEE Intelligent Systems*, 20(1):67–74, 2005. (IF 2.316, SNIP 2.693, authorship 33%)

The article presents ExPlanTech, a consolidated technological multi-agent framework for decision-making support in manufacturing production planning. It provides a proven alternative to known mathematical and system science-modeling technologies for simulating the manufacturing process. The article discusses main features of this framework, it architecture and implementation. It also explains underlying agent coordination and negotiation methods and possible use cases. The main contributions of the author of this thesis to this article are design of the contracting mechanism, its implementation for production planning, and cooperation on the design of an overall software architecture of the system.

The applicability study of the multi-agent contracting schema to solving vehicle routing problems has been published on prestigious multi-agent conference[2] (see paper in Appendices page 47 or [22]).

J. Vokřínek, A. Komenda, and M. Pěchouček. Agents towards vehicle routing problems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 773–780, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. (authorship 80%)

The paper presents a multi-agent vehicle routing problem solver. The solver is based on multi-agent solver abstract architecture discussed above. It utilizes the contract-net protocol based allocation and several improvement strategies. The self-organizing capability of the system successfully minimizes the number of vehicles used and the quality of the solution reaches 81% of the best known solution in polynomial time. The presented solver architecture supports great runtime parallelization with incremental increase of solution quality. In this case the main contributions of the author of this thesis are formalization of the vehicle routing problem as a multi-agent problem, adaptation of the abstract multi-agent solver architecture for this problem, implementation and experiments evaluation.

Additional information about applications of multi-agent planning and contracting for manufacturing domain can be found in [1, 4, 11, 12, 14, 17, 23]. The advances in agent-based vehicle routing problem solving can be found in [7, 8, 9].

[2]Ranked as 6th of 361 Artificial Intelligence conferences by Microsoft Academic Search sorted by conference H-index (by January 5, 2013).

2.2 Interaction Protocols

The interaction protocols are the most important negotiation scheme for multi-agent contracting and planning. The extension of Contract Net Protocol for competitive environment has been published in [21]. Extension for contracting in Virtual Organizations has been published in [16] and extended to following article (see article in Appendices page 55 or [15]).

J. Vokřínek, J. Bíba, J. Hodík, and J. Vybíhal. The RBVO formation protocol. *International Journal of Agent-Oriented Software Engineering*, 3(2/3):135–162, 2009. INDERSCIENCE ENTERPRISES LTD, World Trade Center BLDG. (SNIP 2.122, authorship 80%)

The article presents a protocol designed to support the flexible formation of Request-Based Virtual Organisations with an emphasis on reflecting the conditions of real competitive environments. It supports automated or semi-automated negotiations mainly in the creation part of a virtual organization life cycle and it accounts for the use of Service Level Agreements. The case study is provided to show the applicability of the protocol in a real industrial case. The prototype has been implemented on top of a web-service-based agent platform and validated in an open distributed environment. The main contributions of the author of this thesis to this article are design of the overall system architecture, design of the RBVO formation protocol and cooperation on the design and implementation of the client applications in the presented case study.

The contracting model minimizing information exchanged between agents that can be seen as a extension of an protocol has been published in following paper (see paper in Appendices page 83 or [2]).

J. Doubek, J. Vokřínek, M. Pěchouček, and M. Rehák. Incrementally refined acquaintance model for consortia composition. In M. Klusch, M. Pěchouček, and A. Polleres, editors, *Cooperative Information Agents XII, 12th International Workshop, CIA 2008*, volume 5180 of *Lecture Notes in Computer Science*, pages 280–291. Springer, 2008. (authorship 30%)

The paper presents a specific contracting algorithm that contributes to the process of distributed planning and resource allocation. The algorithm is based on incrementally refined acquaintance models of the actor that provide the right set of approximate knowledge needed for appropriate task decomposition and delegation with minimization of the private knowledge disclosure. The paper reports on empirical evaluation of the algorithm deployment in consortia formation in virtual organizations domain. In this paper the main contributions of the author of this thesis is are cooperation on problem definition and algorithm design, experiments design and results interpretation.

The applicability of multi-agent contracting schemes based on profiles and competencies in the domain of virtual organizations is also captured in [5, 6, 3].

2.3 Social Commitments

With given contracting scheme and protocols the distributed plan in multi-agent system can be represented as a set of social commitments as introduced in following article (see article in Appendices page 95 or [10]).

A. Komenda, J. Vokřínek, and M. Pěchouček. Plan representation and execution in multi-actor scenarios by means of social commitments. *Web Intelligence and Agent Systems*, 9(2):123–133, march 2011. (SNIP 0.886, authorship 20%)

The article presents an approach to plan representation in multi-actor scenarios. A distributed hierarchical plan is represented by social commitments, as a theoretically studied formalism representing mutual relations among intentions of collaborating agents. The article presents a formal model of a recursive form of commitments and discusses how it can be deployed to a selected hierarchical planning scenario. It also discusses decommitment rules definition and their influence on the plan execution robustness and stability. The approach is verified and evaluated in a simulated environment. The experimental validation confirms the performance, stability, and robustness of the system in complex scenarios. The main contributions of the author of this thesis in this paper are design of contracting mechanism using social commitments and its implementation for i-Globe system used for evaluation and cooperation an the formalization plan representation, mainly regarding decommitment rules.

The process of planning in complex, multi-actor environment depends strongly on the ability of the individual actors to perform intelligent decommitment upon specific changes in the environment. The study of commitments stability during plan execution in non-deterministic environment has been presented on prestigious multi-agent conference (see paper in Appendices page 106 or [18]).

J. Vokřínek, A. Komenda, and M. Pěchouček. Decommitting in multi-agent execution in non-deterministic environment: experimental approach. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 2*, pages 977–984. IFAAMAS, 2009. (authorship 35%)

The paper focusses to a reasoning about decommitment alternatives during the planning process that contributes to flexibility and robustness of the resulting plan. The formal definition of three specific decommitment rules (relaxation, delegation and full decommitment) is given in the paper. The potential of an appropriate selection, setting and preference ordering of the decommitment rules to robustness of the plans during an execution is validated in a set of empirical experiments. The main contributions of the author of this thesis are design, formalisation and implementation of decommitment rules, and cooperation on experimental validation, mainly an interpretation of results.

More details about commitments stability supported by the relaxation decommitment rule can be also found in [19].

Bibliography

[1] P. Bečvář, P. Charvát, M. P. J. Pospíšil, A. Říha, and J. Vokřínek. Explantech/extraplant: Production planning and supply-chain management multi-agent solution. *EXP - in search of innovation*, 3(3):116–125, 2003.

[2] J. Doubek, J. Vokřínek, M. Pěchouček, and M. Rehák. Incrementally refined acquaintance model for consortia composition. In M. Klusch, Pěchouček, and A. Polleres, editors, *Cooperative Information Agents XII, 12th International Workshop, CIA 2008*, volume 5180 of *Lecture Notes in Computer Science*, pages 280–291. Springer, 2008.

[3] Hodík, Vokřínek, and P. Becvar. Support for virtual organisation creation - partners' profiles and competency management. *International Journal of Agent-Oriented Software Engineering*, 3(2/3):230–251, 2009. INDERSCIENCE ENTERPRISES LTD, World Trade Center BLDG.

[4] J. Hodík, P. Bečvář, M. Pěchouček, J. Vokřínek, and J. Pospíšil. Explantech and extraplant: multi-agent technology for production planning, simulation and extra-enterprise collaboration. *International Journal of Computer Systems Science and Engineering*, 20(5):357–367, 2005.

[5] J. Hodík, P. Bečvář, J. Vokřínek, J. Bíba, and E. Semsch. e-Cat – VBE members profiling and competency management tool. In *Proceedings of the IPROMS 2006, the Second Virtual International Conference on Intelligent Production Machines and Systems*, 2006.

[6] J. Hodík, J. Vokřínek, J. Bíba, and P. Bečvář. Competencies and profiles management for virtual organizations creation. In *LNAI 4696 – CEEMAS 2007: Multi-Agent Systems and Applications V*, pages 656–668, Berlin, 2007. Springer-Verlag.

[7] P. Kalina and Vokřínek. Algorithm for vehicle routing problem with time windows based on agent negotiation. In *Proceedings of the Seventh International Workshop on Agents in Traffic and Transportation (ATT) at AAMAS 2012*, pages 1–10, Geneve, 2012. European Office of Aerospace Research and Development.

[8] P. Kalina and Vokřínek. Improved agent based algorithm for vehicle routing problem with time windows using efficient search diversification and pruning strategy. In *Proceedings of the Third International Workshop on Artifitial Intelligence and Logistics*, pages 13–18, Lyon, 2012. CNRS-ENS.

[9] P. Kalina and Vokřínek. Parallel solver for vehicle routing and pickup and delivery problems with time windows based on agent negotiation. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, SMC 2012, Seoul, Korea (South), October 14-17, 2012*, pages 1558–1563. IEEE, 2012.

[10] A. Komenda, Vokřínek, and Pěchouček. Plan representation and execution in multi-actor scenarios by means of social commitments. *Web Intelligence and Agent Systems*, 9(2):123–133, march 2011.

[11] V. Mařík, M. Pěchouček, J. Vokřínek, and A. Říha. Application of agent technologies in extended enterprise production planning. In *EurAsia-ICT 2002: Information and Communication Technology*, pages 998–1007, Berlin, Germany, 2002. Springer.

[12] M. Pěchouček, A. Říha, J. Vokřínek, V. Mařík, and V. Pražma. Explantech: Applying multi-agent systems in production planning. *International Journal of Production Research*, 40(15):3681–3692, 2002.

[13] M. Pěchouček, J. Vokřínek, and P. Bečvář. Explantech: Multiagent support for manufacturing decision making. *IEEE Intelligent Systems*, 20(1):67–74, 2005.

[14] M. Pěchouček, J. Vokřínek, and P. Bečvář. Explantech: Multiagent support for manufacturing decision making. *IEEE Intelligent Systems*, 20(9):67–74, 2005.

[15] J. Vokřínek, J. Bíba, J. Hodík, and J. Vybíhal. The rbvo formation protocol. *International Journal of Agent-Oriented Software Engineering*, 3(2/3):135–162, 2009. INDERSCIENCE ENTERPRISES LTD, World Trade Center BLDG.

[16] J. Vokřínek, J. Bíba, J. Hodík, J. Vybíhal, and P. Volf. Rbvo formation protocol. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, pages 454–457, 2007.

[17] J. Vokřínek, J. Hodík, M. Pěchouček, P. Bečvář, and J. Pospíšil. *ExtraPlanT as a Multi-Agent System for Extra-Enterprise Collaboration*, pages 593–600. Information science Reference, 2008.

[18] J. Vokřínek, A. Komenda, and M. Pěchouček. Decommitting in multi-agent execution in non-deterministic environment: experimental approach. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 2*, pages 977–984. IFAAMAS, 2009.

[19] J. Vokřínek, A. Komenda, and M. Pěchouček. Relaxation of social commitments in multi-agent dynamic environment. In *ICAART 2009 - Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 520–525. INSTICC Press, 19-21 January 2009.

[20] J. Vokřínek, A. Komenda, and M. Pěchouček. Abstract architecture for task-oriented multi-agent problem solving. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(1):31 –40, 2011.

[21] J. Vokřínek, J. Hodík, J. Bíba, J. Vybíhal, and M. Pěchouček. Competitive contract net protocol. In *Lecture Notes in Computer Science 4362 - SOFSEM 2007: Theory and Practice of Computer Science*, pages 656–668, Berlin, 2007. Springer-Verlag.

[22] J. Vokřínek, A. Komenda, and M. Pěchouček. Agents towards vehicle routing problems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 773–780, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[23] A. Říha, M. Pěchouček, J. Vokřínek, and V. Mařík. From intra-enterprise towards extra-enterprise production planning. In *Knowledge and Technology Integration in Production and Services*, pages 349–356, New York, 2002. Kluwer Academic / Plenum Publishers.

Appendices

The papers and articles summarised in Chapter 2 are included in this section of the text. To summarise, the included appendices consist of five journal articles (three of them impacted) and three conference papers (two of them published on most prestigious multi-agent conference). The list of included publications with page numbers follows.

Abstract Architecture for Task-oriented Multi-agent Problem Solving

Jiří Vokřínek, Antonín Komenda, and Michal Pěchouček

***Abstract*—Problem solving and planning in decentralized environments is a key technical challenge in numerous industrial applications, ranging from manufacturing, logistics, virtual enterprizes to multirobotics systems. We present an abstract architecture of a multiagent solver and respective algorithm providing decomposition, task allocation, and task delegation. Various features of the abstract architecture, such as computational complexity or admissibility of the underlying optimization heuristics, are analyzed in the paper. Four instances of the abstract architecture implementations are given to demonstrate the applicability of the abstract solver in a wide variety of real-problem domains.**

***Index Terms*—Algorithms, complexity, distributed planning and problem solving, multiagent applications, multiagent systems, task allocation, task delegation.**

I. Introduction

THE PROBLEM of distributed decision making, decentralized planning, and controlling entities in heterogeneous distributed environments is crucial for many application domains [1]. Existing centralized methods depend on one central-planning system that gathers all required data about decentralized entities before the planning process starts. This approach faces various difficulties. One problem is the need for sharing private knowledge and information about the actual status of these entities. The other problem is the need for real-time replanning based on environments and conditions changing dynamically in time. We present an approach, where each entity is in charge of suggesting their individual plans, where cooperation, resource sharing, and deconflition is solved by methods of negotiation.

The problem of distributed planning and problem solving has been often discussed in the artificial intelligence planning and multiagent research communities recently (e.g., [1]–[4]). Distributed planning has been viewed as either 1) planning for activities and resources allocated among distributed agents; 2) distributed (parallel) computation aimed at plan construction; or 3) plan-merging activity. In this paper, we are solving the problem by algorithms based on task allocation and local resource planning in cooperative environments and use the delegation for continual solution improvement, which is performed in noncritical time. The paper is divided into two parts presenting 1) abstract multiagent solver, problem definition, solver architecture, and algorithms, including discussion of its computational complexity and conditions of strategies admissibility and 2) examples of application areas and description of implemented multiagent systems.

Manuscript received November 19, 2009; revised May 9, 2010; accepted July 26, 2010. Date of current version December 17, 2010. This work was supported by Czech Ministry of Education, Youth and Sports under Grant 6840770038, by Student Grant of Czech Technical University in Prague under Grant SGS10-801890, and by U.S. Army Grant W911NF-08-1-0521 and W15P7T-05-R-P209 under Contract BAA 8020902.A. This paper was recommended by Associate Editor R. W. Brennan.

The authors are with the Agent Technology Center, FEE, Czech Technical University in Prague, Prague, Czech Republic.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCC.2010.2073465

II. Multiagent Solver

Multiagent-planning approaches are used for solving a wide variety of planning problems. As analyzed by Brafman and Domshlak [5], the multiagent-planning techniques can be beneficial for such problems, where the domain sizes of individual agents are considerably smaller (e.g., in logarithmic relation to each other) than the overall size of the problem (even if the planning complexity of an individual agent is exponential) and the number of dependencies between agents is low.

The distributed planning and problem solving has been analyzed by Durfee [1]. One of the related strategies discussed is a *task-sharing* approach. The principle is based on passing of tasks from a busy agent to a vacant agent(s). The process can be summarized in four basic steps.

1) *Task decomposition:* The tasks of agents are decomposed into subtasks. Sharable subtasks are selected.
2) *Task allocation:* The selected tasks are assigned to the vacant agents or agents, which ask for them.
3) *Task accomplishment:* Each agent tries to accomplish its (sub)tasks. The tasks, which need further decomposition are recursively processed and passed to other agents.
4) *Result synthesis:* The results of the tasks are returned to the allocating agent, since it is aware how to use it in the context of the higher tasks.

From the perspective of distributed problem solving, task allocation and the result synthesis are the most crucial parts. However, from the planning perspective, the other two phases are more important. The allocation problem is usually solved by contracting and negotiation techniques, which imply problems related to the resource allocation domain, e.g., cross booking, overbooking, backtracking, and others. In the allocation phase, a hierarchy of agents is established, which may not be fixed in heterogeneous multiagent systems.

The decomposition and delegation principle is widely used in agent-based approaches for problem solving and planning, and shows great applicability to realistic problems. Taking into account Brafman and Domshlak's analysis, Durfee's task-sharing approach efficiency is tightly bound to the solver's ability to

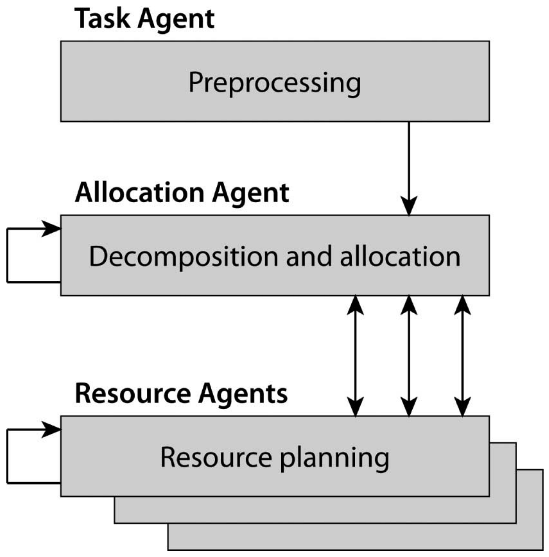


Fig. 1. Abstract architecture of agent-based solver/planner.

reduce the problem sizes for individual agents and keeping the dependencies between agents low.

In the domains, where the optimization/planning problem can be decomposed into independent tasks the multiagent approach shows its benefits. Such a task can be allocated and executed by different agents with low or no influence on each other. In this paper, we assume that such problem decomposition exists. In the rest of the paper, we assume that tasks are independent if not stated otherwise.

We can define the abstract multiagent solver architecture as a composition of three types of agents (see Fig. 1).

1) *Task agent:* This agent is for preprocessing of the problem. It should use a domain-specific heuristic, generic ordering strategy, and randomized method.
2) *Allocation agent:* This agent is for problem decomposition into tasks and delegation of the tasks to resource agents. It maintains task allocation and result synthesis. This agent's strategies and algorithms are domain-independent.
3) *Resource agent:* This agent is for individual case-specific resource planning. In case of further decomposition, the task is handed over to another task agent.

The multiagent system built upon this architecture is composed of one task Agent, one allocation agent, and a set of resource agents. The resource agents represent distributed nature of the multiagent problem.

For complex hierarchical systems, this abstract architecture can be reflected in the multiagent system recurrently, it can be reduced (i.e., one agent undertakes a role of more than one abstract agent type), or it can be parallelized (more abstract solvers are instantiated with potentially overlapping agents, e.g., several task agents or allocation agents handling various problems in parallel). In large systems, concurrent interactions may arise that need to be handled. The agents' interactions are guided by interaction protocols, which are mostly built on Smith's contract net protocol (CNP) [6].

A. Multiagent Problem

The multiagent solver uses the principles of problem decomposition and delegation to autonomous agents that solve parts of the problem individually. The overall solution is then obtained by merging the individual agents' results.

The optimization based on CNP interactions in cooperative environments is usually described as utilitarian social welfare maximization [7]. Therefore, the abstract algorithm objective function can be defined as maximization of social welfare, which is as follows:

$$\mathrm{sw} = \sum_{a\in\mathcal{A}} u_a \tag{1}$$

where $\mathcal{A} = a_1, \ldots, a_n$ is the population of agents and u_a is the utility of agent a. In our case, the social welfare can be computed as a sum of resource agents ($\mathcal{R} \subset \mathcal{A}$) utilities that can be defined as follows:

$$u_a = \sum_{t\in\mathcal{T}_a} (\mathrm{rew}(t) - \mathrm{cost}(t,a)) = \left(\sum_{t\in\mathcal{T}_a} \mathrm{rew}(t)\right) - \mathrm{cost}(\mathcal{T}_a) \tag{2}$$

where $\mathcal{T}_a$ is a set of tasks allocated to the agent $a \in \mathcal{R}$, $\mathrm{rew}(t)$ is a reward for fulfilling task t, $\mathrm{cost}(t,a)$ is the cost of agent a to perform task t, and

$$\mathrm{cost}(\mathcal{T}_a) = \sum_{t\in\mathcal{T}_a} \mathrm{cost}(t,a) \tag{3}$$

is the cost of the overall plan of an agent. The total reward for fulfilling a set of all tasks $\mathcal{T}$ is as follows:

$$\mathrm{rew}(\mathcal{T}) = \sum_{a\in\mathcal{R}} \mathrm{rew}(\mathcal{T}_a) = \sum_{a\in\mathcal{R}} \sum_{t\in\mathcal{T}_a} \mathrm{rew}(t) \tag{4}$$

so the social welfare can be expressed as follows:

$$\mathrm{sw} = \mathrm{rew}(\mathcal{T}) - \sum_{a\in\mathcal{R}} \mathrm{cost}(\mathcal{T}_a) = \mathrm{rew}(\mathcal{T}) - \sum_{t\in\mathcal{T}} \mathrm{cost}(t,a). \tag{5}$$

Since we assume the same quality of task fulfilling by any agent, the reward $k = \mathrm{rew}(\mathcal{T})$ is not influenced by the allocation of tasks to the agents. We can derive social welfare as follows:

$$\mathrm{sw} = k - \sum_{t\in\mathcal{T}} \mathrm{cost}(t,a). \tag{6}$$

As denoted earlier, the goal of CNP-based multiagent optimization in cooperative environments is social welfare maximization. Given by (6), it is the same as minimization of solution cost, where $\mathrm{cost}(t,a)$ is evaluated by the resource agent a undertaking task t. The *objective function* of the abstract solver is then

$$\sum_{t\in\mathcal{T}} \mathrm{cost}(t,a). \tag{7}$$

The task allocation stage of the solver searches for the best suitable mapping of the tasks $\mathcal{T}$ to the resource agents $\mathcal{R}$ that minimizes the objective function given by (7). We can define the goal of the allocation as finding such a partition $\mathcal{P}$ of the set of tasks $\mathcal{T}$ that

$$\underset{\mathcal{P}}{\mathrm{argmin}} \sum_{i=1}^{v} \mathrm{cost}(\mathcal{T}_i) \tag{8}$$

Algorithm 1 The abstract algorithm of a multi-agent solver.

```
Input: Set of tasks $T$, set of Resource Agents $R$
Output: $T$ allocated on $R$
        and local plans of Resource Agents exist

function solve($T$, $R$) begin
    apply ordering heuristic on $T$
    forall $t : T$ begin
        allocateCNP($t$,$R$)
        if allocation not successful then
            exit with failure or
            mark $t$ as not allocated and continue
        end
        forall $a : R$ begin
            apply dynamic improvement strategy
        end
    end
    forall $a : R$ begin
        apply final improvement strategy
    end
end

function allocateCNP($t$, $R$) begin
    forall $a : R$ begin
        find $winner$ with the lowest insertion
          estimation of $t$
    end
    if $winner$ is found then
        assign $t$ to the $winner$
    else
        allocation not successful
    end
end
```

where v is the number of resource agents, $\mathcal{T}_i$ is a subset of tasks allocated to the resource agent a_i, cost($\mathcal{T}_i$) is the cost of the overall plan of agent a_i performing $\mathcal{T}_i$ defined by (3), and

$$\mathcal{T}_i \subseteq \mathcal{T}, \bigcup_{i=1}^{v} \mathcal{T}_i = \mathcal{T} \tag{9}$$

$$\forall i, j : \mathcal{T}_i \cap \mathcal{T}_j = \emptyset \,\text{iff}\, i \neq j. \tag{10}$$

B. Abstract Algorithm

The abstract algorithm representing the presented multiagent solver attempting to minimize objective function defined by (7) is captured by Algorithm 1. According to the abstract architecture depicted in Fig. 1, it contains three phases as following.

1) The first phase of the function `solve` is task preprocessing provided by the task agent. The ordering heuristic represents case-specific sorting of the tasks to increase the solver's efficiency in the particular domain. In some cases, the ordering has no influence, but in others, it may provide significant improvement especially in domains with stronger task dependencies.
2) The second phase is iteration over all tasks and allocation performed by the allocation agent minimizing the insertion cost computed by resource agents (the `allocateCNP` function). As part of this iteration, the dynamic improvement based on cooperation of allocation agent and all resource agents takes place—the improvement strategy is applied to every resource agent after allocation of each task (see following for the description of improvement strategies).
3) The third phase of the `solve` function is the final improvement of the solution. After allocation of all tasks the improvement strategy is executed by all resource agents.

The algorithm is based on local optimization of a single-task insertion and subsequent improvement. Each iteration of the algorithm provides a greedy (order-dependent) task allocation supported by locally optimized solution of resources utilization (which can be seen from global point of view as hill-climbing search). The algorithm does not use any backtracking mechanism or exhaustive search of the state space. It has a significant impact on the algorithm's computational complexity (see Section II-C), but it is susceptible to finding locally efficient solution only. The global solution quality is improved by execution of improvement strategies.

The resource agent uses a case-dependent resource-planning heuristic for these computations. The functions for allocation are as follows.

1) *Insertion estimation cost*$^{estI}(t, a)$*:* The estimation of the cost of the task insertion. It represents the increase of the agent's a cost function caused by undertaking the task t.
2) *Insertion cost*$^{insert}(t, a)$*:* The real cost of the task insertion. This value is determined by adding a new task t to the plan of the agent a in the current state. It is the result of the particular resource-planning algorithm of the resource agent.

The opposite functions used by improvement strategies are as follows.

1) *Removal estimation cost*$^{estR}(t, a)$*:* The estimation of the cost of the task removal. It represents the decrease of the agent's a cost function caused by dropping the task t.
2) *Removal cost*$^{remove}(t, a)$: The real cost of the task removal. This value is determined by removing the task t from the plan of agent a in the current state. It is the result of the particular resource-planning algorithm of the resource agent.

The allocation in the CNP part of the Algorithm 1 is based on the determination of the winner agent. The winner of task t is a resource agent a with the lowest insertion cost; therefore,

$$\text{winner} = \underset{a \in \mathcal{R}}{\text{argmin}}\ \text{cost}^{\text{estI}}(t, a). \tag{11}$$

The allocation agent *allocates* an unallocated task $t \in \mathcal{T}$, where $\forall a_i \in \mathcal{R} : t \notin \mathcal{T}_{a_i}$ to a winner agent a

$$\text{allocate}(\mathcal{T}_a, t) \Rightarrow t \in \mathcal{T}_a \tag{12}$$

provided that local plan of agent a exists and the agent a is able to fulfill this task using the plan for the cost estimation $\text{cost}^{\text{estI}}(t, a)$ used in (11).

Algorithm 2 The abstract algorithm improvement strategies.

```
function improveDW($a$,$R$) begin
    $t_w$ = the $worst\,task$ of agent $a$
    forall $a' : R \setminus a$ begin
        find $winner$ with the lowest $cost^{estI}$ of $t_t$
    end
    if $cost^{estI}$ of $winner$ is lower then $cost^{estR}$ of $a$ then
        delegate $t_w$ from $a$ to $winner$
    end
end

function improveDA($a$,$R$) begin
    forall $t$ : tasks of agent $a$ begin
        forall $a' : R \setminus a$ begin
            find $winner$ with the lowest $cost^{estI}$ of $t$
        end
        if $cost^{estI}$ of $winner$ is lower then $cost^{estR}$ of $a$ then
            delegate $t$ from $a$ to $winner$
        end
    end
end

function improveRA($a$,$R$) begin
    forall $t$ : tasks of agent $a$ begin
        remove $t$ from agent $a$
        allocateCNP($t$,$R$)
    end
end
```

One of the improvement strategies (see following) is based on identification of the most resource consuming task of resource agents. We define *worst task* t_w of agent a as follows:

$$t_w = \underset{t \in \mathcal{T}_a}{\text{argmax}}\ \text{cost}^{\text{estR}}(t,a) \tag{13}$$

i.e., the savings (difference of the total plan cost with and without this task) are maximized.

The improvement strategies basically swap tasks between agents—we say an agent a_i *delegates* task $t \in \mathcal{T}_i$ to an agent a_j

$$\text{delegate}(\mathcal{T}_i, \mathcal{T}_j, t) \Rightarrow t \notin \mathcal{T}_i \wedge t \in \mathcal{T}_j. \tag{14}$$

The *admissible delegation* of task t from agent a_i to agent a_j, where $i \neq j$ is a delegation that satisfies the *improvement condition*

$$\text{cost}^{\text{estR}}(t,a_i) - \text{cost}^{\text{estI}}(t,a_j) > 0. \tag{15}$$

The improvement strategies used by abstract algorithm are as follows (see Algorithm 2 for more details).

1) *Delegate worst* (*DW*)*:* Each resource agent a_i identifies its worst task t_w according to (13) and tries to delegate it to another agent a_j if the improvement condition defined by (15) holds and a_j is the allocation winner according to (11).
2) *Delegate all* (*DA*)*:* Each resource agent a_i delegates all its tasks $\mathcal{T}_i$ to the winner of each task if the improvement condition is satisfied.
3) *Reallocate all* (*RA*)*:* Each resource agent successively removes all its tasks from the plan and allocates them again using the CNP strategy. The result of the allocation can be the same as before task removing, or a change of the position of the task in the current agent plan, or delegation to another agent. To ensure proper function of RA improvement strategy, we require for successive removing and inserting of task t from/to agent a that

$$\text{cost}^{\text{insert}}(t,a) \le \text{cost}^{\text{remove}}(t,a) \tag{16}$$

i.e., when removing and reinserting task $t \in \mathcal{T}_a$, the cost($\mathcal{T}_a$) does not increase.

C. Complexity Analysis

The general computational complexity of the multiagent solver is introduced in [5]. Using transformation of the multiagent-planning problem to the distributed constraint satisfaction problem, the worst case time complexity of the multiagent planning is upper bounded by

$$f(\mathcal{I}) \times \text{exp(comm)} + \text{exp(int)} \tag{17}$$

where $f(\cdot)$ is the factor inducted by requesting each agent to plan, while committing to a certain sequence of actions, $\mathcal{I}$ is the complexity of an individual agent's planning, exp(comm) represents a factor exponential in min–max number of per-agent commitments, and an additive factor exp(int) represents the interactions of agents.

The consequences of (17) lead to interesting features of the multiagent solver, such as that there is 1) no direct exponential dependence on the number of agents; 2) no direct exponential dependence on the size of the planning problem or size of the joint plan; and 3) no direct exponential dependence on the length of individual agent plans [5].

In our case, the feature 2) resulting from (17) does not have a strong impact if the decomposition algorithm of the allocation agent is exponential because the size of its problem is the same as the size of the overall problem, but for the resource agents (and other subordinate agents in the case of a complex hierarchical structure) this feature holds. However, the exponential factors are usually reduced by the polynomial heuristics—decomposition, allocation, optimization, and resource-planning strategies implemented in real applications. The ordering strategy of the task agent does not have a strong influence on the worst case complexity because of its additive nature and low complexity (provided that tasks can be compared in constant time). The multiagent solver benefits in the domains, where problem can be easily decomposed to independent tasks, and/or where the polynomial heuristics for the resource planning exist.

For the abstract algorithm (see Algorithm 1), (17) can be represented as follows:

$$n \times \log(n) + n \times O^{\text{alloc}} + n \times m \times O^{\text{impr}} \tag{18}$$

where n denotes the number of tasks and m is the number of resource agents. The first part represents the ordering heuristic and its complexity corresponds to the complexity of standard sorting algorithms. The middle part is the complexity of the allocation part of the algorithm and the last part is the improvement part of the algorithm.

The allocation and improvement time complexities of the algorithm are defined as follows:

$$O^{\mathrm{alloc}} = m \times O(\mathrm{cost}^{\mathrm{estI}}(t,a)) + O(\mathrm{cost}^{\mathrm{insert}}(t,a))$$
$$O^{\mathrm{impr}} = fi(n')$$

where $n' = n/m$ is the average number of tasks allocated to a particular resource agent and $fi(n')$ is the factor representing the complexity of the implemented agents' improvement strategy. We assume

$$O(\mathrm{cost}^{\mathrm{estI}}(t,a)) = O(\mathrm{cost}^{\mathrm{insert}}(t,a)) = fr(n')$$
$$O(\mathrm{cost}^{\mathrm{estR}}(t,a)) = O(\mathrm{cost}^{\mathrm{removal}}(t,a)) = fr'(n')$$

where $fr(n')$ is the factor representing the complexity of the implemented agents' resource-planning strategy for the task insertion and $fr'(n')$ is the factor representing the complexity of the implemented agents' resource-planning strategy for the task removal. Therefore, the general worst case time complexity of the presented abstract algorithm is as follows:

$$n \times (\log(n) + m \times (fr(n') + fi(n'))). \qquad (19)$$

The complexity of improvement strategies defined in Section II-B are as follows:

$$fi^{\mathrm{DW}}(n') = O^{\mathrm{worst}} + fr'(n') + m \times fr(n')$$
$$fi^{\mathrm{DA}}(n') = fi^{\mathrm{RA}}(n') = n' \times (fr'(n') + m \times fr(n'))$$

where O^{worst} is the complexity of identification of the worst task in the plan, i.e., finding the task with greater $\mathrm{cost}^{\mathrm{estR}}(t,a)$. It can be upper bounded by the iteration through all n' tasks and computation of removal cost of each one, so its worst case complexity is upper bounded by $O^{\mathrm{worst}} = O(n' \times fr'(n'))$. For combination of all described improvement strategies, the improvement complexity is upper bounded by

$$fi(n') = n' \times fr'(n') + m \times n' \times fr(n'). \qquad (20)$$

The factor $m \times n' = m \times (n/m) = n$, so the improvement complexity has no relation to the number of agents. Combining (19) and (20) and taking $n' = n$, the *worst case time complexity* of the abstract algorithm is as follows:

$$n \times \log(n) + n^2 \times fr'(n) + m \times n^2 \times fr(n). \qquad (21)$$

The presented complexity analysis shows the polynomial impact of the decomposition and delegation principles used by the multiagent abstract solver. The impact of the two factors introduced by (17) is the following:

1) the complexity of the operations of resource agent are multiplied by n^2;
2) the influence of the number of resource agents is linear.

The complexity analysis shows us an important feature of agent-base solver. When using polynomial heuristics for task insertion and removal, the implemented multiagent solver provides polynomial worst case complexity. Together with linear computational scaling with the number of agents makes the presented abstract architecture suitable for many application areas.

Algorithm 3 The incremental improvement algorithm.

```
Input: Set of Resource Agents $R$
Output: Improved solution

function improve($R$) begin
    $improvement$ := $true$
    repeat until $improvement$ is $false$
        $improvement$ := $false$
        forall $a$ : $R$ begin
            apply dynamic improvement strategy
            if solution has been improved then
                $improvement$ := $true$
            end
        end
    end
end
```

In real-application areas, ordering heuristics can be found that result in allocation with no need of using improvement strategies (e.g., production-planning system described in Section III-D). In other cases, the planning of resource agents is implemented with low complexity (e.g., linear) and the improvement strategy has greater importance. In Section III, we present several applications developed using the described abstract multiagent solver.

D. Incremental Improvement Strategy

The basic multiagent solver can be enhanced by the incremental improvement strategy. Algorithm 1 allocates the tasks and runs improvement strategies (both dynamic and final), keeping the computational complexity low (i.e., guaranteeing a good response time of the algorithm). In many cases, it is beneficial to perform incremental improvement of the solution when we have enough time to wait for better results or the environment changes dynamically. The changes of the task constraints, resources availability, and execution uncertainty affects the efficiency of the static plan during execution. Algorithm 1 optimizes (7) in a single run upon conditions that are valid at the time of algorithm execution. When some conditions change, the solution quality diverts. In such a case, the incremental improvement strategy should be used to keep the solution up to date with the dynamic environment changes and/or improve the solution over the time.

The nature of the task allocation process of the multiagent solver enables us to run the improvement strategies continuously and interrupt its improvement during any run without loosing the correctness of the solution (assuming the delegation as atomic process). The incremental improvement strategy described in Algorithm 3 is an anytime algorithm monotonically improving the quality of the solution. The improvement strategies (see Algorithm 2) are executed until the solution is no longer being improved (i.e., the overall cost of the solution defined by (7) does not change between iterations).

The inner loop iterates over all resource agents and provides the same features (complexity and convergence) as the improvement part of Algorithm 1. The outer loop terminates when the quality of two consequent solutions is the same.

E. Resource Agent Strategy Admissibility

The presented multiagent solver features strongly depends on the functions provided by the resource agents. We investigate the allocation process correctness and delegation stability, and define the constraints to functions provided by admissible resource agent strategy.

When using standard CNP, we require the estimation functions of resource agent a to provide accurate estimations of inserting/removal of task t, e.g.,

$$\mathrm{cost}^{\mathrm{estI}}(a,t) = \mathrm{cost}^{\mathrm{insert}}(a,t) \tag{22}$$

$$\mathrm{cost}^{\mathrm{estR}}(a,t) = \mathrm{cost}^{\mathrm{remove}}(a,t). \tag{23}$$

In the case of more advanced allocation protocols with backtracking (e.g., extended CNP [8]), this accuracy can be relaxed. In all cases, these constraints defined on the resource agent functions ensure that Algorithm 1 is able to find a local minimum of the cost function (7) and to guarantee the proper behavior of improvement strategies.

The estimation functions provided by resource agent strategy are *admissible* only if

$$\mathrm{cost}^{\mathrm{estI}}(a,t) \leq \mathrm{cost}^{\mathrm{insert}}(a,t) \tag{24}$$

$$\mathrm{cost}^{\mathrm{estR}}(a,t) \geq \mathrm{cost}^{\mathrm{remove}}(a,t) \tag{25}$$

where for the upper bound of the estimation error holds

$$|\mathrm{cost}^{\mathrm{insert}}(a,t) - \mathrm{cost}^{\mathrm{estI}}(a,t)| < \varepsilon_I \tag{26}$$

$$|\mathrm{cost}^{\mathrm{remove}}(a,t) - \mathrm{cost}^{\mathrm{estR}}(a,t)| < \varepsilon_R \tag{27}$$

and delegation improvement condition according to (14) and (15) is modified to

$$\mathrm{cost}^{\mathrm{estR}}(t,a_i) - \mathrm{cost}^{\mathrm{estI}}(t,a_j) > \varepsilon_I + \varepsilon_R. \tag{28}$$

Keeping the defined constraints (improvement condition, estimation error, and admissibility of estimation functions), the execution of any of the improvement strategies results in reduction of the solution cost or the solution is unaffected (i.e., the new solution is not worse than the previous one) and the incremental improvement strategy termination is ensured after a finite number of steps (if the environment does not change during the execution of the improvement algorithm).

Using features discussed in previous sections, we can define the *admissible resource agent strategy* (its internal heuristics and estimation, and allocation functions) has to:

1) use admissible estimation functions according to (24) and (25);
2) bound the estimation error according to (26) and (27);
3) fulfill the improvement condition defined by (28) (the delegation is admissible if the solution cost improves);
4) when using the reallocate-all improvement strategy, (16) must hold (i.e., when reallocating, the new solution has to be better or at least the same as the previous one);
5) the algorithm execution time must be low compared to the frequency of the environment changes (the environment is considered static during the execution of the algorithm).

III. Applications

Due to high flexibility, robustness, and scalability, the agent technologies promise a wide industrial applicability [9] even in the real-time environments with a need of dynamic reconfiguration and replanning [10]. This section presents four examples of the multiagent systems implemented using the presented abstract architecture. It demonstrates the applicability of the concept in a wide variety of real-problem domains—vehicle routing problems (VRPs), strategic mission planning, multirobot frontiers exploration, and production planning. All presented multiagent systems share the same abstract architecture, algorithm, and improvement strategies. For each application domain, a particular resource optimization heuristic has been designed and implemented by the resource agents. All the presented systems' implementations have low-computational complexity and provide high-quality solutions.

A. VRP Solver

The VRP is a well-known optimization problem. The problem is NP-hard and is defined as routing of a fleet of gasoline delivery trucks between a terminal and a number of service stations. The trucks have load capacity limitations and deliveries have to be accomplished at minimum total cost (distance traveled).

The agent-based approach to a variant of the VRP solver has been presented, for example, in [11]. Zeddini *et al.* use three types of agents—client, bidder, and vehicle agents. The approach is based on the CNP allocation and the optimization is based on exchange of tasks between the vehicle agents. The vehicle agents use an insertion heuristic and improvement strategy for task swapping between them. The error of the solution (compared to the optimal solution) presented in the paper is 4%–29% for standard benchmark problems.

A similar approach has been used in [12] for a dynamic variant of *k*-VRP (new tasks are added during the execution), where the initial allocation is generated using a centralized algorithm. The dynamic task allocation is made by the CNP protocol. Then, two improvement phases are applied. The *intraroute optimization* is applied to each agent route and *interroute optimization* is performed between vehicle agents.

The VRP family solver built upon an abstract multiagent solver, which is introduced by this paper is presented in [13]. The solver uses three ordering strategies, the combination of all improvement strategies with dynamic improvement enabled or disabled, and the resource agent algorithm implemented using a standard cheapest insertion heuristic for traveling salesman problem [14].

The objective function is based on the distance traveled by vehicles (represented by resource agents) and fully corresponds to (7). The implemented RA strategy fulfills the improvement conditions defined by (15) and (16) and admissibility conditions defined by (22) and (23).

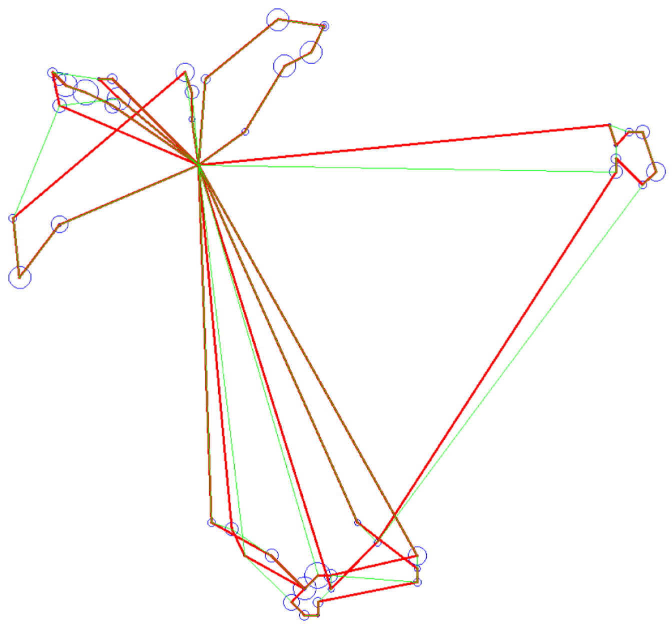


Fig. 2. Example of the VRP solver result.

The complexity of the resource agent cost computation functions is kept low, i.e., $fr(n') = O(n')$ and $fr'(n') = O(1)$. The worst case time complexity defined by (21) (using $n' = n/m$) is $O(n^3)$, which has been also proved experimentally on benchmark instances.

The example of the solution provided by the solver is in Fig. 2. There are 52 nodes served by seven vehicles (an optimal number of vehicles has been obtained). The circles in the nodes represent the size of transportation demand. The quality of the solution to the problem presented in Fig. 2 is 93.4% compared to the optimal path length. We measure the quality as $\frac{\text{cost}_{\text{solver}}}{\text{cost}_{\text{optim}}} \times 100[\%]$, where $\text{cost}_{\text{solver}}$ is the solution cost provided by the solver and $\text{cost}_{\text{optim}}$ is the cost of the best known solution. The optimal vehicle's path is depicted by thin green lines and the solution produced by the multiagent solver is represented by thick red lines.

For the 115 evaluated benchmark instances (standard VRP benchmark problems), the multiagent solver provides solutions with the quality of at least 81% compared to the optimal solution with average quality better than 91% (corresponds to solution error of 19%, respectively, 9%). The self-organizing capability of the system successfully minimizes the number of vehicles used. The results show that the application of dynamic improvement strategy provides better results than batch processing with final improvement strategy only. The best performance has been reached using DA and RA improvement strategies. The implemented solver demonstrates very good applicability of the abstract multiagent solver to the family of routing problems and easy adaptation to problem variants.

B. *Strategic Mission Planning Using Social Commitments*

The multiagent abstract solver presented in this paper has been used in a system for distributed planning and coordination in dynamic nondeterministic multiactor mixed-initiative environment [15]. The system provides flexible planning, replanning, and task allocation. The system addresses several issues that have to be solved in order to fulfill the requirements on a system planning in dynamic nondeterministic environments. An overview of the problems is as follows.

1) *Distributed planning:* Planning in such an environment is practically realizable only as a distributed process.
2) *Distributed resource allocation:* An integral part of the planning process is resource allocation both of the acting entities in the world and of the static resources, and as the planning process is distributed, the resource allocation has to be distributed as well.
3) *Distributed plan execution and synchronization:* Constituted distributed plan consisting of several personal plans has to be executed by the entities. The personal plans need to be coordinated in distributed manner, as the entities do not know each other's plans.

From the perspective of allocation agent, the planning process can be divided into three phases: 1) the hierarchical task network I-Plan planner [16] is used for creating an abstract plan for a long-time horizon; 2) the plan instantiating process uses a distributed resource allocation based on the CNP [6]; and 3) with the help of this protocol, the appropriate subordinate agents (resource agents) are found and the responsibilities for the plan actions are fixed. The internal resource agents optimization uses the as-early-as-possible scheduling heuristic. The heuristic causes the earliest possible execution of the plan actions, which affects the length of the whole plan in the nondeterministic environment. The effect is directly proportional to the amount of nondeterminism in the world (which has been simulated as random prolonging of the actions durations). The agents' hierarchy in the system is captured by Fig. 4, there is a commander creating tasks for builders, builders implementing the abstract task agent and allocation agent roles in parallel, and finally, we have a set of trucks implementing the abstract resource agent roles.

All plans are described in the form of social commitments substituting plan actions. The commitment is a knowledge-base structure describing agent's obligation to change the world-state and a set of rules defining what the agent should do if the obligation is not satisfiable [17]. The commitments enable expressive description of the decommitment rules, and thus, the replanning process, it captures the improvement strategies executed on the moment when the environment changes in the way that collide with the plan.

In other words, the replanning process (i.e., the plan improvement process) by means of social commitments can be described as successive recommitting [17]. For the decommitting purposes, three basic decommitment rules were used: full decommitment, delegation, and relaxation [18].

The deployment scenario of the system is a disaster relief operation, where the resources have to be transported to the impacted zone [19]. There are material resources, emergency units, and transport units in the scenario. The emergency units create plans and request the transportation for itself and for required material resources. The problem is to find such a global

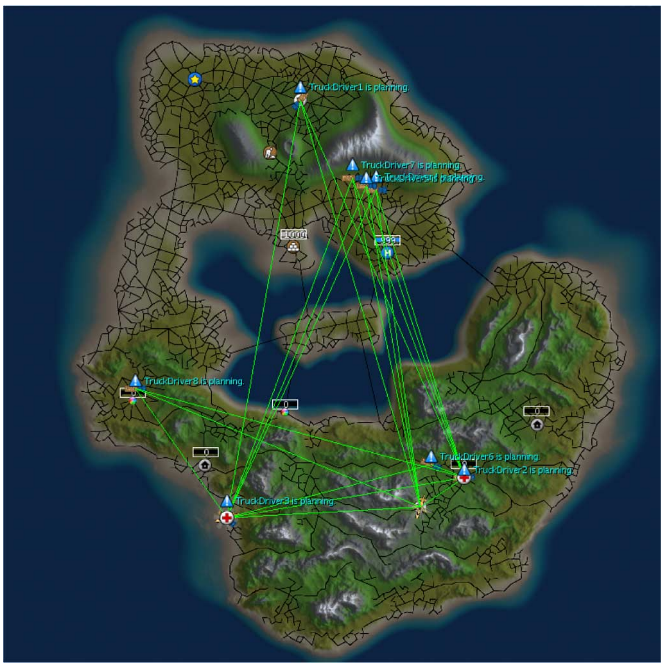


Fig. 3. Strategic mission planning system—scenario island screenshot.

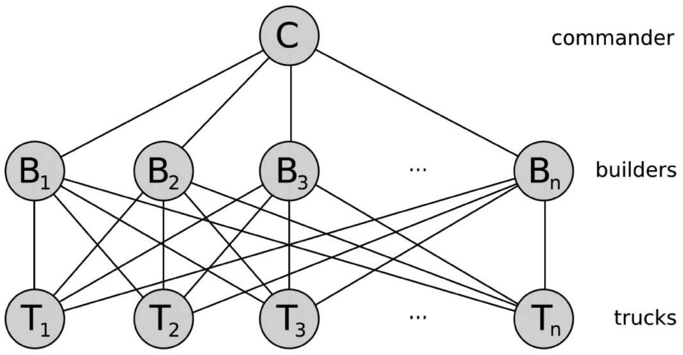


Fig. 4. Strategic mission planning system—hierarchy of agents.

plan that all units and material are transported to the demanded area as soon as possible using limited transportation resources (see a screenshot of the scenario island in Fig. 3).

The entire system provides fast convergence to the efficient solution with time complexity of $O(n^3)$. The heuristic of the resource agent strategy is admissible in terms defined in Section II-B and provides $fr(n') = O(1)$ and $fr'(n') = O(n')$. The plan execution stability in the dynamic environment is enhanced using improvement strategies captured by decommitment rules, and thus, incrementally invoked abstract algorithm presented in Section III.

C. Cooperative Frontiers Exploration

The presented multiagent solver has been also utilized for cooperative frontiers exploration problem [20]. The problem is to find the shortest path for a convoy of vehicles through a partially known urban area. The street map of the area is *a priori* known, but the actual condition of the routes is not. There is the convoy moving through the city using a path-planning algorithm that incorporates the information obtained by a set of small autonomous vehicles. These vehicles explore the area ahead of the convoy (see the scenario screenshot in Fig. 5).

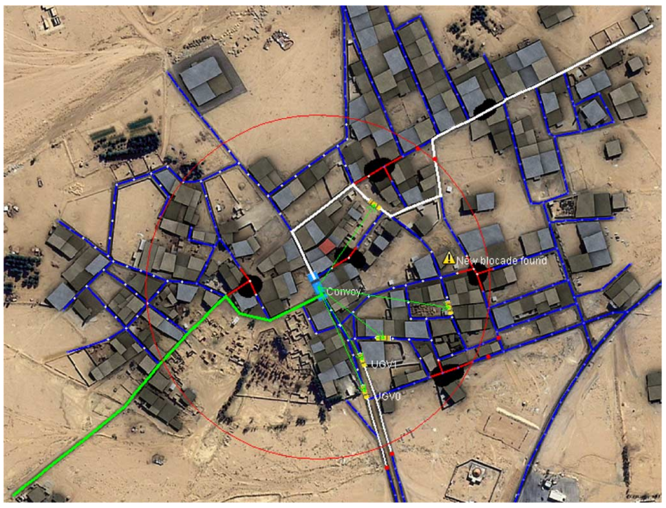


Fig. 5. Cooperative frontiers exploration—scenario screenshot.

The goal of the multiagent system is to find the shortest path through the area ensuring that 1) the convoy will not stop or u-turn because of a street blockade and 2) the total traveled path of all vehicles is minimized. The multiagent system is composed of one convoy agent (task agent and allocation agent role) and a set of unmanned ground vehicles (UGV) agents (resource agent role). Besides path planning and navigating through the city, the convoy agent generates a set of frontiers for exploration [21]. In this case, it is not a classical mapping task in an unknown area, but the frontiers represent points of interests in the known street map that should be investigated to ensure the convoy may freely pass through the desired route. A similar robotic problem has been also solved using a multiagent system for distributed robotic planning, e.g., in [22]. The first goal (convoy nonstoping movement toward a target) is secured using the incremental re-planning algorithm for convoy path planning based on anytime planning algorithm D-star [23]. The second goal (minimization of the traveled path) is handled by a multiagent solver using the presented architecture and algorithms. It is shown [20] that the multiagent solver provides almost real-time response and significantly reduces the convoy traveled path even with small number of UGVs keeping the overall traveled distance overhead low.

The convoy agent dynamically creates points of interests (exploration frontiers) and allocates them to the UGV agents using Algorithm 1. In the case of any new information is discovered, Algorithm 3 is used and also new frontiers are generated and allocated (or some frontiers can be removed). The UGV agents use a route optimization heuristic that attempts to minimize the traveled distance similar to the VRP described earlier. It fulfills the improvement condition defined by (15) and admissibility conditions defined by (24) and (25). The computational complexities of the strategy are $fr(n') = O(n')$ and $fr'(n') = O(1)$. The experimentally evaluated worst case time complexity of the multiagent solver has been upper bounded by $O(n^3)$.

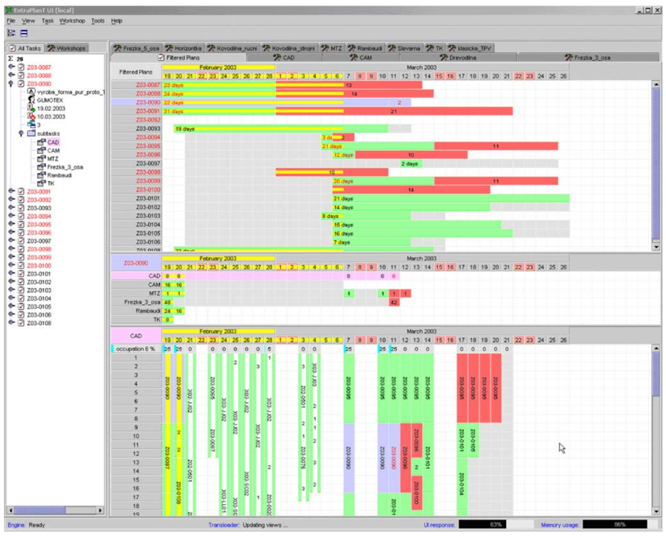


Fig. 6. Screenshot of multiagent production-planning system.

D. Production Planning

Classical-planning systems (using scheduling algorithms with various heuristics, constraint logic programming, genetic algorithms, and simulated annealing [24], [25]) work centrally and allocate resources usually in one run for every product (order) presented in the system. These methods use mostly stochastic algorithms and generate near-optimal solutions for minimization of defined criteria (e.g., a sum of weighted tardiness and inventory costs). Such a solution is fully sufficient, while the required replanning and rescheduling affects the entire plan. The plan is usually completely rebuilt and a random aspect of the algorithms can cause major (unwanted) changes in plans after replanning. This might not be suitable for many manufacturing areas. With physical distribution of the production units, it is advantageous to decompose and distribute the planning problem [1].

The multiagent technology addresses all the phases of the manufacturing decision-making support, while there are few implementations of multi-agent systems that cover more than a single stage. There are solutions for low-level scheduling or control systems, the product configuration and quotation phases to short-term and long-term production planning and supply-chain management [26].

The example of the system that uses the same conceptual fundamentals as multiagent abstract solver presented by this paper is presented in [27] and [28]. One of the goals of the multiagent system is to create a production plan for middle- and long-term horizon to give an overview of resource utilization (see Fig. 6 for a screenshot of a multiagent production-planning system). The production resources are represented by the resource agents maintaining the constrains and capabilities of individual production workshops. The task agent uses a classical task ordering heuristic based on weighted earliest deadline first, which significantly improves the solution and there is no need for improvement strategy execution after allocation phase of the batch of tasks. The production-order decomposition and planning is provided by the planning agent, which allocates the parts of the production order to the resource agents using the CNP according to Algorithm 1. In case of environment changes (e.g., an update of production times estimation, machines breakdowns, delays in production, etc.), Algorithm 3 is executed. The solution of the solver is optimal according to the cost computed as a weighted delay penalty.

Resource agents use an admissible strategy [according to (22), (23), and (15)] that minimizes the weighted average delay of the production orders (see [29] for more details). The computational complexities of resource agent algorithms are $fr(n') = fr'(n') = O(n')$; therefore, the overall solver complexity is upper bounded by $O(n^4)$.

IV. Conclusion

This paper describes an abstract multiagent solver architecture and an algorithm for implementing a wide variety of practical multiagent-planning and problem-solving systems. The algorithm maximizing social welfare of the cooperative agent community is introduced and analyzed. CNP-based task allocation and solution improvement using task delegation provides a powerful tool for problem solving when keeping the computational complexity within reasonable limits. We also discuss the limitations and admissibility constraints of the resource optimization heuristic that has to be designed to implement the multiagent solver for a particular problem domain and define the resource agent strategy admissibility.

The solver benefits in domains, where decomposition of a problem into independent tasks is possible. Finding independent subsets of tasks significantly reduces the need for interactions between agents. The tasks are planned and executed by individual agents independently, and the overall solution is then merged from the partial ones. The system is able to balance the allocation of the tasks to the agents and provide anytime monotonically improved solution.

The applicability of the presented abstract multiagent solver is demonstrated on several real systems operating in the domains of VRPs, strategic mission planning, multirobot frontiers exploration, and production planning. In all application areas, the implemented system provides low-computational complexity ($O(n^3)$ or $O(n^4)$) with good solution quality.

Acknowledgment

The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and online copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

References

[1] E. H. Durfee, "Distributed problem solving and planning," in *A Modern Approach to Distributed Artificial Intelligence*, G. Weiß, Ed. San Francisco, CA: The MIT Press, 1999, ch. 3.

[2] M. E. DesJardins and M. J. Wolverton, "Coordinating a distributed planning system," *AI Mag.*, vol. 20, no. 4, pp. 45–53, 1999.
[3] E. Ephrati and J. S. Rosenschein, "A heuristic technique for multiagent planning," *Ann. Mathematics Artificial Intell.*, vol. 20, no. 1–4, pp. 13–67, 1997.
[4] M. M. de Weerdt, A. Bos, J. Tonino, and C. Witteveen, "A resource logic for multi-agent plan merging," *Ann. Mathematics Artificial Intell., Special Issue Comput. Logic Multi-Agent Syst.*, vol. 37, no. 1–2, pp. 93–130, Jan 2003.
[5] R. I. Brafman and C. Domshlak, "From one to many: Planning for loosely coupled multi-agent systems," in *ICAPS*, 2008, pp. 28–35.
[6] R. G. Smith, "The contract net protocol: High level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.
[7] K. J. Arrow, A. K. Sen, and K. Suzumura, Eds., *Handbook of Social Choice and Welfare (Handbooks in Economics)*. Amsterdam: North Holland, Aug. 2002.
[8] K. Fischer, J. P. Muller, M. Pischel, and D. Schier, "A model for cooperative transportation scheduling," in *Proc. First Int. Conf. Multiagent Syst.* Menlo park, California: AAAI Press/MIT Press, Jun. 1995, pp. 109–116.
[9] V. Marik and J. Lazansky, "Industrial applications of agent technologies," *Control Eng. Practice*, vol. 15, no. 11, pp. 1364–1380, 2007.
[10] P. Vrba and V. Marik, "Capabilities of dynamic reconfiguration of multiagent-based industrial control systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 40, no. 2, pp. 213–223, Mar. 2010.
[11] B. Zeddini, M. Temani, A. Yassine, and K. Ghedira, "An agent-oriented approach for the dynamic vehicle routing problem," in *2008 Int. Workshop Adv. Inf. Syst. Enterprises*, vol. 0, pp. 70–76, 2008.
[12] D. Barbucha and P. Jedrzejowicz, "Agent-based approach to the dynamic vehicle routing problem," in *7th Int. Conf. Pract. Appl. Agents Multi-Agent Syst,* ser. Advances in Soft Computing, vol. 55. Springer Berlin/Heidelberg, 2009, pp. 169–178.
[13] J. Vokrinek, A. Komenda, and M. Pechoucek, "Agents towards vehicle routing problems," in *Proc. Int. Joint Conf. Autonomous Agents Multiagent Syst*, pp. 773–780.
[14] D. J. Rosenkrantz, R. E. Stearns, and P. M. L. II, "An analysis of several heuristics for the traveling salesman problem," *SIAM J. Comput.*, vol. 6, no. 3, pp. 563–581, 1977.
[15] A. Komenda, J. Vokřínek, M. Pěchouček, G. Wickler, J. Dalton, and A. Tate, "I-globe: Distributed planning and coordination of mixed-initiative activities," in *Proc. Knowl. Syst. Coalition Operations 2009*, Chilworth Manor, Southampton, U.K., Mar.–Apr.
[16] A. Tate, "Intelligible ai planning," in *Proc. 20th ES Res. Develop. Intell. Syst. XVII*, M. Bramer, A. Preece, and F. Coenen, Eds. Springer, 2000, pp. 3–16.
[17] A. Komenda, M. Pěchouček, J. Bíba, and J. Vokřínek, " Planning and re-planning in multi-actors scenarios by means of social commitments," in *Proc. Int. Multiconf. Comput. Sci. Inf. Technol.*, vol. 3. IEEE, Oct. 2008, pp. 39–45.
[18] J. Vokřínek, A. Komenda, and M. Pěchouček, "Decommitting in multi-agent execution in non-deterministic environment: Experimental approach," in *Proc. 8th Int. Joint Conf. Autonomous Agents Multiagent Syst.*, 2009, pp. 977–984.
[19] A. Komenda, J. Vokřínek, M. Pěchouček, G. Wickler, J. Dalton, and A. Tate, "Distributed planning and coordination in non-deterministic environments (demo)," in *Proc. 8th Int. Joint Conf. Autonomous Agents Multiagent Syst*, 2009, pp. 1401–1402.
[20] J. Vokrinek, A. Komenda, and M. Pechoucek, "Cooperative agent navigation in partially unknown urban environments," in *Proc. 3rd Int. Symp. Practical Cognitive Agents Robots.*, 2010, pp. 46–53.
[21] A. Visser, Xingrui-Ji, M. van Ittersum, L. A. G. ?lez Jaime, and L. A. Stancu, "Beyond frontier exploration," in *RoboCup 2007: Robot Soccer World Cup XI*, ser. Lecture Notes in Computer Science, vol. 5001. Springer Berlin/Heidelberg, 2008, pp. 113–123.
[22] S. Thayer, B. Digney, M. B. Dias, A. T. Stentz, B. Nabbe, and M. Hebert, "Distributed robotic mapping of extreme environments," in *Proc. SPIE: Mobile Robots XV and Telemanipulator Telepresence Technologies VII*, vol. 4195, Nov. 2000, pp. 84–95.
[23] M. Likhachev, D. Ferguson, G. Gordon, A. T. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," in *Proc. Int. Conf. Automated Planning Scheduling*, Jun. 2005, pp. 262–271.
[24] N. Sadeh and M. S. Fox, "Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem," *Artif. Intell.*, vol. 86, no. 1, pp. 1–41, 1996.
[25] J. K. k, L. Rothkrantz and J. L. 1/2, "Genetic algorithms with limited convergence," in *Information Processing with Evolutionary Algorithms*, ser. Advanced Information and Knowledge Processing. London UK: Springer, 2005, pp. 233–253.
[26] M. Pěchouček, J. Vokřínek, and P. Bečvář, "Explantech: Multiagent support for manufacturing decision making," *IEEE Intell. Syst.*, vol. 20, no. 1, pp. 67–74, Jan./Feb. 2005.
[27] J. Vokřínek, J. Hodík, M. Pěchouček, P. Bečvář, and J. Pospíšil, *ExtraPlanT as a Multi-Agent System for Extra-Enterprise Collaboration*. Information science Reference, 2008, ch. 593-600, pp. 593–600.
[28] J. Hodík, P. Bečvář, M. Pěchouček, J. Vokřínek, and J. Pospíšil, "Explantech and extraplant: multi-agent technology for production planning, simulation and extra-enterprise collaboration," *Int. J. Comput. Syst. Sci. Eng.*, vol. 20, no. 5, pp. 357–367, 2005.
[29] P. Bečvář, P. Charvát, M. P. J. Pospíšil, A. Říha, and J. Vokřínek, "Explantech/extraplant: Production planning and supply-chain management multi-agent solution," *EXP - in search innovation*, vol. 3, no. 3, pp. 116–125, 2003.

Jiří Vokřínek received the Master's degree in technical cybernetics at Czech Technical University in Prague, Prague, Czech Republic, he is working toward the Ph.D. degree at the Agent Technology Center, Department of Cybernetics at Czech Technical University in Prague.

He is a Research Scientist at the Agent Technology Center. His research interests include agent-based planning and simulation, artificial intelligence, multiagent systems, planning and replanning in manufacturing, virtual organizations, supply-chain management, and logistics.

Antonín Komenda received the Master's degree in Technical Cybernetics at Czech Technical University in Prague, Prague, Czech Republic. He is currently working toward the Ph.D. degree from Agent Technology Center of the Department of Cybernetics at Czech Technical University in Prague. His studies was focused on multiagent systems.

He is also a Researcher at the Agent Technology Center. His current research interests include multiagent systems, distributed planning and plan repairing.

Michal Pěchouček received the Master's degree in IT: Knowledge-based systems from the University of Edinburgh, Edinburgh, U.K., and the Ph.D. degree in artificial intelligence and biocybernetics from Czech Technical University in Prague, Prague, Czech Republic.

He is a Professor in artificial intelligence at the Department of Cybernetics, Czech Technical University in Prague. He is currently a Head of Agent Technology Center, Department of Cybernetics, CTU. His research interests include problems related to multiagent systems, especially topics related to social knowledge, metareasoning, acting in communication inaccessibility, coalition formation, agent reflection, and multiagent planning.

ExPlanTech: Multiagent Support for Manufacturing Decision Making

Michal Pěchouček and Jiří Vokřinek, *Gerstner Laboratory, Czech Technical University*

Petr Bečvář, *CertiCon*

ExPlanTech's multiagent approach offers a unified framework for decision-making support and provides a proven alternative to known mathematical and system science-modeling technologies for simulating the manufacturing process.

ExPlanTech is a consolidated technological framework resulting from a series of European Union Research & Technological Development and Trial projects in agent-based production planning. ExPlanTech provides technological support for various manufacturing problems and comprises different components, which you can assemble to develop a customized system that supports a user's decision making in different aspects of production planning. The system should help human users size resources and time requirements for a particular order, creating production plans, optimizing material resources manipulation, managing and optimizing supply chain relationships, visualizing and analyzing medium- and long-term manufacturing processes, and accessing external data.

Using ExPlanTech's multiagent architecture as a foundation for a software system, you can create a component-based, flexible, and reconfigurable system that allows distributed computation and flexible data management. In Gersntner Laboratory, we integrated each ExPlanTech component in an agent wrapper that complies with the FIPA (Foundation for Intelligent Physical Agents, www.fipa.org) standard for heterogeneous software agents. You can use these components in various configurations or independently as standalone applications. System configurations can contain various planning, data-management, or visualization agents. ExPlanTech also offers an *agentification* process that integrates an enterprise's existing software and hardware into an FIPA-compliant agent.

Deploying agent technologies in manufacturing problems (see the related sidebar) lets you process relevant production data distributed across the enterprise. A classical approach that collects and processes data centrally has difficulty dealing with situations with voluminous and frequently changing production-planning data. An agent approach lets you process data proactively at its place of origin and exchange only necessary results. ExPlanTech, or any agent-based technology, certainly doesn't provide an uncomplicated solution of NP-hard planning problems. However, agent technology lets you integrate heavy-duty AI problem solvers (such as constraint satisfaction systems, linear programming tools, genetic algorithms, and so on). Such technology works well for integrating manufacturing enterprises into a supply chain. In terms of planning, it's irrelevant whether a system reasons about an in-house manufacturing workshop or about a subcontracted company. Additionally, production managers are often interested in production process modeling and simulation. Using ExPlanTech in a simulation environment can simplify experimenting with changes to production lines and help show how they affect the manufacturing process. ExPlanTech doesn't offer agents and components for control and real-time diagnostics. Even though ExPlanTech technology is in principle open to providing support for control and real-time diagnostics, the current implementation doesn't feature any such agents or components.

ExPlanTech

Given our long experience working with the manufacturing industry, we identified several key requirements that industrial users in manufacturing frequently request. They want a system that is

- *Open, extensible, and general.* Allows customization for several different domains and lets you expand functionality (such as extending production planning to supply chain management).

Agent Technologies and Multiagent Systems

Agent technologies and the concept of *multiagent systems* (MASs) originated in artificial intelligence and computer science, drawing from principles of component-based software engineering, distributed decision making, parallel and distributed computing, autonomous computing, and advanced methods of interoperability and software integration.[1] Agent-based system operations are based on collaborative (or sometimes self-interested) interactions of autonomous and loosely coupled software or hardware entities—*agents*. An agent can integrate existing software systems required for operating the manufacturing enterprise, hardware modules such as computer numeric control machines, and various programmable logic controllers with advanced planning systems, simulation environments, diagnostic algorithms, or sophisticated control mechanisms.

Agents technologies suit domains that have one of the following properties:

- Requires solving highly complex problems or controlling highly complex systems
- Has distributed, not centrally available information required for solving problems or controlling systems
- Has a dynamically changing environment and problem specification
- Must integrate a high number of heterogeneous software (and possibly hardware) systems

Application areas

Several agent technology application areas typically relate to manufacturing. In production, we solve highly complex planning problems, so we must control dynamic, unpredictable, and unstable processes. We might also need agent-based diagnostics, repair, reconfiguration, and replanning.

For virtual enterprises and supply chain management, we have requirements for forming business alliances, planning long-term and short-term cooperation deals, and managing (including reconfiguration and dissolving) supply chains. So, we also can use various agent technologies for agents' private knowledge maintenance, specification of various ontologies, and ensuring service interoperability across the supply chain. For Internet-based business, we can use agent technologies for intelligent shopping and auctioning, information retrieval and searching, remote access to information, and remote system control.

Additionally, we can use MASs to manage transportation and material handling and for optimal planning and scheduling—especially in cargo transportation, public transport, peacekeeping missions, military maneuvers, and so on. Also, agent technologies nicely pair with managing utility networks such as energy distribution, mobile operators, and cable providers. We might use distributed autonomous computation for simulation and predication of alarm situations, prevention of blackouts and overload, and intrusion detection.

Available systems

Classical planning systems (using scheduling algorithms with various heuristics, constraint logic programming, genetic algorithms, and simulated annealing[2,3]) work centrally and allocate resources usually in one run for every product order in the system. These methods use mostly stochastic algorithms and generate near-optimal solutions to minimize the defined criteria (for example, sum of weighted tardiness and inventory costs). Such solutions are fully sufficient for planning in stable environments. However, in an environment with requirements to continually revise the plan, these approaches would breach the *calculative rationality requirement* (the minimal time required for two relevant changes in the environment is larger than the maximal time needed to process the change). When replanning is required, the plan is usually completely rebuilt and the algorithms' random aspect can cause major, unwanted changes, which makes this approach unsuitable for many manufacturing areas. For physically distributed production units, it's advantageous to break down and distribute the planning problem.[4,5]

Multiagent technology can address a wide range of manufacturing decision-making support problems, but few MAS implementations cover more than a single type of a problem. Solutions exist for low-level scheduling or control systems as well as product-configuration and quotation phases for short- and long-term production planning and supply chain management.[6–8]

PROSA (products-resource-order-staff architecture) is a reference architecture for manufacturing control.[9] It's mainly oriented to interholon architecture and identifies kinds of *holons* (agents)—their responsibilities, functionality, structure, and interaction protocols. PROSA defines three main classes of holons: *product, resource,* and *order*. Product agents manage production procedures and process techniques—for instance, which operations to perform to achieve the product. Resource agents represent resources such as machines. Order agents represent manufacturing orders and are responsible for following deadlines. The PROSA architecture's authors also designed *staff* agents to give the previous basic agents sophisticated knowledge support.

The Gerstner Laboratory's ProPlanT (*pro*duction *plan*ning *t*echnology) is a hierarchical technology; it comprises several basic agent classes that let you model the enterprise structure by level (which depends on the chosen granularity). From this point of view, PROSA's basic structure seems rather flat. ProPlanT's architecture seems quite static and consists of a relatively reasonable number of agents. Contrary to PROSA's architecture, the number of products, tasks, or orders doesn't affect the size of the ProPlanT agent community. On the other hand, exchanged messages in ProPlanT are more complex compared to messages in PROSA. The designers of the PROSA architecture admit that using the basic structure as the reference architecture has serious drawbacks: Because of the scalability problem, a community could end up with numerous agents (according to the size of the factory and number of products). It could provide unpredictable behavior. Additionally, PROSA's optimization has not been adequately addressed.

These drawbacks led PROSA designers to aggregate related holons to create the staff holon, a bigger holon with its own identity. Staff holons empower basic holons but play only an advisory role to avoid conflicts with the system's hierarchical rigidity. We can say that holons in PROSA are loosely coupled while in ProPlanT-like architectures a strong link exists among agents on different hierarchical levels.

More recent solutions, such as the Micro-Boss system (created at the Robotics Institute at Carnegie Mellon University), let scheduling systems constantly revise their scheduling strategies while constructing or repairing a schedule. Micro-Boss can meet new demands on the planning system and uses deterministic algorithms. Unfortunately, despite many decentralized aspects, it's a centralized approach.

The Robotics Institute also successfully integrated Micro-Boss into the MASCOT (*m*ulti *a*gent *s*upply chain *co*ordination *t*ool) system. They developed MASCOT for dynamic supply chain creation and coordination, and it doesn't affect the intra-enterprise level. It solves the problem of dynamic reconfiguration and supply chain creation and adequately covers demands on integration and enterprise cooperation. At the University of Calgary, they developed MetaMorph[10] as a generic multiagent architecture. It tries to cover every phase of the manufacturing process using mediator-centric federation architecture but provides only a simple research prototype.

Additionally, many other initiatives have investigated agent-based organization of manufacturing processes. The MADEFAST project demonstrated collaborative engineering possibilities.[11] The AARIA system integrates manufacturing capabilities (for example people, machines, and parts) in a MAS so that each agent interoperates with other agents in and outside its own factory.[12] AARIA uses a mixture of heuristic scheduling techniques: forward and backward, simulation and intelligent scheduling. A proposed *production reservation* approach (an alternative planning strategy, in which tasks are scheduled in the order they arrive at the system) has adopted the classical contract-net bidding technology. The IMS (Intelligent Manufacturing Systems) consortium has previously studied the role of multiagent systems and holons in manufacturing.[13] Table A shows the state of development and properties of selected systems and architectures.

References

1. V. Mařík and D. McFarlane, "Industrial Adoption of the Agent-based Technologies," to be published in *IEEE Intelligent Systems,* 2005.
2. N. Sadeh and M.S. Fox., "Variable and Value Ordering Heuristics for the Job Shop Constraint Satisfaction Problem," *Artificial Intelligence,* vol. 86, no. 1, 1996, pp. 1–41.
3. J. Kubalik, L.J.M. Rothkrantz, and J. Lazansky, "Genetic Algorithm with Limited Convergence," *Information Processing with Evolutionary Algorithms: From Industrial Applications to Academic Speculations*, M. Grana et al., eds., JCIS/Assoc. for Intelligent Machinery, 2005, pp. 216–235.
4. K.S. Decker and V.R. Lesser, "Generalizing the Partial Global Planning Algorithm," *Int'l J. Intelligent Cooperative Information Systems*, June 1992, pp. 319–346.
5. E.H. Durfee, "Distributed Problem Solving and Planning," Multiagent Systems: *A Modern Introduction to Distributed Artificial Intelligence*, MIT Press, 1999.
6. J.M. Swaminathan, S.F. Smith, and N.M. Sadeh, "Modeling Supply Chain Dynamics: A Multiagent Approach," *Decision Sciences*, vol. 29, no. 3, 1998, pp. 607–632.
7. P. Bečvář et al., "ExPlanTech/ExtraPLANT: Production Planning and Supply-Chain Management MultiAgent Solution," *EXP: In Search of Innovation*, vol. 3, no. 3, 2003, pp. 116–125.
8. N.M. Sadeh et al., "MASCOT: An Agent-Based Architecture for Dynamic Supply Chain Creation and Coordination," *Production Planning & Control*, vol. 12, no. 3, 2001, pp. 212–223
9. H. Van Brussel et al., "Reference Architecture for Holonic Manufacturing Systems: PROSA," *Computers in Industry*, vol. 37, no. 1, 1998, pp. 255–274.
10. F.P. Maturana, W. Shen, and D.H. Norrie, "MetaMorph: An Adaptive Agent-Based Architecture for Intelligent Manufacturing," *Int. J. Production Research*, vol. 37, no. 10, 1999, pp. 2159–2173.
11. M. Cutkosky, J. Tenenbaum, J. Glicksman, "Madefast: Collaborative Engineering over the Internet," *Comm. ACM*, vol. 39, no. 9, 1996, pp. 78–87.
12. H. Parunak, A. Baker, and S. Clark, "The AARIA Agent Architecture: An Example of Requirements-Driven Agent-Based System Design," *Proc. 1st Int'l Conf. Autonomous Agents*, ACM Press, 1997, pp. 482–483.
13. S. Bussmann, M. Jennings, and M. Wooldridge, *Multiagent Systems for Manufacturing Control: A Design Methodology*, Springer-Verlag, 2004.

Table A. Selected systems and architectures' state of development and properties.

System	Domain	Open	Lightweight	Standard	Transparency	Simulation	Development
Centralized	Planning, scheduling	No	No	No	No	No	Commercial systems
PROSA	Planning, scheduling	—	—	No	Yes	Yes	Architecture
ProPlanT	Planning, scheduling	Yes	Yes	No	Yes	No	Prototype
MASCOT	Supply chain management	Yes	—	No	Yes	No	Pilot
MetaMorph	Planning and control	—	No	KQML (Knowledge Query and Manipulation Language)	Yes	Yes	Pilot
AARIA	Control, scheduling	No	No	No	Yes	No	Architecture

Table 1. ExPlanTech components' state of development and properties.

Domain	Development	Open	Lightweight	Low cost	Standard	Transparency
Control	No	—	—	—	—	—
Planning, scheduling	System	Yes	No	Yes	Yes	Yes
Supply chain management	System	Yes	Yes	Yes	Yes	Yes
Simulation	Prototype	Yes	No	Yes	No	Yes

- *Lightweight and low cost.* Isn't computationally demanding, can run on various hardware, and optimizes reuse of the existing computational infrastructure.
- *Standard.* Uses standard interfaces to integrate new modules and functionalities (possible at runtime) with the existing infrastructure.
- *Transparent and tractable.* Provides transparent and tractable decisions.
- *Simulation and integration.* Links to both physical production machinery and simulation and planning algorithms using the same mechanisms.
- *Replanning and reconfiguration.* Facilitates local, efficient replanning and minimizes required manufacturing reconfiguration due to physical malfunctions.

Our multiagent technology answers the listed challenges to different extents. See the sidebar and Table A for a brief analysis of the available multiagent manufacturing systems and an explanation of how they fit the requirements. Table 1 illustrates the ExPlanTech multiagent system's relevant properties.

Architecture

The ExPlanTech framework adopts the ProPlanT multiagent architecture. It contains an approximately fixed number of nontrivial agents, each providing different system functionality—for example, planning, simulation, and user access. We built ExPlanTech on top of the Java Agent Development Environment (JADE, http://jade.cselt.it).

It was easy to use JADE, which allows rapid development of sophisticated and reliable *multiagent systems*. A predefined agent core with already-implemented control and message transport protocols frees the author of MASs from low-level programming and resource management. The designer can focus on high-level functions and easily build user-targeted application. Any application built on the JADE platform complies with FIPA interoperability standards (www.fipa.org) for implementing independent software agents. This feature facilitates easy integration of new and third-party components. We implemented JADE (and thus the whole ExPlanTech system) in Java 2, which gives it platform independence and openness. Agents can run on different platforms (MS Windows, Windows CE, Linux, and even programma-

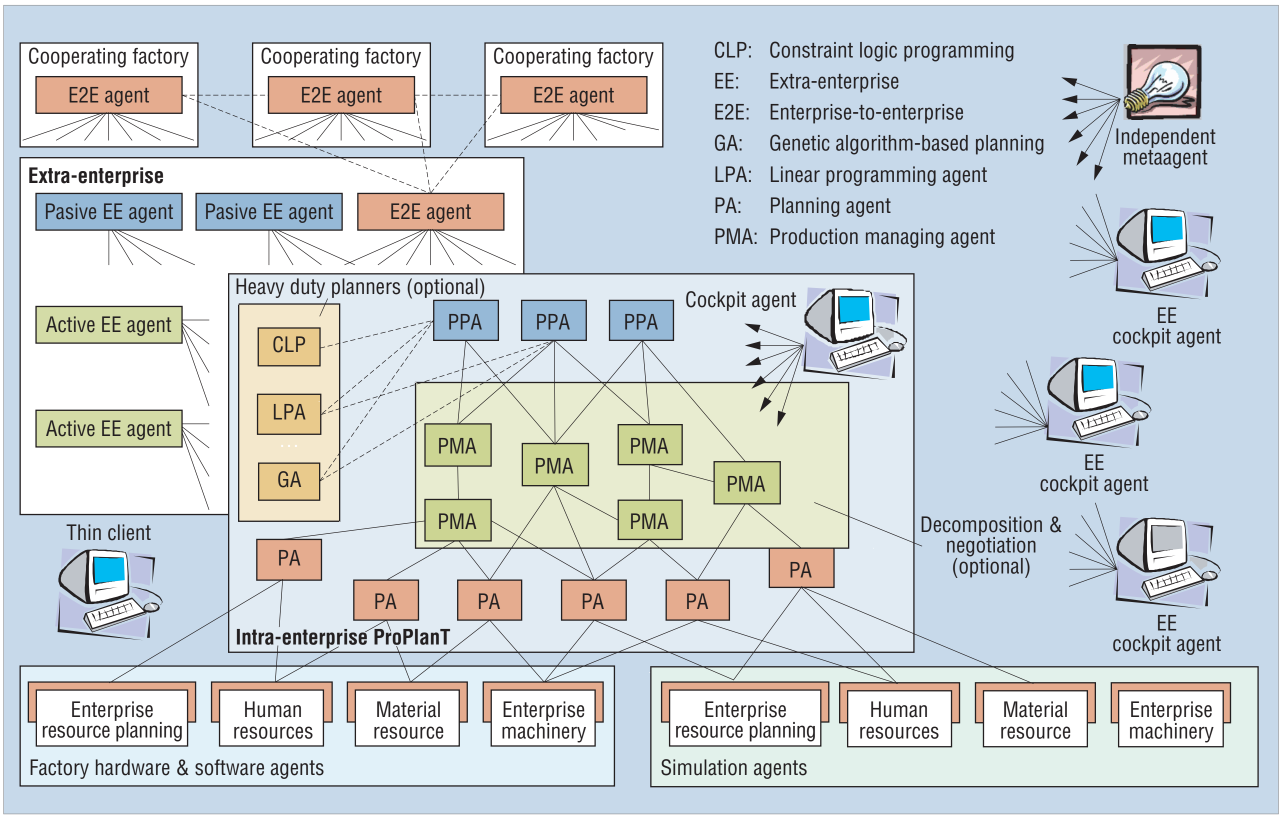


Figure 1. The ExPlanTech architecture.

ble logic controllers) and cooperate without worrying about low-level, platform-specific problems. We've developed an appropriate ontology for semantic interoperability in the manufacturing domain in ExPlanTech.

Planning agents

The core of the any ExPlanTech-based system is a community of appropriate planning agents (see Figure 1). A planning agent makes production plans for individual orders, taking care of conflicts and managing replanning and plan reconfiguration. Planning is implemented by the production planning agent, which primarily focuses on product configuration and quotation using one or a community of PMAs (production managing agents). PMAs plan production by task decomposition and partial-order planning. Additionally, you can develop various existing AI planning engines to handle different types of production—for example, linear programming, constraint logic programming, or genetic algorithm-based planning.

Resource agents

Typically, many resource agents running in the system directly interact with a planning agent and carry out data gathering and specific data preprocessing. ExPlanTech features two types of agents for integrating or representing manufacturing resources (see Figure 1). These agents

- Integrate a factory's hardware & software systems (for example, creating a bridge to a material-resources-provision (MRP) system, or integrating PLC controllers)
- Simulate a specific machine, workshop, or department (for example, a computer numeric control machine or a computer-aided design department).

Cockpit agents

Several different users could want to interact simultaneously with the planning agent. To allow this and control possible conflicts, we developed the cockpit agent (see Figure 1). Cockpit agents offer a user-friendly way to view the state of production processes, plans, given resource loads, and so on. Cockpit agents also let users interact with the system and, according to access rights, change plans or resource parameters (see Figure 2).

Extra-enterprise and enterprise-to-enterprise agents

Although we designed cockpit agents for

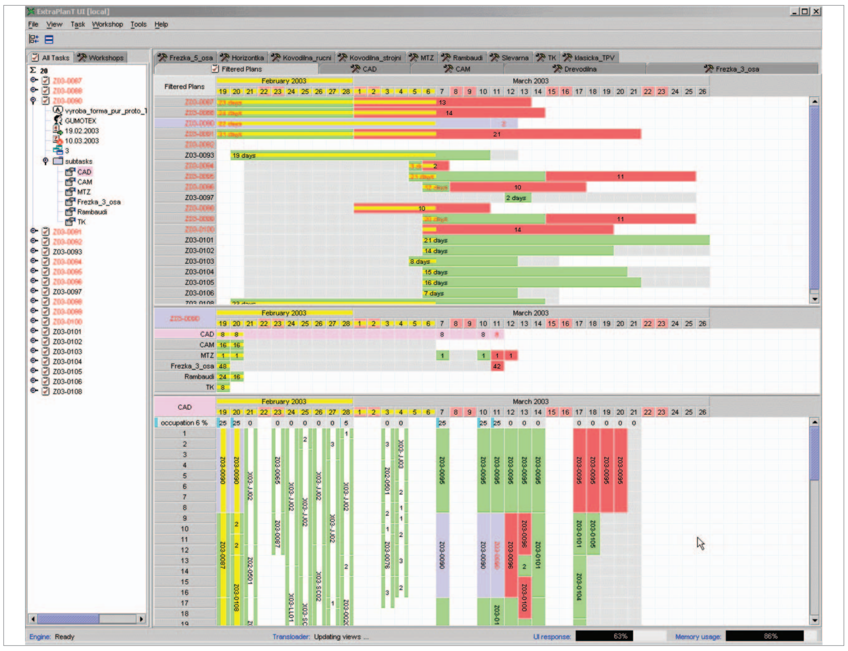


Figure 2. The cockpit agent-based graphical user interface.

use inside the factory (and its security firewalls), extra-enterprise agents let authorized users access the system from outside using a thin-client technology (see Figure 1). An EE agent has made the ExPlanTech system accessible (through a secure connection protocol) via a Web browser, PDA device, or WAP (Wireless Application Protocol)-enabled phone to remote users. An enterprise-to-enterprise agent makes the system accessible to the external software systems, such as remote cockpit agents or E2E agents at cooperating factories or material resources suppliers.

Metaagent

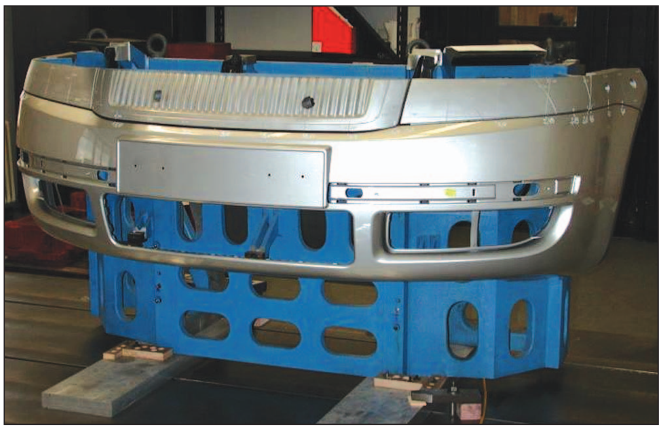
We deployed the metaagent at the intra-enterprise and extra-enterprise levels. It carries out sophisticated methods of metareasoning to independently monitor information flow among the agents and to suggest possible operation improvements (such as workflow bottlenecks, inefficient or unused production components, and long-term performance measurements.)

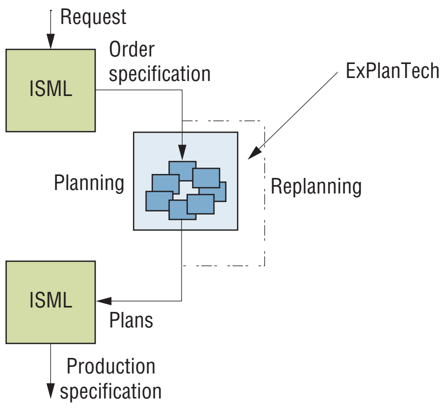
Agent coordination and negotiation

ExPlanTech's key concept is the agentification of existing and new software components. Our system has two levels of software integration: *interaction* and *social*. Interaction integration builds the interaction wrapper (provided by JADE's special class `agent`) that acts as an interface between the agent's body and other agents. Interaction integration also translates messages between the FIPA ACL (Agent Communication Language) and the agent's internal language that invokes its behavior.

More interesting, however, is ExPlanTech's social integration. To efficiently collaborate, the agents need to collect knowledge and data about the other agents with which they may collaborate—we refer to these sets of agents as an agent's *monitoring neighborhoods.*[1] This type of knowledge, often referred to as *social knowledge*, is located in the agents' *acquaintance models*. We developed different acquaintance models for each type of agent. The cockpit agent, which only visualizes the information provided by the planning agents, doesn't need a rich acquaintance model, while the planning agents need sophisticated acquaintance models containing rich social knowledge to provide efficient distributed plans in a timely way.

Distributed planning aims mainly to divide the task into several relevant subtasks (often selecting one of many options) and then subcontract these subtasks to collaborating agents. This is a very complex activity that can't necessarily guarantee a global optimum. Without a precompiled social knowledge, a planning agent must initiate a *contract-net-protocol* (CNP) for every admissible decomposition. In complex situations this is almost impossible. Social knowledge stored in the acquaintance models circumscribes the space of possible decompositions and contracts.





(a) (b)

Figure 3. Example of (a) the Modelárna Liaz product and (b) ExPlanTech's role in the production-planning process.

In practical applications, the concept of the *tri-base acquaintance model* developed for project-driven production planning proved to be useful. This model collects social knowledge in three separate bases:

- *Cooperator-base.* Contains static information (a *white page list* comprising physical information about locations, IP address, and ACL encoding and a *yellow page list* with information about provided services) about the other agents.
- *State-base.* Administers nonpermanent information about the agents in the monitoring neighborhood (operational load, implementation state of various tasks, and trust in competing communities).
- *Task-base.* Contains a list of all currently planned tasks and the *decomposition templates* for new task decomposition if a specific requirement arrives.

When planning a task, the planning agent selects the most optimal task decomposition template in the task base and instantiates the template with the cheapest (or fastest) cooperators in the state base. It further contracts parts of the constructed plan to the cooperators, using the cooperator base's information.

Maintaining the social knowledge represent the principal bottleneck of deploying acquaintance models in real applications. The more useful the acquaintance model the more its data must be kept up-to-date. This becomes very costly in complex agent systems. Several different approaches exist to maintenance, ranging from *periodical revisions*, in which the planning agent periodically asks cooperators about the information's validity in the state base, to *subscription-based interaction*, in which the planning agent subscribes to cooperators for the relevant data.

In *decomposition-based planning*, a permanent or semipermanent hierarchy of agents exists, in which each agent decomposes a task into subtasks and coordinates its completion. By contrast, *fully autonomous planning* relies on agents working together on the same planning problem. They form their local plans, which are later merged (for example, by negotiation and voting), and replanning (for example, Partial Global Planning[2]) resolves conflicts. With *backward-chaining planning*—a compromise between decomposition-based planning and fully autonomous planning—the request backpropagates through the manufacturing flow. This doesn't have a command-and-control hierarchy or a central component, but the agents autonomously push requests for their prerequisites

In ExPlanTech, we used decomposition-based planning mainly for production planning and supply chain management. We used backward-chaining planning primarily for simulation purposes. We haven't widely adopted the concept of fully autonomous planning.

In both the fully cooperative and self-interested (that is, competing) agent communities, their negotiation methods let agents reach an agreement. Although these methods focus primarily on supply-chain management and virtual enterprise organization, we've used negotiation based on a classical CNP also in the intra-enterprise environment. Negotiation is also used for autonomous replanning, especially in domains in which the planning specification changes frequently.

Possible use cases

These use cases represent the most usual ways to use an ExPlanTech-based system.

Production planning, dynamic replanning

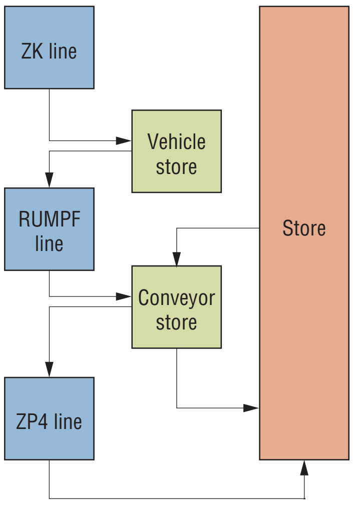
The most obvious use case is intra-enterprise production planning. ExPlanTech provides sets of linear and nonlinear plans and schedules in-house manufacturing activities so that the requested orders and tasks are achieved while optimizing enterprise resources. Given fixed deadlines, the system gives the user resource requirements and an appropriate manufacturing schedule. If the available resources are insufficient to meet the deadline, the system notifies users and initiates a supervised process of replanning and rescheduling. Replanning in ExPlanTech often occurs when the planning problem dynamically changes (for example, if the manufacturing machinery malfunctions). So, replanning solves existing or potential conflicts in production plans. ExPlanTech provides sophisticated tracking of interdependence among particular tasks, which makes replanning very fast and avoids planning again from scratch.

ExPlanTech continually analyses production data to give feedback to the project planner and keep plans up to date. Users can change task specifications and resources capacities at any time, and it recomputes and displays new plans in real time. *Variant planning* lets users examine several possible orders, test their feasibility, and choose the best one.

Supply chain management

Solving the complicated task of automated supply chain management requires overcom-

(a)



(b)

Figure 4. Example of the SkodaAUTO (a) product and (b) motor manufacturing process.

ing many technical and commercial difficulties. Unlike in intra-enterprise planning, ExPlanTech lacks complete knowledge about a supplier's parameters and capacities. This is why the simplest interaction approach (master-slave) doesn't suffice and ExPlanTech offers classical auctioning techniques (such as CNPs). Additionally, ExPlanTech handles secure and authenticated communication through the X-Security component and uses acquaintance models to handle temporary supplier inaccessibility. For supply chain integration and management, ExPlanTech provides EE agents, E2E agents, and MRP agents.

Simulation

Simulation can support decision-making in two ways. First, users can simulate a new factory or an overhauled or upgraded existing factory. The simulation tool supports a high-fidelity analysis of what an alternative's performance will be. Second, simulation can aid decision support in factory control to test how key machines' performance changes would affect the manufacturing process. ExPlanTech's integrated simulation environment lets users simulate different manufacturing scenarios to make technology changes and control safer.

EE access

Users can implement EE access either by a thin-client technology that requires an appropriate browser on the client side or by a thick client technology that assumes installation of software based on Java and JADE technology on a user's computer. Remote users (according to access rights) can exploit the functionality, which ranges from a passive observation of the system to active interventions (for example, planning custom orders).

ExPlanTech deployed

We've collaborated with several industrial partners to deploy ExPlanTech. They didn't all use an identical collection of the software system, so we customized solutions for each.

Modelárna Liaz

Modelárna Liaz is a mid sized pattern shop enterprise in the Czech Republic. The enterprise's customers are mainly from the automotive industry in the Czech Republic, Germany, and Belgium. The pattern shop specializes in single-part production of pattern equipment; permanent molds and dies; measuring and gauging devices; and welding, cooling, positioning, and machining fixtures and cubings (see Figure 3a).

The enterprise adapted ExPlanTech on the planning level. It aimed to improve medium- and long-term horizon efficiency. The important criterion was the load of strategic departments (machines) and delivery times. It implemented multiagent decomposition-based planning within the ExPlanTech. ExPlanTech agentified the factory information system and updated resource agents with real-time production feedback. One planning agent is responsible for the whole planning course. ExPlanTech implemented several cockpit agents for parallel connection to the system.

Once the enterprise resource planning system (denoted in Figure 3b as ISML, Information System in Modelárna Liaz) receives the order specification, ExPlanTech produces a complete set of production plans that are reshipped to the ERP system. Planning a new order or a change in the factory shop floor (represented by the resource agents) triggers a replanning process of all precommitted plans in ExPlanTech.

Besides production planning, ExPlanTech supports factory management with EE access to the planning data and automation of its supply chain management. The complete solution helps to find more efficient intra-enterprise plans and improve EE activities. The faster and more precise cooperation with suppliers and selling free capacities can shorten the production lead-time and create higher use of the factory. After several months of testing, the system proved its potential by improving machine use by 30 percent and reduced the finished product's due time by 5.3 percent.

SkodaAUTO

The SkodaAUTO motor factory, in collaboration with Gedas and CertiCon software companies, has successfully applied the ExPlanTech technology to design a robust planning system for car engines manufacturing. This exemplifies high-volume production, in which a few thousand engines (see Figure 4a) are manufactured daily. A high variability exists in the types of motors to be manufactured. The planning system needed to provide us with hourly plans for a period of six weeks. The production process (see Figure 4b) involves three production lines (ZK, Rumpf, ZP4) and two different parts buffer stores (vehicle and conveyor) and the main store for the finished products.

The agent technology provided a great help in solving the highly complex problem of planning assembly line production. We designed planning to occur on two independent levels:

- On a higher level, ExPlanTech produces a rough plan. This plan specifies an approximate amount of engines to be produced each day so that all the requested constraints are met. We have used a linear programming based heavy-duty agent for elaborating this higher level plan.
- On a lower level, the agents (each representing a line or a buffer store) analyze the provided higher-level plan and check for conflicts. In an ideal situation, the amount of conflicts is reasonable so that agents can negotiate and solve conflicts by swapping the tasks within days. The lower-level planning algorithm primarily performs conflict resolution by negotiation. The lower-level planning also provides daily ordering of the tasks.

Behr

The use case for Behr, an automotive supplier in the field of cooling and air-conditioning systems, employs mainly ExPlanTech's production simulation, supported by simplified planning and special cockpit agents and metaagents. The simulation aims primarily to compare the long-term effectiveness of several shop floor layouts. The simulation also lets Behr find production bottlenecks and optimal product buffer positions and evaluate the impact of important machine failures. These results are very important in decision support during the design of new or reconfiguration of an existing factory or even during important control decisions. Behr carried out adoption of ExPlanTech within the MPA (Modular Plant Architecture) project funded in part by the European Commision.

ExPlanTech's—and that of agent-based technologies in general—main virtue is in its high level of integration openness. When implementing an ExPlanTech-based, production-planning system, it's possible to reuse most of the previously existing IT infrastructure and software equipment as well as to integrate new decision-making support modules. At the same time, users have noted the system's high level of decision-making transparency and ability to involve human experts in the planning process.

In any of the listed deployments, the agent-based solution doesn't guarantee to provide optimal solutions with fewer computational resources (such as time and memory) than classical AI planning systems. This is one of the most common disappointments that agent-based technology adopters experience. However, the multiagent paradigm's ability to combine distributed AI algorithms with heavy-duty problem solvers and heuristic knowledge of the planning problem provides sophisticated solutions in specific cases.

JADE-based systems such as ExPlanTech require substantial amounts of computational resources to run promptly. As we have always developed a limited number of agents (up to 30), with any of the deployed applications, this hasn't been a problem. Scalability of the JADE-based software system has proven to be a bottleneck (especially in situations where several agent-based systems are working with the same hardware or software resources). A server-based version of the ExPlanTech system was developed to overcome this obstacle. Users access the system via a community of lightweight cockpit agents or from Web browsers by means of EE agents.

Currently, at Gerstner Laboratory, we're investigating in the concept of lightweight agent platforms that would allow massive scalability in the number of agents. We developed a novel lightweight platform—A-globe (http://agents.felk.cvut.cz/aglobe)—which recently received the System Innovation Award at a prestigious Cooperative Information Agent (CIA) workshop at the NetObject Days event in Erfurt. ■

Acknowledgment

We prepared this article under the support of the European Union Networking's Innovative Production Machines and System project and the EU Integrated Project, European Collaborative Networked Organisations Leadership (ECOLEAD).

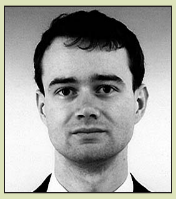
References

1. M. Pěchouček, V. Mařík, and O. Štěpánková, "Role of Acquaintance Models in Agent-Based Production Planning System," *Cooperative Information Agents* IV, vol. 1, Springer-Verlag, 2000, p. 179–190.
2. K.S. Decker and V.R. Lesser, "Generalizing the Partial Global Planning Algorithm," *Int'l J. Intelligent Cooperative Information Systems,* June 1992, pp. 319–346.

The Authors

Michal Pěchouček is the head of the Agent Technology Group at Gerstner Laboratory, a senior lecturer in artificial intelligence at Czech Technical University, and a part-time senior consultant for CertiCon. His research interests include agent-based computing, industrial deployment of agents, and social reasoning. He's the industry action coordinator for the AgentLink management committee, head of the EUMAS advisory board, and cochair of several international conferences, such as the industry track of the International Joint Conference on Autonomous Agents and Multiagent Systems 2005. He is a principal investigator on several research projects with direct defense and manufacturing industry involvement. He received his PhD in AI from CTU. Contact him at Gerstner Lab., Czech Technical Univ. in Prague, Technická 2, 166 27 Prague 6, Czech Republic; pechouce@labe.felk.cvut.cz.

Jiří Vokřinek is a researcher in the Agent Technology Group at Gerstner Laboratory. His research interests include artificial intelligence, multiagent systems, manufacturing planning and replanning, and virtual organizations. He's involved researching advanced agent-based technologies for supporting virtual organizations (IST Integrated Project ECOLEAD) and agent-based production processes simulations (CONCEERN, *Con*ex *C*entral *E*uropean *E*lectronics *R*ecycling *N*etwork). He received a university degree in technical cybernetics from CTU. Contact him at Gerstner Lab., Czech Technical Univ. in Prague, Technická 2, 166 27 Prague 6, Czech Republic; vokrinek@labe.felk.cvut.cz.



Petr Bečvář is a researcher at the Centre of Applied Cybernetics and a project manager at CertiCon. His research interests include process classification, expert systems, technical diagnostics, and intelligent human-computer interfaces. He received his PhD in technical cybernetics from the University of West Bohemia. Contact him at CertiCon, Václavská 12, 120 00 Praha 2, Czech Republic.

Agents Towards Vehicle Routing Problems[*]

Jiří Vokřínek
Agent Technology Center
Czech Technical University in Prague
vokrinek@agents.felk.cvut.cz

Antonín Komenda
Agent Technology Center
Czech Technical University in Prague
komenda@agents.felk.cvut.cz

Michal Pěchouček
Agent Technology Center
Czech Technical University in Prague
pechoucek@agents.felk.cvut.cz

ABSTRACT

A multi-agent VRP solver is presented in this paper. It utilizes the contract-net protocol based allocation and several improvement strategies. It provides the solution with the quality of 81% compared to the optimal solution on 115 benchmark instances in polynomial time. The self-organizing capability of the system successfully minimizes the number of vehicles used. The presented solver architecture supports great runtime parallelization with incremental increase of solution quality. The presented solver demonstrates applicability to the VRP problem and easy adaptation to problem variants.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Vehicle routing problem, heuristic, multi-agent solver, benchmarks

1. INTRODUCTION

The Vehicle Routing Problem (VRP) is a well-known optimization problem introduced in [3]. The problem is defined as routing of a fleet of gasoline delivery trucks between a terminal and a number of service stations. The trucks have load capacity limitations and deliveries have to be accomplished at minimum total cost (distance traveled).

This paper presents an agent-based solver producing feasible solution of the VRP instance in a polynomial time that doesn't use exhaustive searches or randomizing methods.

[*]Presented effort was sponsored by Czech Ministry of Education grant 6840770038 and by CERDEC, U.S. Army contract FSD BAA 8020902.A.

Cite as: Agents Towards Vehicle Routing Problems, Jiří Vokřínek, Antonín Komenda and Michal Pěchouček, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 773-780

1.1 Problem Statement

The VRP falls into the category of the NP-hard problems and so it is difficult to solve it in reasonable time. It is based on interdigitation of two underlaying problems that are also NP-hard – the Multiple Traveling Salesman Problem (MTSP) and Bin Packing Problem (BPP). A feasible solution to the VRP is a solution of MTSP that satisfies the capacity constraints (decision variant of BPP). By relaxation of one of the underlying problems (MTSP, BPP respectively), we can transform VRP into the other subproblem (BPP, MTSP respectively).

The Vehicle Routing Problem can be formalized as:

Definition 1. Let us have a set of cities $c_1, \ldots, c_k$ with known mutual distances and positive demands $d_1, \ldots, d_k$, the Vehicle Routing The problem is to find a set of m tours that together visit all nodes, each node is visited exactly once and by only one tour, the sum of the tours is minimal and the sum of the node demands served by each tour doesn't exceed the vehicle capacity C.

We define the set of tasks $T = \{n_1, ..., n_k\}$, where n_i is a doublet of a city and the corresponding demand:

$$n_i = (c_i, d_i), \tag{1}$$

$$\forall i \forall j : c_i \neq c_j \textbf{ iff } i \neq j.$$

The vehicle capacity constrain is defined for each route as:

$$\sum_{i=1}^{l} d_i \leq C, \tag{2}$$

where l is a number of tasks in the route and d_i is the demand of i^{th} task of the route. For ensuring feasibility of the solution we require:

$$\forall i : d_i \leq C. \tag{3}$$

It is obvious that for $d_i > C$, there is no way to handle this demand by a single truck and thus no solution for the problem exists.

In practical applications, the VRP is defined either with a fixed number of vehicles or as a problem with a minimal number of vehicles demanded. The determination of the minimal number of vehicles is a decision variant of BPP and is related also to determining the minimal number of routes for MTSP. Even though this problem is NP-hard, we can easily define the lower and upper bound of the number of vehicles (routes):

THEOREM 1. *The number of vehicles m in the feasible solution of the Vehicle Routing Problem is bounded by*

$$\frac{\sum_{i=1}^{k} d_i}{C} \leq m \leq k$$

where k is the number of cities, C is the capacity of the vehicle, and d_i is a demand of the i^{th} city.

PROOF. The lower bound of the number of vehicles respects the capacity constraints stated in Definition 1. For m vehicles the cumulative capacity of the whole fleet is $m \times C$ and the cumulative size of the demand is

$$\sum_{j=1}^{m} \sum_{i=1}^{l} d'_{i,j} = \sum_{i=1}^{k} d_i,$$

where $d'_{i,j}$ is a demand of the i^{th} city on the j^{th} route. Respecting Equation 2 the cumulative size of the demands in the feasible solution cannot exceed the cumulative capacity of the fleet. The upper bound is given by Equation 3 in the case where every demand is served by one route, i.e. $l = 1$ and $m = k$. $\square$

1.2 Existing Solution Techniques

In the original Dantzig's article [3] the problem is formulated as a linear program providing a near-optimal solution. Classical solution techniques include wide variety of exact methods (branch and bound, branch and cut, set covering, spanning tree, shortest path relaxation, etc.), heuristics (constructive, two-phase, improvement, etc.), and metaheuristics (simulated annealing, tabu search, genetic algorithms, ant algorithms, neural networks, etc.). More information about solution techniques can be found for example in [6], [7], or [9]. This paper focuses on k-VRP (k is the number of cities), which is proven to be NP-hard and the best known approximation of the k-VRP is $\frac{5}{2} - \frac{3}{2n}$ for the metric case (triangle inequality is satisfied) [5].

The Agent-based approach to a variant of the VRP solver has been presented for example in [10]. Authors use three types of agents – Client, Bidder and Vehicle agents. The approach is based on the contract-net protocol (CNP) allocation and optimization based on exchange of tasks between the Vehicle Agents. The Vehicle Agents use an insertion heuristic and improvement strategy for task swapping between them. The error of the solution (compared to the optimal solution) presented in the paper is 4–29%.

A similar approach has been used in [1] for a dynamic variant of k-VRP (new tasks are added during the execution), where the initial allocation is generated using a centralized algorithm. The dynamic task allocation is made by the CNP protocol. Then two improvement phases are applied. The *intra-route optimization* is applied to each agent route and *inter-route optimization* is performed between Vehicle Agents (1 or 2 random tasks are moved between agents). The optimization is performed continuously during vehicle rides until all tasks have been fulfilled or a new dynamic task comes (1 hour interval in the experiments). The error of solution on all static requests has been reached 0–8%, but it is not described how much computation time has been used for static instances and also there is no discussion of stability and the speed of convergence of the solution quality (the optimization algorithm terminating condition is not defined).

In both [10] and [1] there is no discussion of the number of Vehicle Agents and handling of the potential allocation failure because of capacity constrains (no Vehicle Agent is capable of undertaking the next task) which can arise when the number of Vehicle Agents is lower than the upper bound defined by Theorem 1.

2. MULTI-AGENT SOLVER

The multi-agent planning approaches are used for solving a wide variety of planning problems. As analyzed in [2] the multi-agent planning techniques can be beneficial for such problems where the domain sizes of individual agents are considerably smaller (e.g. in logarithmic relation) than the overall size of the problem (even if the planning complexity of individual agent is exponential) and the number of dependencies between agents is low. Although the Vehicle Routing Problem consists of several NP-hard problems which may not satisfy the presented conditions (e.g. the BPP part produces an agent domain comparable to the overall problem size for the agent maintaining task distribution), this paper shows the applicability of the agent-based approach and discusses it's benefits and limitations.

2.1 Architecture

In this section we introduce the polynomial agent-based Vehicle Routing Problem solver producing a feasible solution, according to Definition 1, minimizing the routes' cost and the number of vehicles used. The solver is composed of three types of agents (see Figure 1). They are

- **Task Agent** for processing of demands and allocation invocation,
- **Allocation Agent** for maintaining allocation and the improvement process, and
- **Vehicle Agent** for route planning and optimization.

The multi-agent solver is composed of one Task Agent, one Planning Agent and a set V of Vehicle Agents (each agent utilizes one of the strategies described below). The number of Vehicle Agents respecting Theorem 1 is constrained by the lower and upper bound.

Since the agent-based solver doesn't use the exact method for solving the BPP part of the problem it may be impossible to find a solution using a minimal number of vehicles. In such a case, the whole process has to be restarted with an increased number of Vehicle Agents or an agent has to be added during the solving process. On the other hand, the upper bound of the number of Vehicle Agents guarantees the feasibility of solution but may produce worse results. Thanks to the agent paradigms, we can expect some sort of self-organizing behavior that emerges during the process of solution improvement. Let us formulate the following hypothesis:

HYPOTHESIS 1. *The implicit self-organizing ability of the agents emerges to the solution that corresponds to the minimal number of vehicles. The solutions produced by the system converge to the same number of vehicles regardless of the initial number of Vehicle Agents (provided that the solution is found).*

To solve the given VRP instance, we generate a set of solvers utilizing a different combination of strategies and

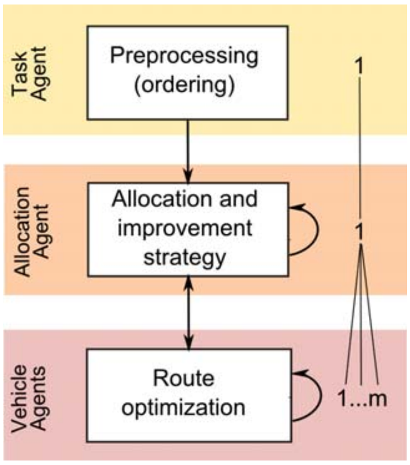


Figure 1: Generic architecture of agent-based VRP solver.

processing an input instance in parallel. The solvers produce solutions of the given problem independently and then the solutions are aggregated by the *Composite Solver* which selects the best one that minimizes the cost of the solution:

$$cost = \min_{S} cost(S), \tag{4}$$

where S is a solution provided by a solver and $cost(S)$ is the cost of the solution (see Equations 5 and 8). From the strategies described in the next subsections, we have built 42 solvers competing to answer the fundamental VRP question.

2.2 Task Agent

The Task Agent is responsible for collecting the demands and passing them to the Allocation Agent. It optionally applies the ordering preprocessing to the set of incoming tasks. It is able to pass the tasks to the Allocation Agent one by one (this represents the dynamic variant of VRP) or in a batch.

We suppose the ordering of the tasks directly influences the BPP part of the VRP. In the case of wrong ordering, the VRP solution may not be feasible for the lower bound of the number of Vehicle Agents because of capacity constraints (Equation 2). In the case of correct ordering, there may be a bigger chance to optimally allocate the tasks to the Vehicle Agents. These assumptions lead us to formulate the following hypothesis:

Hypothesis 2. *Applying appropriate ordering to a set of tasks, the chance of finding the solution of VRP with a minimal number of vehicles is increased and the cost of the solution is decreased.*

Since the number of possible orderings increases rapidly with the size of the task set, we are not able to investigate all the orderings (in fact, the complexity of such an exhaustive search is comparable to solving the VRP itself). The First Fit Decreasing (FFD) heuristics is a classical BPP algorithm that has been recently proved [4] to provide the tight bound of $\frac{11}{9}OPT + \frac{6}{9}$ (e.g. when the instance of BPP can be optimally solved with 6 bins, the FDD is able to solve it with the maximum of 8 bins). Since it is a good algorithm for BPP, we decided to build the strategy of Task Agent inspired by this algorithm. The Task Agent strategy doesn't directly affect the allocation of the tasks to the Vehicle Agents, but it may have strong influence on the Allocation Agent strategy efficiency because of ordering capability. To inspect the influence of the Task Agent strategy, we have created also an opposite strategy to FFD for comparison.

The Task Agent uses the following three ordering strategies:

1. **Most Demand First** (MDF) based on the FFD, where the tasks are ordered with decreasing demands,
2. **Least Demand First** (LDF), which is opposite to MDF, so the tasks are ordered with increasing demands,
3. **First In First Out** (FIFO), where tasks are not ordered and their sequence corresponds to the time of arrival.

The time complexity of the Task Agent preprocessing (MDF and LDF strategies) corresponds to the complexity of standard sorting algorithms, which is $O(nlog(n))$. The complexity of the FIFO strategy is $O(1)$. Therefore the complexity of the Task Agent algorithm is $O^{TA} = O(nlog(n))$.

The task processing (passing them to the Allocation Agent) can be handled by one of two strategies:

1. **Batch processing** (NORM), where all available tasks are sent as one batch, and
2. **Iterative processing** (ITER), where task are sent one by one.

2.3 Allocation Agent

The Allocation Agent applies a defined strategy to allocate the tasks to the set of Vehicle Agents. The allocation strategy searches for the best suitable mapping of the tasks to the Vehicle Agents that minimizes the overall cost. The goal of Allocation Agent is to find such a partition $\mathcal{P}$ of the set of all tasks T that

$$\operatorname*{argmin}_{\mathcal{P}} \sum_{i=1}^{v} cost(N_i), \tag{5}$$

where v is a number of Vehicle Agents, N_i is a subset of tasks allocated to the i^{th} Vehicle Agent, $cost(N_i)$ is a cost function computed by the i^{th} Vehicle Agent (Equation 8), and $N_i \subseteq T, \bigcup_{i=1}^{v} N_i = T$ where $\forall i, j : N_i \cap N_j = \emptyset$ **iff** $i \neq j$.

Equation 5 conforms to Definition 1, where Equation 2 is covered by Vehicle Agents.

The Allocation Agent algorithm consists of two phases: (i) *allocation phase* and (ii) *improvement phase*.

The first phase builds a feasible solution and the second performs incremental improvement. Both phases are captured by Algorithm 1.

Two allocation strategies implemented by the Allocation Agent are:

1. **Contract-net based** (CNP) – is based on the well-known contract-net protocol. For every task the best Vehicle Agent is selected according to insertion estimation (see Section 2.4) satisfying capacity constrains (Equation 2). This strategy contains no backtracking and in case of allocation failure (because of capacity constraint of Vehicle Agents) the whole process is restarted with a higher number of Vehicle Agents. This strategy is described by Algorithm 2.

2. **Capacity backtracking strategy** (BC) – is based on the previous strategy, but with backtracking in case of allocation failure. In case when no Vehicle Agent can undertake a new task because of the capacity constraint, the best Vehicle Agent is selected regardless of capacity limitations. This agent removes the worst tasks until the new task fits the increased free space. After that, the removed tasks are allocated again. The reallocation counter controls the number of reallocations and when it reaches the pre-defined maximum, the number of Vehicle Agents is increased and the process is restarted. This strategy is described by Algorithm 3.

The CNP strategy can fail because of capacity constrains and can be restarted up-to $k - \frac{\sum_{i=1}^{k} d_i}{C}$ times when the number of Vehicle Agents reaches upper bound (see Theorem 1).

The BC strategy can provide feasible allocation with a lower number of Vehicle Agents because of backtracking, but also can be trapped in an infinite reallocation loop. The reallocation counter helps to recover from this loop. In the worst case, the strategy is being restarted until the number of Vehicle Agents reaches the upper-bound and the BC provides the same result as CNP.

After successful allocation, the improvement phase takes place. We have designed three improvement strategies that Allocation Agent can use. The strategies are improvement heuristics that produce the same or better solution after each run. In all cases, the strategy is repetitively executed on all vehicle Agents until the solution overall cost (see Equation 5) stops improving (see Algorithm 1).

The improvement strategies are (see Algorithm 4 for more details):

- **Delegate worst** (DW) – each Vehicle Agent identifies it's worst task and tries to delegate it to another agent if the savings are higher than the insertion cost (see Equations 10 and 9).
- **Delegate all** (DA) – each Vehicle Agent delegates all its tasks (only if the savings are higher than the insertion cost).
- **Reallocate all** (RA) – each Vehicle Agent successively removes all its tasks from the plan and allocates it again using the CNP strategy. The result of the allocation can be the same as before task removing, or a change of the position of the task in the current agent plan, or delegation to another agent.

The results produced by the two-phase algorithm are influenced by the Task Agent processing strategy. The NORM strategy allows to allocate all tasks and then perform the improvement phase. The ITER strategy provides a high degree of *dynamism* where tasks are allocated one by one and the optimization is performed after allocation of each task.

The worst-case time complexity of the Allocation Agent algorithm is

$$O^{AA} = n \times O^{alloc} + n^2 \times m \times O^{impr}, \qquad (6)$$

where n is the number of tasks, m is the number of agents, O^{alloc} is the complexity of the allocation strategy, and O^{impr} is the complexity of the improvement strategy. The complexity of individual strategies is shown in Table 1 where rc is the reallocation counter threshold (constant) and O^{estI},

Algorithm 1 The Allocation Agent main algorithm.

```
function solve($T$, $V$) begin
    forall $t : T$ begin
        run allocation strategy for task $t$
        if allocation not successful then
            restart solver with increased
              number of Vehicle Agents
        end
    end
    $improvement := true$
    repeat until $improvement$ is $false$
        $improvement := false$
        forall $v : V$ begin
            run improvement strategy for agent $v$
            if solution has been improved then
                $improvement := true$
            end
        end
    end
end
```

Algorithm 2 Contract-net based allocation strategy.

```
function allocateCNP($t$, $V$) begin
    forall $v : V$ begin
        find $winner$ with the lowest insertion
          estimation of $t$ not exceeding capacity $C$
    end
    if $winner$ is found then
        assign $t$ to the $winner$
    else
        allocation not successful
    end
end
```

Algorithm 3 Capacity backtracking allocation strategy.

```
function allocateBC($t$, $V$) begin
    allocateCNP($t$, $V$)
    if allocation not successful then
        if reallocation counter is reached then
            allocation not successful
            return
        end
        forall $v : V$ begin
            find $winner$ with the lowest insertion
              estimation of $t$ ignoring capacity constrain
        end
        while $winner$ has not enough capacity for $t$ begin
            remove the worst task of $winner$
              and put it to $REALOC$
        end
        assign $t$ to the $winner$
        forall $r : REALOC$ begin
            allocateBC($r$,$V$)
        end
    end
end
```

Algorithm 4 Improvement strategies of Allocation Agent.

```
function improveDW($v$,$V$) begin
    $t$ = the worst task of agent $v$
    forall $a : V \setminus v$ begin
        find winner with the lowest ins. estimation of $t$
    end
    if ins. cost of winner is lower then savings of $v$ then
        swap $t$ from $v$ to winner
    end
end

function improveDA($v$,$V$) begin
    forall $t$ : tasks of agent $v$ begin
        forall $a : V \setminus v$ begin
            find winner with the lowest ins. estimation of $t$
        end
        if ins. cost of winner is lower then savings of $v$ then
            swap $t$ from $v$ to winner
        end
    end
end

function improveRA($v$,$V$) begin
    forall $t$ : tasks of agent $v$ begin
        remove $t$ from agent $v$
        allocateCNP($t$,$V$)
    end
end
```

O^{estR}, O^{worst}, O^{ins}, and O^{rem} are complexities of Vehicle Agent algorithms for task insertion/removal estimation, task insertion, finding the worst task, and task removal defined in the next section.

2.4 Vehicle Agent

The Vehicle Agent represents a single truck and is responsible for optimization of the route cost through the assigned tasks. It starts and finishes the route at the depot. In fact, this routing problem corresponds to the traveling salesman problem, where new customers come successively (as the Allocation Agent progress with task allocation).

Let $N = (n_1, \dots, n_l)$ be a set of l tasks allocated to the agent, $I = (i : 1, \dots, l)$ is an ordered index set, where $I \in \mathcal{I}, |I| = l$ and $\mathcal{I}$ is a set of all index permutations.

According to Definition 1 the objective function of Vehicle Agent can be formalized as follows:

$$\operatorname*{argmin}_{I \in \mathcal{I}} d(n_d, n_{I_1}) + \sum_{j=1}^{l-1} d(n_{I_j}, n_{I_{j+1}}) + d(n_{I_l}, n_d), \quad (7)$$

where $d(n_i, n_j)$ is a distance traveled between task i and j, and n_d is the depot, ensured that Equation 2 holds.

Given the I minimizing Equation 7, the cost function of the Vehicle Agent is then

$$cost(N) = d(n_d, n_{I_1}) + \sum_{j=1}^{l-1} d(n_{I_j}, n_{I_{j+1}}) + d(n_{I_l}, n_d). \quad (8)$$

The Vehicle Agent is able to compute the cost function during interactions with the Allocation Agent in the case of addition of new tasks, removal of an already assigned task, and estimation of adding/removing a task.

O^{CNP}	$m \times O^{estI} + O^{ins}$
O^{BC}	$rc \times (O^{CNP} + m \times O^{estI} + n \times O^{rem} + O^{ins})$
O^{DW}	$O^{worst} + (m-1) \times O^{estI} + O^{ins} + O^{rem}$
O^{DA}	$n \times O^{estR} + (m-1) \times O^{estI} + O^{ins} + O^{rem}$
O^{RA}	$O^{rem} + m \times O^{CNP}$

Table 1: Complexity of Allocation Agent strategies.

The algorithm used by Vehicle Agent is based on the well-known cheapest-insertion heuristics [8]. When inserting new tasks n_{l+1} the algorithm searches (in case of satisfying Equation 2) for the best suitable index j, where

$$\operatorname*{argmin}_{j \in I'} d(n_{I'_{j-1}}, n_{l+1}) + d(n_{m+1}, n_{I'_j}) - d(n_{I'_{j-1}}, n_{I'_j}), \quad (9)$$

and $I' = (d) \cup I \cup (d)$. The inner part of the Equation 9 is the insertion cost estimation. The new task n_{l+1} is then inserted on the position $k = j - 1$ in the agent's plan, e.g.:

$$\begin{aligned} I &:= (i_1, \dots, i_{k-1}, l+1, i_k, \dots, i_l), \\ N &:= N \cup n_{l+1}, \\ m &:= l + 1. \end{aligned}$$

The same heuristics is used for identification of the worst task (invoked by the delegation strategy of Allocation Agent), so the worst task n_j is such a task where:

$$\operatorname*{argmax}_{j \in I'} d(n_{I'_{j-1}}, n_{I'_j}) + d(n_{I'_j}, n_{I'_{j+1}}) - d(n_{I'_{j-1}}, n_{I'_{j+1}}), \quad (10)$$

i.e. the savings (the right part of the equation) of removing such tasks are maximized.

In the ***dynamic*** variant of the VRP, the algorithm is simply modified by constructing $I' = (c) \cup (i_p, \dots, i_l) \cup (d)$ where n_c is the current position of the agent and n_{i_p} is the next task to be serviced. The plan is not searched from the beginning but from the current point of execution – new task cannot be inserted to the already traveled path $(n_{i_1}, \dots n_{i_{l-1}})$. This modification has minimal impact on the functionality or the source codes of the whole system. The Vehicle Agent is also able to easily cover modifications of VRP such as the multi-depot variation, heterogenous capacities, additional route constraints, etc.

The complexity of insertion heuristics for inserting the i^{th} task is $O^{ins} = O(i)$, so the overall complexity of the planning l tasks allocated to an agent is $O(\frac{l^2}{2})$. The complexity of finding the worst task is the same as planning a task and insertion estimation, thus $O^{worst} = O^{estI} = O(i)$, removing one task costs only one operation as does removal estimation, thus $O^{rem} = O^{estR} = O(1)$.

2.5 Complexity

The general computational complexity of the multi-agent solver is introduced in [2]. Using transformation of the multi-agent planning problem to the distributed constraint satisfaction problem, the worst-case time complexity of the multi-agent planning is upper-bounded by

$$f(\mathcal{I}) \times exp(comm) + exp(int), \quad (11)$$

where $f(\cdot)$ is the factor inducted by requesting each agent to plan while committing to a certain sequence of actions, $\mathcal{I}$ is the complexity of an individual agent's planning, $exp(comm)$

represents a factor exponential in min-max number of per-agent commitments, and an additive factor $exp(int)$ represents the interactions of agents.

The consequences of Equation 11 lead to interesting features of the multi-agent solver, such as (i) no direct exponential dependence on the number of agents, (ii) no direct exponential dependence on the size of the planning problem or size of the joint plan, and (iii) no direct exponential dependence on the length of individual agent plans [2].

In our case the feature (ii) does not have a strong impact on the BPP part of the problem because in the worst case the size of the problem of our agents is the same as the size of the overall problem (for the MTSP part of the problem the feature holds). On the other hand the exponential factors are reduced by the polynomial heuristics – allocation and improvement strategies of the Allocation Agent and the insertion heuristic of the Vehicle Agent. The ordering strategy of the Task Agent does not have a strong influence on the worst case complexity because of its additive nature and low complexity.

By combining complexities of the individual agent strategies described earlier, we can define the complexity of the overall solver. Because of the space limitations of the article, we are not able to describe the complexity of every combination of strategies, so we present only the results here. The upper-bound of the worst-case time complexity of the solver is (taking into account the worst-case number of restarts, reallocations and backtracking)

$$O(n^3), \tag{12}$$

where n is the number of tasks (i.e. nodes or demands). There is no influence of the number of agents, but the number of Vehicle Agents is reflected by its dependency on the number of tasks (see Theorem 1) and increase the worst-case complexity exponent by one because of restarts towards the upper-bound number of vehicles.

2.6 Experiment Baseline Solver

For the evaluation purposes we have created a baseline solver reconstructing the optimal (or the best known) solution. It is based on the known results of the benchmark instances. It is composed of the presented three types of agents with special strategies. For the Task Agent it is:

- **Optimal order**, where tasks are ordered in an optimal way. This strategy is based on the previously known optimal solution. The order of the tasks is given by merging optimal routes, where tasks are ordered by their distance from the depot to the corresponding city computed on the vehicle route.

The Allocation Agent algorithm is composed only of the following allocation strategy (no optimization strategy is needed):

- **Optimal allocation**, where tasks are allocated to the Vehicle Agents according to routes in the known optimal solution.

The Vehicle Agent optimal strategy is bounded to the optimal allocation (it is obvious that with non-optimal allocation, the Vehicle Agent strategy is not able to produce the globally optimal solution). So the optimal Vehicle Agent strategy is:

- **Optimal route**, where tasks are ordered in the same way as in the known optimal solution.

This baseline solver is used for comparison in the experiments as a whole (reconstructing the best known solution) or as a part of the standard solver for investigating influences of the individual agents.

3. EXPERIMENTS

The presented solver has been evaluated on the VRP benchmark instances from two sources. The first source is VRPLIB[1] (s̈ymetric CVRP instances̈), and the second is a compilation of instances from the COIN-OR project[2]. All the instances use the Euclidean distance for edge weights and the number of nodes varies from 16 to 484. We have used 115 instances with a known solution (optimal or best known).

The comparison of solutions has been made against solutions reconstructed by the baseline solver (see Section 2.6). We measure the computation time and quality of the solution defined as:

$$\frac{cost_{solver}}{cost_{optim}} \times 100[\%], \tag{13}$$

where $cost_{solver}$ is the solution cost provided by the solver (see Equation 4) and $cost_{optim}$ is the cost of the best known solution. All the experiments have been run on a standard laptop with 3GB of RAM and 2.5GHz dual core processor. The solver has been implemented as JAVA application with no performance optimizations.

3.1 Solution Quality

The quality of the solution has been evaluated for the composite solver (using 42 parallel solvers with combination of strategies as described in Section 2). The aggregated results of all experiments show that the best solution quality reached is 100%, the lowest quality is 81.3% and the average solution quality over all instances is 91.3%. The results per instance (the x-axis represents instances with an increasing number of nodes to the right) can be seen in Figure 2 – the dots are the best solutions of the composite solver, the vertical lines represent the span of results of other solvers (e.g. non-winning strategies combinations). The results of the experiment also indicate that there is no dependence of the solution quality on tightness of the instance. The composite solver has reached the solution quality of more than 81% on all the benchmark instances and the average quality has been 91%. The quality of 100% has been reached for 3 instances, for 28 instances we reached quality better than 95% and there have been 63 instances with solution better than 90%.

3.2 Ordering Strategy

These experiments evaluate the influence of the Task Agent ordering strategy on the quality of the solution provided by the solver. The strategies of Task Agent are compared to the baseline Optimal Order strategy. Table 2 shows the solution quality of the solver through all instances aggregated for ordering strategies (solvers are divided into the groups according to the ordering strategy used and the best solution

[1]http://www.or.deis.unibo.it page /research_pages/ORinstances/VRPLIB/VRPLIB.html
[2]http://www.coin-or.org page /SYMPHONY/branchandcut/VRP/data/

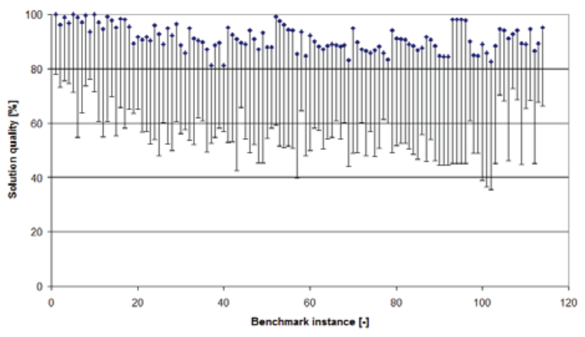


Figure 2: Solution quality of the composite solver per each benchmark instance.

Task Processing	max	min	average	deviation
MDF	100.0	75.83	87.62	6.30
LDF	99.1	68.46	87.06	7.39
FIFO	100.0	75.87	88.45	5.87
Optimal Ordering	100.0	69.8	88.84	6.39

Table 2: A comparison of the task processing methods. The values are the best results in form of solution quality.

is used for each instance and ordering). The best solution for all ordering strategies reaches 100% of solution quality and the average quality is almost the same for all evaluated ordering strategies. The worst performance provides LDF (also the biggest deviation of results). It seems to be provably worse than MDF, which looks like potential support of Hypothesis 2 and good influence of FFD heuristics for BPP. Unfortunately, the FIFO strategy provides a similar statistical performance as MDF. Moreover, the Optimal Ordering does not show big difference of the solution quality, so we can state that **Hypothesis 2 is refused**. We have found no strong influence of the selection of the ordering strategy on efficiency of the solver, we have not even found influence of usage of the FFD heuristic to the solution quality.

3.3 Insertion Heuristic

This experiment demonstrates the error of the Vehicle Agent insertion heuristic. The solution of a baseline solver is compared to the solution of the solver with the baseline Task Agent and Allocation Agent (i.e. optimal Order and optimal Allocation) and the optimal number of regular Vehicle Agents.

The maximum reached solution quality for this solver setting has been 100%, minimum 80% and the average quality has been 95%. The average error of the insertion heuristics computed for each Vehicle Agent route's individually has been 0.8%, the maximal single route error has been 9%.

3.4 Strategies Composition

This experiment evaluates the efficiency of Allocation Agent strategies combined with a different task processing strategy of Task Agent. The effectiveness of the used strategies can be expressed as an aggregation of the ranks across the benchmark instances and solvers (see Figure 3).

The strategies utilized by the solver are expressed using their abbreviations as defined in Section 2 in the form:

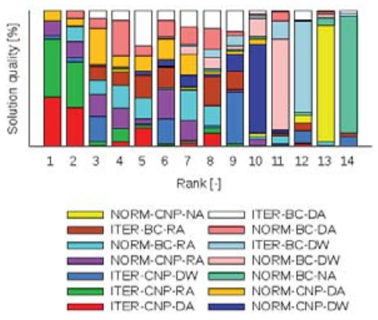


Figure 3: A histogram showing which task processing, allocation, and optimization strategies are more effective in all instances.

task processing strategy – allocation strategy – improvement strategy. If no improvement strategy is used, it is denoted as `NA`. We have run the solvers for all task ordering strategies and selected the best results to eliminate the influence of the task ordering strategy. The histogram in Figure 3 shows the `ITER-CNP-DA` together with `ITER-CNP-RA` are in 78% of the instances ranked as the best generic solvers. On the other hand, the worst strategies are `NORM-BC-NA` and `NORM-CNP-NA` which are ranked as last and one before last in 86% and 85% of the instances. Strategies `NORM-CNP-DW`, `NORM-BC-DW`, and `ITER-BC-DW` also show considerably bad effectiveness. These occupy ranks from 10–12 (out of 14) of the chart in 66% of instances.

3.5 Computation Time

This experiment focuses of the computation time of the composite solver. Figure 4 shows the average computation time of all combinations of strategies and orderings on the benchmark instances. During the time, the composite solver provides the solution presented in Figure 2. The x-axis represents the number of nodes in the instance. The y-axis represents the processing time of the composite solver (the aggregated time of 42 solvers for each instance). For all instances, the first solution (from one of the 42 solvers) has been obtained in a few milliseconds (11 milliseconds for the largest instance with 484 nodes) and the longest processing time of a single solver is close to 10 seconds (again for the instance with 484 nodes). The lowest processing time of the composite solver has been 6 milliseconds (*P-n16-k8*) and the longest time has been 49 seconds (*E484-19k*). The computation time of the solver in the experiments is upper-bounded by $\frac{n^3}{1500}$, which is better than the worst case complexity denoted by Equation 12.

3.6 Number of Vehicle Agents – Optimization

In this experiment we investigate the influence of the initial setting of the number of Vehicle Agents. We have compared two settings – the lower-bound and upper-bound.

Thanks to improvement strategies the solution provided by the solver starting with a lower-bound number of Vehicle Agents (L-B) and the solver with an upper-bound number of Vehicle Agents (U-B) is the same. On all instances the solution of U-B solver converged to the exactly the same solution as L-B solver. The only difference was the instance *A-n62-k8*, where the U-B solver provides the solution of quality

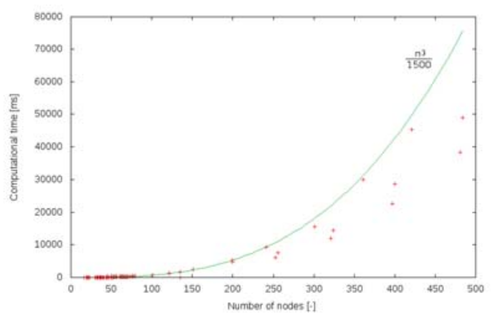


Figure 4: Computation time (aggregated) of the algorithm for all instances.

Number of vehicles	max	min	average	deviation
Unlimited	100.0	81.82	92.22	4.95
Minimal	100.0	76.95	91.0	5.64

Table 3: A comparison of solution quality of the best solutions by the numbers of vehicles.

0.88%, but the L-B solver stuck on 0.86% (both produce the solution using 8 vehicles, which is the optimal value). The computation time of the U-B solver was 2 times longer than L-B solver on average, in the worst case it was 5.4 times longer and in the best case 1.6 times faster (it was slightly faster in 6% of all cases). In the case of the U-B solver, the allocation and improvement strategies reduce the number of used Vehicles Agents and the solution converge to the solution of L-B solver. This experiment demonstrates the emergent self-organization ability of the agent-based system and **strongly supports Hypothesis 1**.

3.7 Number of Vehicle Agents – Constraint

This experiment focuses on the constrained number of vehicles. The solution obtained by the composite solver doesn't always meet the potential constraint for a minimal number of vehicles used (the BPP part of the VRP). A solution using the minimal number of vehicles has been produced by the solver for 92 instances. Table 3 shows the difference between the best solutions produced by the composite solver constrained by the minimal number of vehicles (given by the known optimal solution) and the solution with an unlimited number of vehicles. In the constrained case, some of the solvers do not provide the solution and thus the composite solver provides worse result. For three particular instances (*E421-41k*, *P-n55-k8*, *P-n55-k15*), the solution with an optimal number of vehicles could not be found using any combination of solver strategies (those three are not included in the computation of values for the case with minimal number of vehicles in Table 3).

4. CONCLUSIONS

We have designed and evaluated a multi-agent VRP solver that provides a feasible solution in polynomial time. The quality of the solution has been over 81% of all benchmark instances that greatly outperforms the known lower-bound approximation. The time complexity of the solver on experimental instances has been upper-bounded by $O(n^3)$. We have defined the bounds for the Vehicle Agents and shown that thanks to self-organization principles the presented agent-based solver converges to the same solution when starting from the lower bound and upper bound number of vehicles.

The presented solver architecture supports great runtime parallelization and it is able to provide the first solution in a very short time with good solution quality (for the biggest instance the first solution is produced under 10 milliseconds with 50% solution quality).

The results show that iterative task processing provides better results than batch processing and the backtracking strategy is not very efficient. The best performance has been reached with iterative task processing and ordinary CNP allocation with Delegate All and Reallocate All improvement strategies. The possible extension of the solver would be to introduce more improvement strategies and the mechanism of their runtime combination but in exchange of the computation time (and also worst-case complexity) increase. Another extension should be randomization of the task ordering strategies.

The presented solver demonstrates very good applicability to the VRP problem and easy adaptation to problem variants. Generalization of this approach to other problems (e.g. multiple traveling repairman problem and its variants) seems to be a promising way of future research.

5. REFERENCES

[1] D. Barbucha and P. Jedrzejowicz. Agent-based approach to the dynamic vehicle routing problem. In *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*, volume 55 of *Advances in Soft Computing*, pages 169–178. Springer Berlin / Heidelberg, 2009.

[2] R. I. Brafman and C. Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, pages 28–35, 2008.

[3] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[4] G. Dósa. The tight bound of first fit decreasing bin-packing algorithm is ffd(i) <= 11/9opt(i) + 6/9. In *ESCAPE*, pages 1–11, 2007.

[5] M. Haimovich, R. Kan, and L. Stougie. *Vehicle routing : methods and studies*, chapter Analysis Of Heuristics For Vehicle Routing Problems, pages 47–61. Elsevier, Amsterdam, 1988.

[6] G. Laporte, M. Gendreau, and J.-Y. Potvin. Classical and modern heuristics for the vehicle routing problem. Technical report, GERAD, 1999.

[7] Y. Marinakis and A. Migdalas. Annotated bibliography in vehicle routing. *Operational Research*, 7(1):27–46, January 2007.

[8] D. J. Rosenkrantz, R. E. Stearns, and P. M. L. II. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, 6(3):563–581, 1977.

[9] P. Toth and D. Vigo. *The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications*. Society for Industrial and Applied Mathematics, U.S., 2001.

[10] B. Zeddini, M. Temani, A. Yassine, and K. Ghedira. An agent-oriented approach for the dynamic vehicle routing problem. *International Workshop on Advanced Information Systems for Enterprises*, 0:70–76, 2008.

The RBVO Formation Protocol

Jiří Vokřínek*, Jiří Bíba and Jiří Hodík

Agent Technology Center
Gerstner Laboratory
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2, 166 27 Prague, Czech Republic
E-mail: vokrinek@agents.felk.cvut.cz
E-mail: biba@agents.felk.cvut.cz
E-mail: hodik@agents.felk.cvut.cz
*Corresponding author

Jaromír Vybíhal

Profinit, s.r.o.
Tychonova 2
160 00 Prague, Czech Republic
E-mail: jaromir.vybihal@profinit.eu

Abstract: The proposed protocol has been designed to support the flexible formation of Request-Based Virtual Organisations (RBVOs) with an emphasis on reflecting the conditions of real competitive environments. It supports automated or semi-automated negotiations mainly in the creation part of a Virtual Organisation (VO) life cycle and it accounts for the use of Service Level Agreements (SLAs). The protocol consists of three phases: (1) potential partner search, (2) negotiation of SLAs and the RBVO establishment, and (3) RBVO (execution and) dissolution. The protocol complies with Foundation for Intelligent Physical Agents (FIPA) standards and it has been utilised in the multiagent prototype in a real industrial case. The prototype has been implemented on top of a web-service-based agent platform.

Keywords: negotiation; protocol; virtual organisation; electronic business; e-business; web services; agent platform; service level agreement; SLA.

Reference to this paper should be made as follows: Vokřínek, J., Bíba, J., Hodík, J. and Vybíhal, J. (2009) 'The RBVO Formation Protocol', *Int. J. Agent-Oriented Software Engineering*, Vol. 3, Nos. 2/3, pp.135–162.

Biographical notes: Jiří Vokřínek is a Researcher in the Agent Technology Center at Gerstner Laboratory, Czech Technical University in Prague, Czech Republic. He has a university degree on Replanning in Multi-agent Systems. His research interests are artificial intelligence, multi-agent systems, planning and replanning in manufacturing, virtual organisations, supply-chain management and logistics. He has participated in several research projects focused on agent-based planning and simulation.

Jiří Bíba works as a Researcher in the Agent Technology Center at Gerstner Laboratory of the Czech Technical University in Prague, Czech Republic. He holds a Master's degree in Distributed Artificial Intelligence. His main research interests are the reconfiguration and evolution of commitments in competitive multi-agent environments, negotiation means for flexible contracting using decommitments for the optimisation of commitments, formal representation of commitments and monitoring mechanisms for inspecting adherence of agents to their commitments.

Jiří Hodík works as a Research Fellow in the Agent Technology Center at the Gerstner Laboratory, Czech Technical University in Prague, Czech Republic. He graduated in Technical Cybernetics in 2001. His research interests include artificial intelligence, multi-agent systems, auctions, multicriteria decision making, virtual markets, trust and reputation building and management, and virtual organisations. During a visiting scholarship to the State University of New York at Binghamton, he worked on immune system methods for information security.

Jaromír Vybíhal has a Master's degree in Biomedical Engineering at the Faculty of Electrical Engineering of the Czech Technical University in Prague, Czech Republic. He gained great experience from the Agent Technology Center of the Gerstner Laboratory in the field of competitive negotiation, reconfiguration and contracting in multi-agent systems. He is currently working as IT Consultant in Profinit s.r.o.

1 Introduction

The formation of a Virtual Organisation (VO) is based on a negotiation between independent partners that are willing to cooperate. Individual partners (mostly Small and Medium Enterprises (SMEs)) are motivated to join the VO to increase their business opportunities and to participate in larger-scale jobs.

The VOs naturally operate in a competitive environment. Every partner follows its own goals and maximises its utility. Each of the individual utility functions may use different metrics and they are usually hidden from the others. Standardised protocols for contracting are often insufficient for bargaining over contracts in such an environment, as the related negotiation mechanisms do not account for it (Vokřínek *et al.*, 2007). This paper presents the Request-Based Virtual Organisation (RBVO) Formation Protocol, which aims at the creation of a VO in the competitive domain by two levels of iterative negotiation – the potential partners search and Service Level Agreement (SLA) negotiation. The execution of the VO is not directly covered by the protocol – it controls only the dissolution of the VO.

2 Theoretical background

The modern cooperation concepts go from subcontracting, through the supply chains and then to VOs. Wiendahl and Lutz (2002) has identified three basic types of subcontracting between partners in a network. Although other reasons for subcontracting exist as well (*e.g.*, strategic reasons), these three are also basic for VOs:

1 *Classic subcontracting*: The production of one partner (producer) is an input for the other one (consumer).

2 *Technology-driven subcontracting*: One partner processes a task, but lacks a competency for some of its parts. Therefore, for those parts of the task a suitable supplier is subcontracted.

3 *Capacity-driven subcontracting*: This is similar to the previous one, but the partner responsible for the task lacks a capacity. The missing capacity is outsourced.

This article proposes a protocol for cooperation establishment in the competitive environment. The protocol focuses on virtual consortia formation, which is in principle a technology- and capacity-driven subcontracting.

2.1 The virtual organisation concept

The VO is understood (*e.g.*, Faisst, 1997; Van Wijk *et al.*, 1998; Gruber and Nöster, 2005) to be "… a specific form of network organizations". Gruber and Nöster (as well as Van Wijk *et al.*, 1998, for example) also specifies the key features defined by most of the definitions: the extensive use of information technology to coordinate the partners, sharing risk and knowledge with partners and focus on core competencies. Fischer describes a core competency of an enterprise as a set of skills, technologies and know-how crucial for the added value provided by the enterprise (Fischer *et al.*, 1996). The core competency of a VO consists of the members' core competencies that are crucial for the added value of the VO. The other commonly mentioned features are (presented, for example, by Faisst, 1997 or Capó *et al.*, 2004): autonomy and independence of members, operating towards the customer as a single company and temporality of an existence, which is mission-oriented. Very often mentioned features are also a distribution of members and slight bureaucratic overhead, and one face to the customer.

In the same manner Aubrey (1991) describes the features of VO members. They are *autonomous* (each entity is an independent company or freelancer with its own interests, commitments and goals), *distributed* (entities are naturally distributed in the real world) and *heterogeneous* (each entity may use different technologies and procedures). All these aspects are directly addressed by distributed artificial intelligence and its component of multiagent technologies (Molina *et al.*, 1998) that have already been utilised in the domain of VOs. Fischer (1996) defines the VO and the agents employed in it. Petersen *et al.* (2001) describes the use of agents for the modelling of Virtual Enterprises (VEs, a special case of a VO concept).

Most of the presented features of a VO are commonly accepted, although along with them, there are implementations of a VO methodology that do not fully comply with all the enumerated features. For example, Faisst (1997) draws attention to VOs that may exist without being supported by information technology. Also, the autonomy of members is restricted in some works, where altruistic behaviour is expected from them. In the VO, it must hold that "loyalty is shared among the partners and the cooperation is based on trust and information technology" (Van Wijk *et al.*, 1998). It is necessary for fruitful collaboration. On the other hand, the VO members would participate in such collaboration only if it is also fruitful for them – purely altruistic behaviour is not typical in a real business environment. Nevertheless, VOs that optimise a common profit only also exist. In such cases, there are no private profits of individual members that could be decreased due to increases in the profits of others. Actually, the VO members are optimising their private profits independently based on the fact that profits are shared.

2.2 *Request-based virtual organisations*

The concept of RBVOs defined by Roberts *et al.* (2005) comprises a cluster of partnering organisations that can get along without a hierarchical ordering into a monolithic organisation. The RBVOs are short-lived entities that are formed to respond to business opportunities offered in electronic commerce. RBVO operations are based on predefined SLAs.

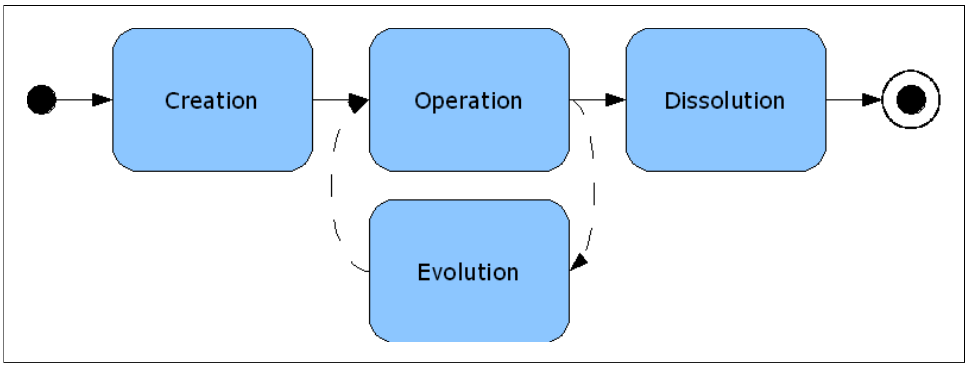
The organisation and functioning of RBVOs' activities are ensured by a community of intelligent agents that automate the procedures and operations of RBVOs (Mařík and McFarlane, 2005). In the RBVO defined by Roberts, the agents serve as assistants for human decision makers; in the agent system each participating SME is represented by its agent, which is able to undertake predefined automated decision-making support on behalf of the SME or enables a user to interact with the system on behalf of the SME. Other possible roles played by agents in VOs are defined, for example, in Hodik *et al.* (2007).

2.3 *The VO life cycle*

The VO life cycle and its phases have been described many times in previous works (*e.g.*, Fischer *et al.*, 1996). The basic phases (see Figure 1), which are included or extended in most definitions, are:

- the creation phase, which is the first phase after discovering a business opportunity. During this phase the VO is created: the VO task is defined, the VO team is formed and then the VO is initiated.
- the operation phase, which contains all the value-adding processes of the VO. In some cases there is a need for an evolution (also called adaptation) of the VO during this phase, *e.g.*, in the case of initiation of new VO members.
- the dissolution phase, which finalises and evaluates the VO operation and potentially opens future cooperation. When the task of the VO is accomplished, the VO operation may be evaluated.

Figure 1 The VO life cycle – the three basic phases and the optional evolution/ modification phase (see online version for colours)



Targeting these main life cycle phases, various authors split them into more phases and define additional subphases as well. The first main phase is creation. Fischer *et al.* (1996) distinguishes two phases of the creation process. In the first phase, the product is defined and the related business process is separated from the product is defined and the related set of business processes is formulated. In the second one, the team of VO members is negotiated and formed. In this second phase (*i.e.*, negotiation and formation, and supporting it with multiagent technology), Fischer distinguishes four subphases:

1 identification of potential partners

2 generation of alternative mappings from partners to individual business processes

3 evaluation of strategic interests and risk

4 finalisation of partners and mapping to partial processes.

A similar concept is presented by Tagg (2001), who has identified three phases of the life cycle for VEs: VE development (establishment phase), business development, and operational. The first and second phases correlate to the two phases defined by Fischer. During these two phases, the VO is created and set up during the establishment and business development phases. These phases consist of negotiations between potential members, checking for potential partners' credibility and authenticity, and contracts (tasks and responsibility allocation). The establishment phase is launched not only at the beginning of the VO; it also covers the expansion and adaptation of a VO team. The operational phase covers doing business and VO performance monitoring and management.

Extending Fischer's work, Faisst replaces the creation phase with three other ones: identification, formation and design (Faisst, 1997). The works of Fischer and Faisst are referred to by Rocha and Oliveira (1999) and Preece (2001), who point out four phases of the VO life cycle: identification of needs, partners selection, operation and dissolution. The creation phase is also extended by Van Wijk *et al.* (1998), who defines seven steps of the VO life cycle: modification of strategy, cooperation strategy, weighing cooperation alternatives, selection of partners, design and integration, management, dissolution and evaluation.

In work related to competency cells, Neubert also mentions the VO creation process (Neubert *et al.*, 2001). When a cell discovers a business opportunity (attracts a customer's production task), the first step that is done in creating a cooperation network is production planning. The cell creates a production plan or subcontracts a specialised cell to do it. The next steps are searching for partners and cooperation formation. The output of the second step (searching for partners) is a set of the potential configurations of a network. From this set the best possible configuration is chosen after negotiation with the agents of potential partners in the third step (cooperation formation). When a suitable configuration of cells is found, they are requested to confirm the obligations.

Most of the authors focus on the creation phase. On the other hand, there are authors like Camarinha-Matos and Afsarmanesh, who concentrate on the operation phase as well. They have defined a life cycle that consists of four phases: creation, operation, modification and dissolution (*e.g.*, in Camarinha-Matos and Afsarmanesh, 1998). The modification phase contains significant adaptations of VOs that cannot be executed during the operation phase. This phase is also called the *evolution phase* (Camarinha-Matos and Afsarmanesh, 2001).

There are projects, such as CONOISE/CONOISE-G, which concentrate on the operation phase as well (Shao *et al.*, 2004). The ‘happy path’ of their VO life cycle consists of the following phases: formation, operation and dissolution. These phases may be extended by *perturbation*, which is applied in case of some of these events:

- significant deviation is identified and its implications are eliminated by the VO adaptation/evolution
- the VO manager has identified a provider who can provide some service that is included in the plan of the VO in a better way (quality, price, *etc.*) than the provider already included in the VO team, and has negotiated a substitution.

2.4 Cooperation, coordination and commitments in competitive environments

The VO establishment is based on an agreement on the cooperation of individual partners. The concept of *social commitments* was introduced by Wooldridge and Jennings (1999). This concept may be applicable in some VO domains, but it does not address the problem of the unilaterally advantageous dropping of commitments. In most VO domains, an explicit employment of rewards and penalties is needed as a clear qualification of the utilities that the party gains or loses. The concept of such an explicit utility evaluation is then a part of the commitments; the party providing a service commits not only to perform appropriate actions (in order to gain the promised utility which is its motivation), but also to provide compensation in case of failure (*e.g.*, a compensation of the profit lost to the other party). The most complete approach on the commitments in the competitive environment has been presented by Sandholm and Lesser (2001) as Levelled Commitment Contracts (LCCs), which include an explicit utility evaluation in the form of a contract price and penalties. In order to provide a complete decision-making mechanism, the authors applied several significant restrictions (*e.g.*, the utility function needs to be identical for all participants, the opportunity-cost business probability function for every agent is common knowledge, *etc.*). These assumptions are limiting (Bíba and Vokřínek, 2006) and basically prevent the direct deployment of LCC in a real application. Nevertheless, LCC introduces a basis for the notion of commitments in competitive environments.

An SLA introduces a formalisation of a business relationship (or a part of a business relationship) between two parties (most often between a provider and a customer), which is a key concept for service management (Trienekens *et al.*, 2004). Usually it specifies the delivery of products or services for a certain price, meeting specified deadlines and quality requirements together with financial guarantees and other contract terms. It may concern continuous, discrete or one-shot service/goods deliveries. For an RBVO, it represents a description of work flows, schedules, resource allocations, participant roles, prices, sanctions, guarantees, legacy-related and other contract management and coordination issues. The SLA introduces a consistent (possibly reduced) electronic form of the contract signed by the contract parties as a paper document (the reduction may concern mainly nontechnical/financial parts) expressed in a machine-readable language (most often in XML, which is nowadays considered as an interoperable business information exchange format).

2.5 VO/RBVO formation mechanisms

There exist various methods of negotiating and coordinating team actions. Lomuscio *et al.* (2003) define negotiation as "…the process by which a group of agents communicate with one another to try to reach agreement on some matter of common interest". They define two components of the negotiation mechanism: the negotiation strategies and the negotiation protocol. The former defines the lists of actions that individual agents have planned to reach their desired goals. The latter (the protocol) defines rules for messages that are allowed in the message sequence. The rules:

- restrict the allowed types of messages (the 'performative' in the Foundation for Intelligent Physical Agents (FIPA)[1] messages)
- provide constraints for the message content
- define time-outs for receiving the message.

The most popular negotiation and coordination methods are derived from the Contract Net Protocol (CNP) and from the auctions. An introduction of the most important ones follows.

2.5.1 Contract Net Protocol

The Contract Net Protocol (CNP) is one of the most popular negotiation protocols ever used in Multi-Agent Systems (MAS). It comes from economics and is used in communities of altruistic as well as self-interested agents. The CNP was described by Smith (1980), who described a single-shot protocol for requesting and selecting a provider of a product or service in a group of one coordinator (who requests) and one or more participants (who may provide the needed item) (Smith, 1980). In the beginning of the session, the coordinator requests for offers from the participants and the interested ones reply with their offers. The coordinator evaluates the received offers and chooses the most suitable participant(s) or dissolves the session. Finally, if one or more offers are chosen, the coordinator grants the business to participants offering them.

In its basic form, the CNP provides a lot of freedom in each step of the interactions and the obligation to fulfil the contract defined in the call is not required in the basic CNP (*e.g.*, acceptance of proposals depends on the proposals themselves and the actual state of the coordinator at the moment of the proposals evaluation). The CNP fits well in collaborative environments where there is one subject evaluating the possibilities and the others are providing the most suitable offers for the call. In environments in which preference is somehow explicitly expressed (*e.g.*, by money) and/or in environments with competitive participants the CNP must be extended by rules and features, *e.g.*, known from the auctions (Ovcharenko *et al.*, 2006).

2.5.2 Auction

The auction is a method of optimal reallocation of resources according to the actual demand and supply, which are usually measured by monetary units. Many types of auctions exist; they vary in features like bid adaptation possibility, number of sellers and buyers, discrete or continuous evaluation of bids, number of criteria for evaluating a bid and others. The definitions of basic auctions are usually provided for a negotiation about a single issue with invariable features. Basically two auction mechanisms are possible: the one-shot and the iterative.

In the case of the former one, there is only one round of a negotiation. It means that the negotiation coordinator announces a proposal to which the participants respond by obligatory offers. Then the coordinator evaluates the received offers and announces the winning offer(s). There are two basic types of one-shot auctions; they differ in the price that the winner has to pay:

1 in the first-price-sealed-bid auction, the winner pays the price that he/she proposed

2 in the second-price-sealed-bid auction, the price to be paid is defined by the second best proposal.

The second-price-sealed-bid is usually called the Vickrey auction. Although it is an application of the Vickrey auction to the single-item single-unit domain, it is not the only Vickrey auction. The Vickrey auction is naturally a single-item multi-unit. For one kind of commodity (single item), it provides its redistribution of the commodity according to the match of the curves of supply and demand.

The iterative auctions are the English auction, where the price of the auctioned issue is being increased until only one participant is paying for it (for the reverse auction, the price is being decreased until only one interested provider remains), and the Dutch auction, where the price is too high (low in reversion auction) at the beginning of the negotiation and then it is being decreased (increased in the reverse auction) until a participant accepts it. When the English and the reverse English auctions are combined together, the Double auction is created. In that auction there are groups both of participants interested in selling and those interested in buying. The sellers overbid themselves by decreasing the required price, while the buyers increase it. When some selling and buying bids match, the auction is successfully finished. A very special subgroup in the group of iterative auctions are Continuous auctions, in which the bids are evaluated online and when some of them match, the exchange is executed. Independent of an identified match, the auction continues to identify another match of bids.

The iterative auctions are more complicated than the single-shot auctions. In case of a one-criteria iterative auction, where each proposal may by evaluated by a number, *e.g.*, price, the solution is clear: the one with the highest (not dominated) offer is the winner (actually, the evaluation requires a comparability of each two values from the domain of definition to which the evaluation is mapped and the comparability must be transitive). Individual offers depend on the type of auction and the preferences of the participants. In case of a multicriteria description of the proposals such evaluation becomes incredibly difficult and therefore an iterative multicriteria auction is the most complicated one.

2.5.3 Legal Agreement Protocol

The Legal Agreement Protocol (LAP) (Perugini *et al.*, 2007) extends the Provisional Agreement Protocol and is based on Australian contract law. The protocol allows an M:N negotiation which is split into several phases. The first phase allows a not-binding negotiation (the agreed conditions do not imply any commitment for any of the parties) which enables the parties to reach a mutually advantageous compromise. The next phase consists in a binding negotiation over a binding offer (which can be accepted or rejected).

Once a contract is established, it is possible to terminate it in several ways: by fulfilling the contract (this does not require communication), unilateral decommitment under agreed penalties given by the agreement, mutual agreement about cancelling the contracts without penalties and a contract breach (not solved by the protocol – to be resolved per curriam). One instance of the protocol is started for each task (a single-task negotiation) and multiple tasks are negotiated independently in concurrent protocol instances (*i.e.*, multiple contracts). The LAP allows flexible negotiations including backtracking, withdrawing offers, temporary rejections, *etc.* (it is possible to implement various search algorithms such as depth-first search and A*). Decommitments are not negotiated upon, but are carried out unilaterally by informing the other party about the decommitment. The protocol does not directly support contract renegotiation – it is covered by cancellation of the contract or decommitment while new contract conditions are negotiated in a new contract. The protocol assumes safe message delivery and the absence of communication is involved as an interaction option of the protocol (the protocol explicitly considers timeouts). The authors of the LAP have proved various properties of the protocol, *e.g.*, the protocol is free of a communication deadlock (communication is always terminated, though the matter of mutual commitments and their status is not considered) (Perugini *et al.*, 2007).

2.5.4 Competitive Contract Net Protocol

The Competitive Contract Net Protocol (C-CNP) (Vokřínek *et al.*, 2007) is a FIPA-like protocol designed for flexible contracting in a competitive environment (*e.g.*, e-commerce and VOs) and aims at covering the whole contract life cycle, specifically:

- the contract conclusion phase
- the optional decommitment phase
- the contract termination phase.

Not all the parties involved in a multiround negotiation of commitments need to be addressed by Call-for-Proposal (CFP) messages. The protocol allows participants to impose their proposals (based on third-party information) onto an already running negotiation. The 1:N negotiation is held in a pairwise manner. During the execution phase any of the parties involved in pairwise commitments may attempt to decommit from the contract. The multiround decommitment negotiation on conditions of dissolving the cooperation may end up either by the decommitting party backing off (the contract returns to normal) or dropping the commitments under the payment of a penalty (the penalty may be fixed during the contract-conclusion negotiation or may remain open and be adjusted in time). Finally, in the termination phase, the results are evaluated with respect to the agreed commitments. Eventually, penalties for noncompliance with commitments are negotiated. The message content is assumed to describe the contract as a whole, *i.e.*, full and explicit descriptions of commitments (*i.e.*, not merely the task assignment, but also resource allocation, quality of service, schedules, *etc.*), rewards and sanctions are provided (such message content may be, for example, an SLA). Thus, the negotiation is also assumed to be multiattribute rather than single-attribute. The multiround manner of the protocol allows multiple, simultaneously running negotiations as well as multilevel ordering of subsequent protocols (*i.e.*, a participant of a C-CNP may become a coordinator of another subsequent C-CNP negotiation, *e.g.*, for outsourcing).

2.5.5 Renegotiable Competitive Contract Net Protocol

The Renegotiable Competitive Contract Net Protocol (RC-CNP) (Bíba *et al.*, 2008) extends the C-CNP through renegotiation phases, provides means for a fully flexible contracting in competitive environments and enables a consistent evolution of commitments starting at their creation and terminating with their fulfilment, adaptation or breach (even a partial breach) under punishment (payment of a penalty) – *i.e.*, it covers a complete commitment life cycle within a group of mutually committed agents (the commitments are pairwise between coordinator and participants). The protocol allows M:N multiattribute negotiations and it can be extended by not-binding phases. Moreover, it is capable of dealing with a possible temporary communication inaccessibility by utilising communication timeouts and related default transitions (as well as a possible synchronisation backtracking).

3 The RBVO Formation Protocol

The protocols and approaches presented in the previous section give solid bases for any negotiation-based cooperation establishment. In this work, we focus on the RBVO formation. It is based on a negotiation between independent actors that are willing to cooperate. Individual actors are motivated to join the VO to increase their business opportunities and to be able to participate in larger-scale contracts. An RBVO is a mutually agreed consortium of such individuals that is formed to cover a complex business opportunity. This business opportunity is introduced by the *coordinator*, which leads the negotiation with several potential *participants* – partners selected to cover some part of the business opportunity.

The protocols described before are not sufficient or are too complex for practical usage in the domain of RBVO formation. Thus we have designed the RBVO Formation Protocol, inspired by CNP and C-CNP, which allows the implementation of arbitrary business strategies or auctions. It is targeted at the negotiation protocol component of the negotiation mechanism (Lomuscio *et al.*, 2003). It does not address the planning step introduced in Neubert *et al.* (2001), but focuses on the searching for partners and cooperation formation steps. This work also builds on the results of other works (*e.g.*, Rocha and Oliveira, 1999; Preece, 2001; Van Wijk *et al.*, 1998) that focus on establishing a cooperation using the concept of VO as well.

In the RBVO domain, we understand the business opportunity introduced by the coordinator as the prepared production plan and RBVO formation as Neubert's configuration of the network.

3.1 Protocol description

The RBVO Formation Protocol supports a contract negotiation on several levels. It consists of three phases:

1 the search for potential partners (prenegotiation of contract)

2 negotiation of SLAs with selected partners and establishment of the RBVO (one partner or a small number of partners that together cover the required competencies)

3 (execution and) dissolution of the RBVO.

The first two phases enable multiround negotiations and concern the creation phase of an RBVO life cycle. Both the coordinator and the individual potential partners are allowed to withdraw from the negotiation for any reason. During the first two phases, the final RBVO configuration is agreed together with the related commitments (given by pairwise or multiparty SLAs) for all the parties. The final phase concerns the simplified execution and termination phases of the RBVO life cycle.

A sequence diagram of the RBVO Formation Protocol is shown in Figure 6 and the hybrid state diagram is in Figure 7. The individual phases (see Figure 2) are described in detail as follows:

Phase 1 the search for potential partners

The first phase aims at a prenegotiation of the contract conditions with possible partners (equipped with the required competencies) with respect to the ratings of their offers so that the number of partners selected for detailed negotiations of SLAs is reduced and the best candidates are chosen. The negotiation is started by sending a *Collaboration Request* (CR) message as a CFP. The CR describes the decomposed tasks, respective competencies required for their accomplishment and constraints (*e.g.*, geographical location of a potential candidate, due dates). The coordinator and the candidates enter into a pairwise multiround bargaining (in a *propose/counter-propose* manner) in which they try to agree on preliminary cooperation rules. Thus, the coordinator obtains several possible configurations of the resulting RBVO. The coordinator decides on the best configuration and sends the respective candidates *preliminary-accept* messages containing SLA proposals. The other candidates are not rejected immediately, but should remain in the first phase of the bargaining process while the preselected candidates enter the second phase. The waiting candidates may get their chance if the coordinator fails to reach agreements on SLAs with some of the preselected candidates. The granularity of information in CRs is generally less fine than in SLAs (some of the attributes may be irrelevant to negotiate upon until the pre-agreement is reached). Both the coordinator and the candidates are allowed to terminate the negotiation for any reason by sending *refuse* participation (candidates) or *reject* participation (coordinator) messages.

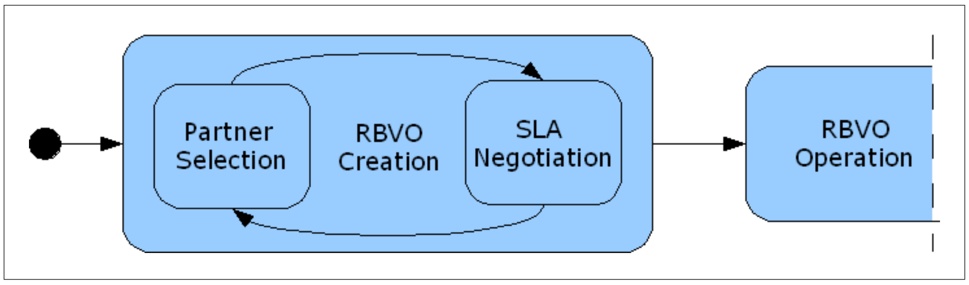
Phase 2 negotiation of SLAs and establishment of the RBVO

The coordinator and the candidates preselected for the negotiation of the detailed terms and conditions of the resulting SLAs try to reach a final agreement reflecting various aspects like time schedules, qualitative parameters, prices and penalties. The pairwise multiround bargaining (again in a *propose SLA/counter-propose SLA* manner) may be terminated by any of the parties provided a mutually advantageous compromise on the contract conditions appears unreachable. In such a case, some of the waiting candidates, equipped with respective competencies can be invited to the final negotiation by a preliminary accept message from the coordinator. As soon as all SLAs are finalised, *confirm-SLA* messages are sent to the candidates and the RBVO is practically established. The waiting candidates that are not included in the RBVO receive *reject-participation* messages. The RBVO is created and the execution and termination phases of the RBVO life cycle can take place.

Phase 3 (execution and) dissolution of the RBVO

As soon as a participant accomplishes its tasks, it sends an *SLA-done* message to the coordinator. The coordinator terminates the cooperation by confirming the dissolution of the RBVO.

Figure 2 The RBVO life cycle supported by the RBVO Formation Protocol (see online version for colours)



The RBVO Formation Protocol, as designed, does not provide direct support for the evolution (also named modification or perturbation) phase of the VO life cycle. The evolution may be invoked during the operation (execution) phase in case of incidents that endanger accomplishment of the RBVO mission or in case of an opportunity to increase the efficiency of the RBVO.

On the other hand, such a situation may be understood as a very special case of RBVO formation with additional constraints. The constraints are twofold:

1 Existing tasks which have been already started or finished can no longer be modified.

2 Existing tasks which have not been started can be modified.

The assignment of the latter ones may be modified. The modification process consists of:

- the substitution of the tasks to be modified by new decompositions considering the already fulfilled/finished tasks
- the invocation of both the prenegotiation and SLA negotiation phases of the protocol to find partners to take on the alternative obligations defined by the new decomposition subset.

The alternative obligations are applied to the involved participants as well as to the coordinator. It means that the coordinator may be obliged to pay a penalty in case of the cancellation of the contract, although the partner has not started working on it yet – the partner may have already booked resources for it and even if the partner will not use them, the related costs have to be paid.

This modification process influences the assignment only of the tasks included in the set to be modified. The other tasks are not influenced but they may generate constraints (especially for the available starting date and due dates) that the coordinator must take into account during the evaluation of offers received from potential participants. Actually, the list of tasks considered for the assignment modification may vary many times during this phase because it is affected by the ongoing negotiation.

3.2 Timeouts and accessibility synchronisation issues

According to Lomuscio *et al.* (2003) the timeouts for the messages should be defined. Not all messages have to be secured by timeouts. The critical messages are the collaboration request, propose, counter-propose, pre-accept, SLA propose and SLA counter-propose messages.

Due to the nature of the distributed environment, there are potential problems with timeouts and synchronisation of the protocol for each participant. An inconsistency of the protocol state can be caused by:

- a *message loss*, when a party sends the message and the other party does not receive it
- a *protocol breach* caused either intentionally by a party (*e.g.*, when it is unwilling to communicate with a certain party or an unexpected message is received) or by a software bug
- *communication inaccessibility*, when a party is not able to send messages to others
- a *timeout*, when some message is sent or received after the deadline (*e.g.*, the participation proposal).

All the above occasions lead to an inconsistency in the protocol state of the coordinator and one or more participants. The protocol has been designed to operate in dynamic environments, where it is often not possible to ensure that the participants will follow the protocol without intended or non-intended breaches (*e.g.*, because of timeouts or lost messages). Mainly, the initial CR may remain unanswered by the addressed participants for any of the presented reasons. In this case, the coordinator continues to execute the protocol and the participants have to synchronise their state. To keep the business logic consistent, the following suggestions are made:

- Missing received communication is understood as *refuse, reject* or *refuse SLA sent* (depending on the current protocol state).
- If the coordinator receives a message that breaches the protocol, it ignores it.
- If a participant receives an unexpected message, it has to synchronise its state and continue according to the protocol.

3.3 Protocol extension by exceptional messaging

In some cases, there is a need to adapt the protocol to enable more freedom for the business strategies. For example, the timeout of some proposals can be shorter than the timeout for the CR. In this case, the coordinator has to move the protocol state with a particular partner and continue with a *pre-accept* or *counter-proposal* message before another partner sends the initial proposal. In fact, this is not a protocol breach or inconsistency. Individual coordinator-to-participant relations are not affected, but due to the selected business strategy, the coordinator has to decide the proposal evaluation before all the proposals are received.

Let us discuss this issue with the following example. The coordinator is looking for a rent-a-car provider. Its business strategy is to look for the lowest available price with a desired daily rate under 200 euros. There are three potential participants: *A*, *B* and *C*; so the coordinator starts the negotiation by sending a CR to all of them with the timeout of one day. Participant *A* answers immediately with a proposal for 250 euros per day with a one-day validity. Participant *B* answers in one hour with a proposal for 180 euros, but this proposal is valid for one hour only. Participant *C* sends no proposal within the next hour. At this moment, the coordinator has to decide either to wait for the last proposal and potentially lose the best offer or to finish the negotiation by accepting the proposal of Participant *B*. The coordinator chooses the second option and sends a *pre-accept* message to Participant *B* and continues to the SLA agreement. When the SLA is confirmed (between the coordinator and Participant *B*), the coordinator sends *reject* to Participant *A*. According to the protocol, it is not possible to send *reject* to Participant *C*, but if Participant *C* sends a proposal in the future it should certainly be rejected as well.

This example illustrates the need for exceptional protocol handling due to complex business strategies. We introduce *in-advance messages* for handling this. Such a message can be understood as a default message that will be used as a response to another party's activity. This type of functionality is mainly needed at the coordinator side, so the usual in-advance messages are *counter-propose*, *pre-accept* and *reject*. In-advance messages are handled by the protocol controller to ensure consistency and correct termination of the protocol.

Another type of exceptional message can be provoked by changes during negotiation. In general, the negotiation is performed in a dynamic environment, where individual agreements affect each other. An example would be as follows: The coordinator receives a proposal from another participant not involved in the negotiation yet. It sends a *counter-propose* message, but meanwhile (as more proposals are received) the first proposal becomes satisfactory. At this moment, the coordinator is willing to accept the proposal, but this leads to a protocol breach. To help with this issue, we introduce *instant messages* that can be sent even if it is not the particular actor's turn. The possible instant messages for the coordinator are *counter-proposal*, *pre-accept* and *reject*. For the participant, the possible instant message is *proposal*.

Both types of messages are summarised in Table 1. In-advance and instant messages address the need for exceptional messaging. The protocol is not breached by using those messages, but the synchronisation issue arises (see Section 3.2).

Table 1 Exceptional messages of the RBVO Formation Protocol

Action	*Message type*	*Actor*
Counter-propose	In-advance	Coordinator
Pre-accept	In-advance	Coordinator
Reject	In-advance	Coordinator
Counter-propose	Instant	Coordinator
Pre-accept	Instant	Coordinator
Reject	Instant	Coordinator
Propose	Instant	Participant

4 Auctions implementation using the RBVO Formation Protocol

The process of RBVO creation consists of the following steps:

Step 1 the search for partners

Step 2 obligations finalisation during RBVO establishment.

The auctions usually concentrate on the latter phase and they do not solve the former one; the potential partners are identified before or during the negotiation initiation. Another difference between the RBVO Formation Protocol and classical auctions is that the auctions mostly define convergent criteria for the proposals and counterproposals offered during the negotiation; the RBVO Formation Protocol does not limit the composition and evaluation process of proposals and counterproposals. Therefore the RBVO Formation Protocol may implement almost any auction mechanism type for the negotiation. This is done by applying the rules defining the chosen auction type to the RBVO Formation Protocol.

In the RBVO Formation Protocol, the auction mechanisms may be applied during the RBVO establishment phase. Many auction mechanisms exist and their suitability for negotiation is domain dependent. Here we provide examples of how the auction mechanisms may be included in the RBVO Formation Protocol:

- The one-shot auctions (*e.g.*, first-price-sealed-bid or second-price-sealed-bid auctions) are implemented by one round of SLA negotiations, when the SLA proposal is provided by the coordinator to the participants, which consequently respond with counterproposals. Such counterproposals are evaluated according to the used auction rules and the winner(s) is (are) announced by the final SLA proposal(s) that they agree and that are finally confirmed by the coordinator.
- The English iterative auction may be implemented through continuous SLA proposing being conducted by the coordinator. During the auction process, the coordinator adapts the offer in order to slightly improve its own potential profit, as it would be if the proposed SLA is confirmed by all parties. The negotiation continues until the group of interested participants is decreased to the minimal group included in the formed RBVO. The interested participants' group is expected to be reduced because increasing the potential profit of the coordinator negatively influences the potential profit of the participants in the RBVO. Therefore they leave the negotiation because the conditions under negotiation may become disadvantageous for them. The last participant(s) standing is (are) the winner(s).
- The Dutch iterative auction may be implemented similarly to the English auction, but the first proposal given by the coordinator is not profitable for any participant to accept it. Then the proposed conditions are continuously adapted in order to make them more interesting for the potential participants until the group of participants covering all the requirements for the RBVO establishment is identified by their acceptance of the proposals given by the coordinator.

5 The PANDA case study

The presented RBVO Formation Protocol has been utilised by the multiagent system prototype within the PANDA[2] project, which aims at collaborative process automation in the ERP/CRM[3] industry. It facilitates the creation of international e-collaborations based on RBVO formation using sector-specific SLAs and a community of intelligent agents. The individual value chain actors act as service providers mainly for consulting, software implementation, installation and customisation, training and maintenance. The motivation is to find the most suitable consortium of service providers to meet customer requirements such as cost, experience in the industrial domain and appropriate ERP solution, geographical location and language.

The system is composed of a set of distributed partner agents and a central or distributed platform that supports services. The central platform services provide public data and they are considered to be fully accessible for the agents (that are online). The agents operate in a semi-accessible mode – it means the agents can be inaccessible for some time (*e.g.*, any agent can be offline or turned off).

The PANDA system prototype is composed of the following components:

- portal – a unified user interface that incorporates all the user interactions with the whole system
- central services – data and directory services providing competency taxonomies, profiles, *etc.*
- agent platform-supporting services – web-services-based FIPA-compliant platform implementation provides agent management, online and offline agent communication channel and directory services
- partner agents – distributed components representing the companies operating in the system. Every company deploys one agent that performs negotiations on behalf of its owner.

The PANDA intelligent agents play the role of the partners' representatives for (semi-)automated negotiation that supports the e-business acceleration in the ERP value chain domain. Although inspired by real business, the outcome of the PANDA intelligent agents system should be generalised for the wider domain and be adaptable for any value chain and different domains.

The generic constrains of the PANDA case should be summarised as follows:

- The system should support independent, self-interested, geographically distributed players registered within the chain.
- Every member can introduce new business opportunities and start the RBVO formation.
- The RBVO is formed upon peer-to-peer negotiations.
- Every member is able to use private preferences and constraints during negotiations.
- The coordinator (the one who starts the RBVO formation) is able to evaluate the cooperation proposals using private preferences and rules. It is also able to choose the type of auction and evaluation method.
- The actors' responses are based on dynamic private knowledge.

The main goal of the PANDA multiagent system is to provide negotiation-based matchmaking methods, taking into account limited information provision, multicriteria evaluation of proposals and private preferences and metrics for each participant. Using public data, previous experience or reputation mechanisms, the coordinator starts the negotiation (using the defined communication protocol – the RBVO Formation Protocol) with a preselected set of potential participants, which provides the highest probability of matching the coordinator preferences after the negotiation phase. The peer-to-peer negotiation is used for gathering semiprivate knowledge to be able to select the appropriate partners (proposals are based on participants' private preferences and availability and cannot be evaluated without negotiation) and construct the RBVO. The finalisation of the matchmaking is done by the RBVO proposals evaluation and there is a possible backtracking when the evaluation gives nonsatisfactory results.

A typical CR contains global constraints (*e.g.*, validity of the CR, owner, deadline, overall indicative budget) and a list of tasks. Each of the tasks contains more detailed constraints (*e.g.*, the service required, starting date, end date, language, location, resource amount required) and provider constraints (*e.g.*, required expertise domains, ERP modules expertise, reputation). By way of illustration, the CR for RBVO can by composed of the following tasks:

Task 1	implementation in Germany, required expertise in the food industry and sales management ERP module
Task 2	consulting in Germany in English, required expertise in sales management ERP module
Task 3	support in Germany in English, required expertise in the food industry.

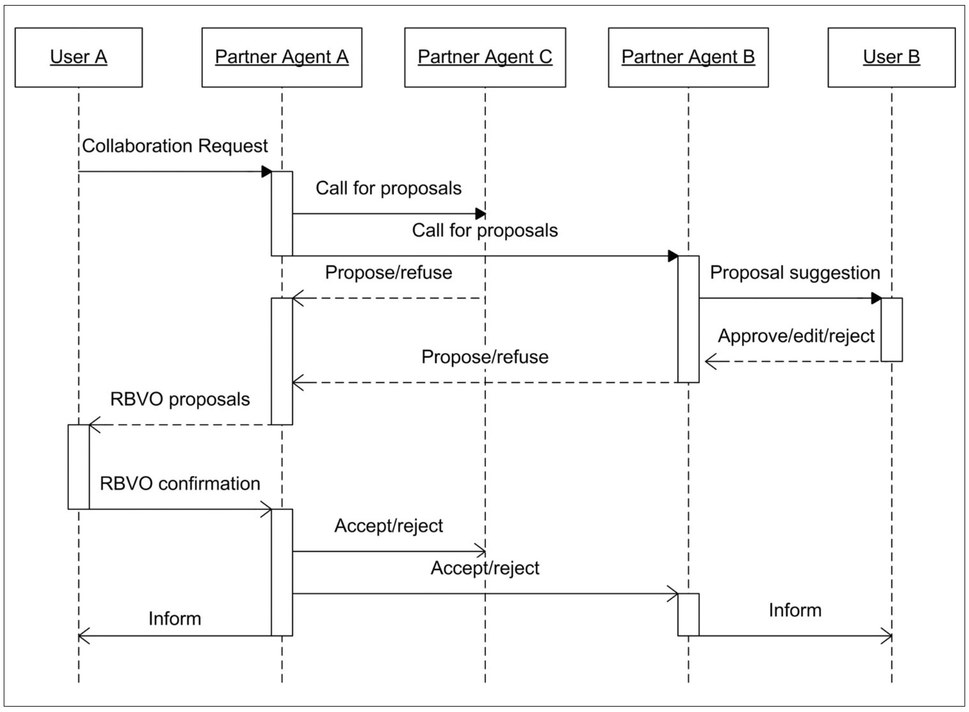
The coordinator is looking for a consortium of participants – an RBVO covering all three tasks in the best condition. The evaluation metrics of the RBVO is private for the coordinator.

5.1 Deployment of the RBVO Formation Protocol in PANDA

The negotiation-based matchmaking is based on the interaction of the coordinator and the potential participants, where all actors follow their own private strategies. The interaction is based on the presented RBVO Formation Protocol. The coordinator introduces a business opportunity and starts to negotiate the potential RBVOs. It facilitates the multiattribute/multicriteria evaluation of the potential RBVOs combined from the received proposals. The goal is to find a pareto-optimal set of RBVO clusters with respect to user-defined constraints and preferences (represented by rules, weights, *etc.*). Such RBVO clusters are dynamically created from a combination of proposals and appropriate counterproposals are generated to converge to the desired solution.

The example of the RBVO Formation Protocol utilisation is given in Figure 3. There are three Partner Agents (*A*, *B* and *C*) and two involved Users (*A* and *B*). The figure illustrates the first stage of the RBVO Formation Protocol with no counterproposals. Company *A* (represented by Partner Agent *A* and User *A*) is the coordinator and Companies *B* and *C* are the participants. While Company *C* is represented by a fully automatic agent with no user interaction, Company *B* is represented by a semi-automated agent in the role of an assistant. Also, the final selection of the best RBVO is approved by the human user.

Figure 3 A simplified example of RBVO Formation Protocol utilisation in PANDA



Let us discuss a realistic scenario based on the CR defined above. Let Participant *B* propose to provide all three services for an overall price of 1000 euros. Participant *C* proposes the provision of Task 1 for the price of 500 euros. There are two potential RBVOs constructed from this proposal:

1 all three tasks will be provided by Participant *B* for 1000 euros

2 Task 1 will be provided by Participant *C* for 500 euros and Tasks 2 and 3 will be provided by Participant *B* for less than 1000 euros.

For the second RBVO, the coordinator does not have enough information to evaluate the price correctly, so the counterproposal should be made to Partner *B* for the provision of Tasks 2 and 3. In the worst case, all the combinations have to be examined. The RBVO Formation Protocol allows such bargaining to reach the most suitable solution, but the logic of the bidding is within the scope of the implemented business strategy. When the final RBVO is selected, the coordinator sends the respective pre-accept messages and continues with the SLA negotiation phase. In the PANDA demonstration prototype, the SLA negotiation is done mainly offline or in the form of text document attachments. The second and third phases of the protocol usually follow the simplest, straightforward, message sequence: *SLA propose*, *agree SLA*, *confirm SLA*, *SLA done*.

5.2 Rule-based matchmaking strategies using the RBVO Formation Protocol

As described before, the RBVO Formation Protocol can be used for the utilisation of any type of auction mechanism. In the PANDA prototype, the business logic is provided by the agent Knowledge Processing Module. It utilises the Drools engine – a business rule management system provided by JBoss.[4] During the negotiation driven by the RBVO Formation Protocol, the Knowledge Processing Module executes a set of business rules to create a proposal as a participant, and evaluate the obtained proposals and generate counterproposals as a coordinator. This approach enables the abstraction of the business logic layer from the protocol and implementation of any kind of business strategy on top of the RBVO Formation Protocol. Table 2 shows an example of the business rules implemented in the prototype. The overall optimisation strategy is to get the cheapest possible RBVO while all the defined constraints are satisfied.

Table 2 Example of business rules implemented by the PANDA partner agent

Business rules
I need to finish the negotiations in x days
Only consider partners that offer services in the same country as the requested services (Y/N)
Only consider services offered by local partners (Y/N)
Reject listed sellers – blacklist (Y/N)
Don't show me consortia > x partners
Offer discount x% for specified buyer
I will always start my bidding at my listing price, irrespective of the project budget (Y/N)
I do not want to participate in projects less than x euros
I do not want to participate in tasks less than x Mandays
I do not want to participate in projects (*i.e.*, set of tasks offered to me) less than x Mandays
I do not want to participate in projects more than x euros
Reject listed buyers – blacklist (Y/N)
Give a discount x% in my reply that include multiple tasks

The whole negotiation matchmaking logic in PANDA is supported by three components:

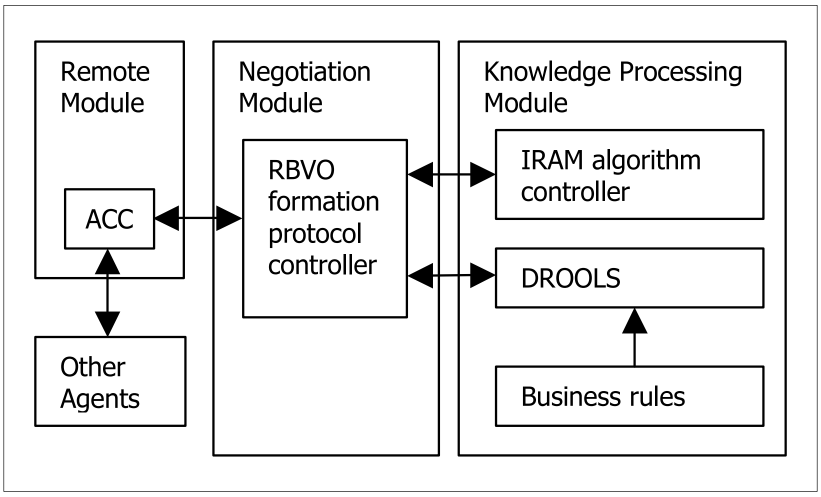
1 the RBVO Formation Protocol as a negotiation frame

2 business rules as a proposal generation and RBVO evaluation tool

3 the Incrementally Refined Acquaintance Model (IRAM) algorithm as an optimisation strategy.

The iterations of negotiation towards RBVO formation are executed by the IRAM algorithm (Pěchouček *et al.*, 2008), which provides fast convergence to the optimum while minimising the loss of private information. It enables one to minimise the number of counterproposals and (thus the information obtained) needed to reach the best RBVO. The algorithm is also able to provide the best known solution in every iteration step. The schema of the agent processing logic is in Figure 4. The agent communicates via the *Remote Module* and Agent Communication Channel (ACC) with other agents according to the RBVO Formation Protocol controlled by the *Negotiation Module*. The *Knowledge Processing Module* behaviour depends on the agent's role in the particular negotiation. The business rules can differ for the coordinator and participant and the IRAM algorithm is utilised by the coordinator only.

Figure 4 The simplified structure of the PANDA prototype partner agent


Remote Module
ACC
Other Agents
Negotiation Module
RBVO formation protocol controller
Knowledge Processing Module
IRAM algorithm controller
DROOLS
Business rules

5.3 Agent system implementation

There are several technologies for integrating web services and agent platforms. One of the approaches is to enable the transparent cooperation of agents operating on an existing agent platform and web services (Esteban, 2007; Overeinder *et al.*, 2008). Such an integration is usually gateway based[5] – the service invocation or agent messaging is transparently translated and passed to the other side. So the agents can be invoked as web services (only simple protocols like request-respond are supported) and agents can communicate with web services in the agent manner. This approach is suitable for the integration of the existing agent system and web services and takes advantages (and also disadvantages) from both – the agent platform and web services technology.

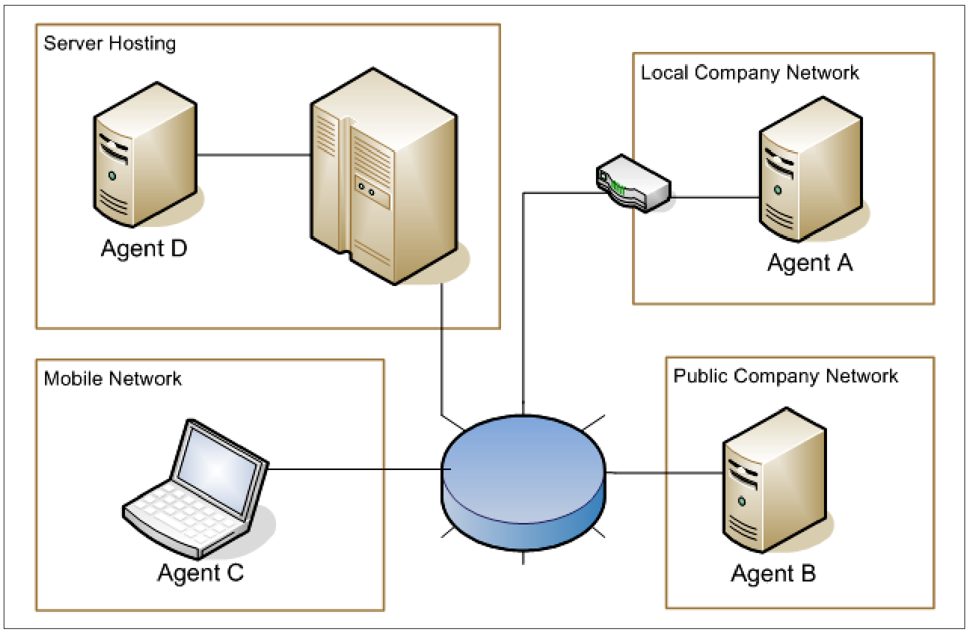
Since the PANDA agent system has been designed to work in an open internet environment from the beginning, the focus has been placed on the maximisation of the system's stability and robustness. To keep all the main features like FIPA compliance, openness, stability, usage of up-to-date standards and technologies, we have designed and implemented the web-service-based agent platform.

The agent platform complies with the FIPA standards, while all the platform services are implemented as web services (see Figure 8). The agent programming interface provides access to all the necessary interactions with the platform (Directory Services, Agent Management System and ACC). On top of the agent platform, the RBVO formation protocol controller has been implemented. Through this layer, the agents no longer need to implement standard message handling (in the manner of sendMessage and handleMessage methods), but the high-level methods are available for controlling the protocol.

For example, when a new negotiation for RBVO formation has to be started, the agent invokes (a standard JAVA method call) the method of the protocol controller with the CR as a parameter. According to the CR, the protocol controller invokes the directory service of the agent platform (the JAVA method call redirected to the WS invocation) and receives the addresses of potential collaborators. Then the protocol controller prepares the CFP and calls the sendMessage method of the platform (the JAVA method call redirected to the WS invocation – the direct invocation of the messaging service of online agents and the invocation of offline platform message service when needed). The messaging web service of the receivers puts the CFP message into the incoming message buffer. Then the protocol controller obtains the CFP, creates a new instance of the RBVO Formation Protocol for this call, and the preparation of a proposal is started. In the case of offline messaging, the agent checks the offline message box (WS invocation) upon starting, and then periodically, to handle temporal connection problems or a network configuration with a limited peer-to-peer communication.

The described implementation of the agent platform allows the development of lightweight independent agents with the ability to operate while being widely distributed across the internet in various network settings (see Figure 5). The web service nature of the platform guarantees agents' full operation as long as the web service invocation of the agent platform services is possible (the agents operate in the passive mode). When the invocation of the web services deployed on each agent (mainly distributed ACC) is enabled, the agents are able to communicate directly in full peer-to-peer manner (the agents operate in the active mode). The agent platform also provides several optional tools for message sniffing, logging and interactions inspection and visualisation.

Figure 5 Example of the deployment of the PANDA agents (see online version for colours)



Note: The agents can be hosted on the server and run on the public network, on the private network (behind address translation), or on the mobile devices.

Figure 6 RBVO Formation Protocol sequence diagram

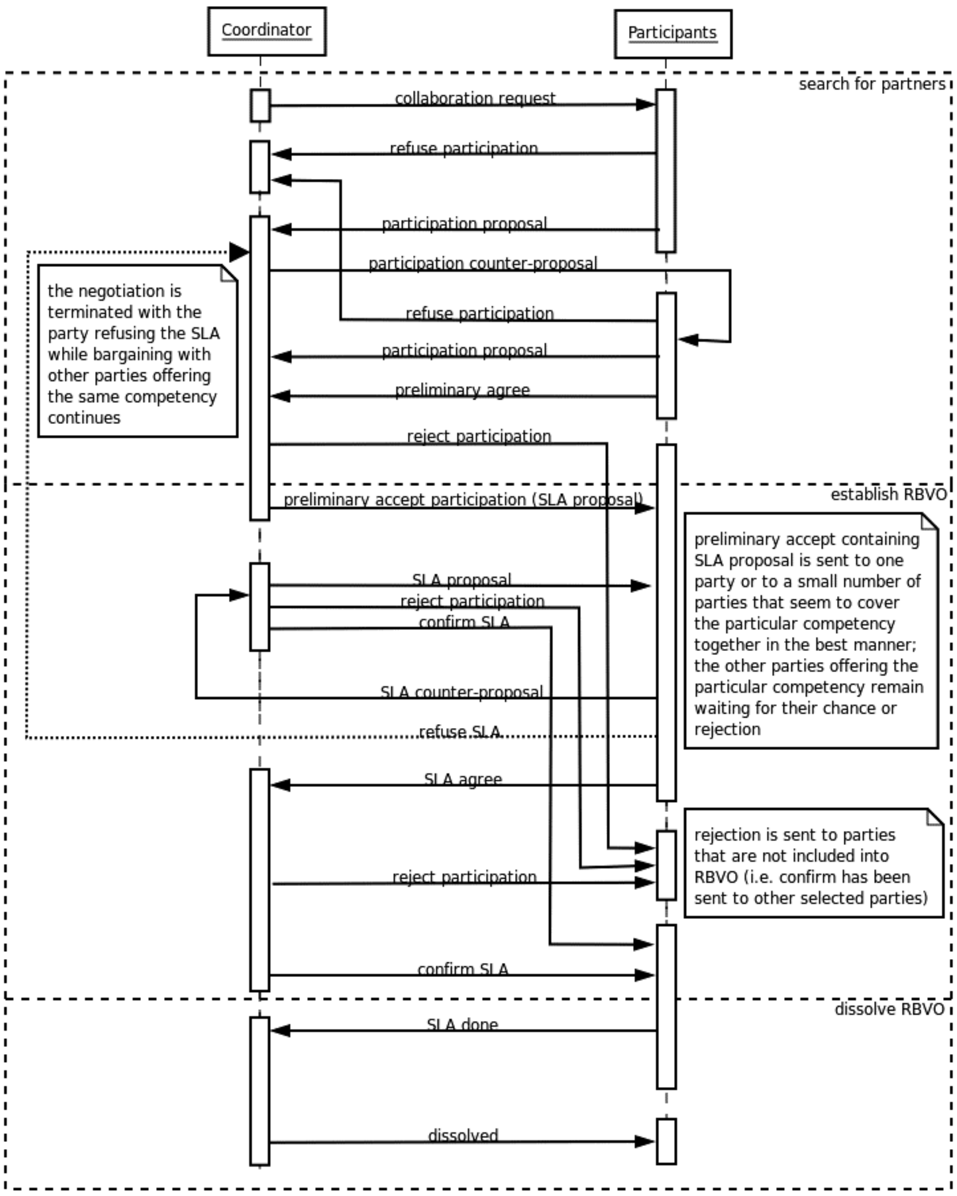

Coordinator
Participants
search for partners
collaboration request
refuse participation
participation proposal
participation counter-proposal
the negotiation is terminated with the party refusing the SLA while bargaining with other parties offering the same competency continues
refuse participation
participation proposal
preliminary agree
reject participation
establish RBVO
preliminary accept participation (SLA proposal)
preliminary accept containing SLA proposal is sent to one party or to a small number of parties that seem to cover the particular competency together in the best manner; the other parties offering the particular competency remain waiting for their chance or rejection
SLA proposal
reject participation
confirm SLA
SLA counter-proposal
refuse SLA
SLA agree
rejection is sent to parties that are not included into RBVO (i.e. confirm has been sent to other selected parties)
reject participation
confirm SLA
dissolve RBVO
SLA done
dissolved

Figure 7 RBVO Formation Protocol hybrid state diagram (see online version for colours)

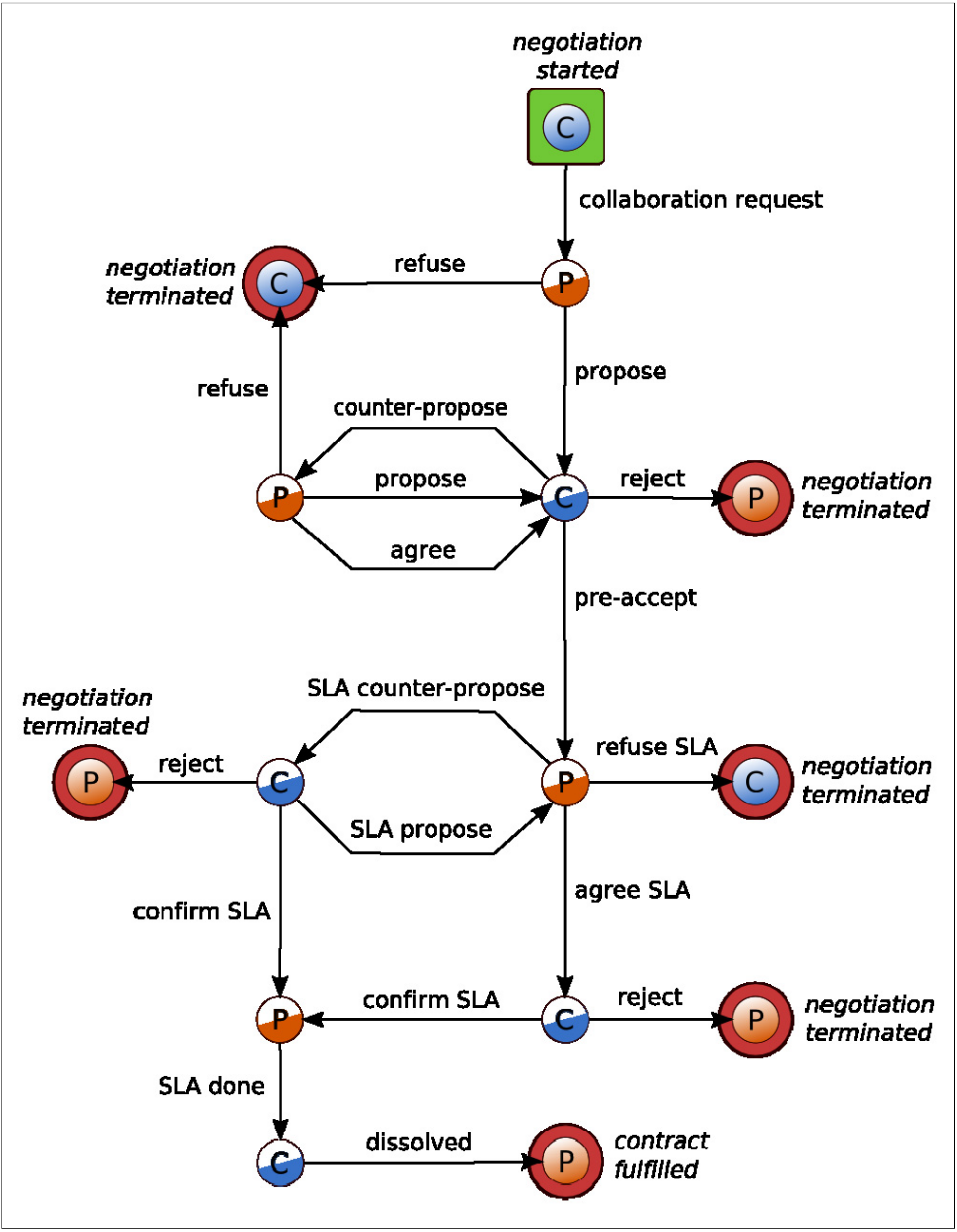
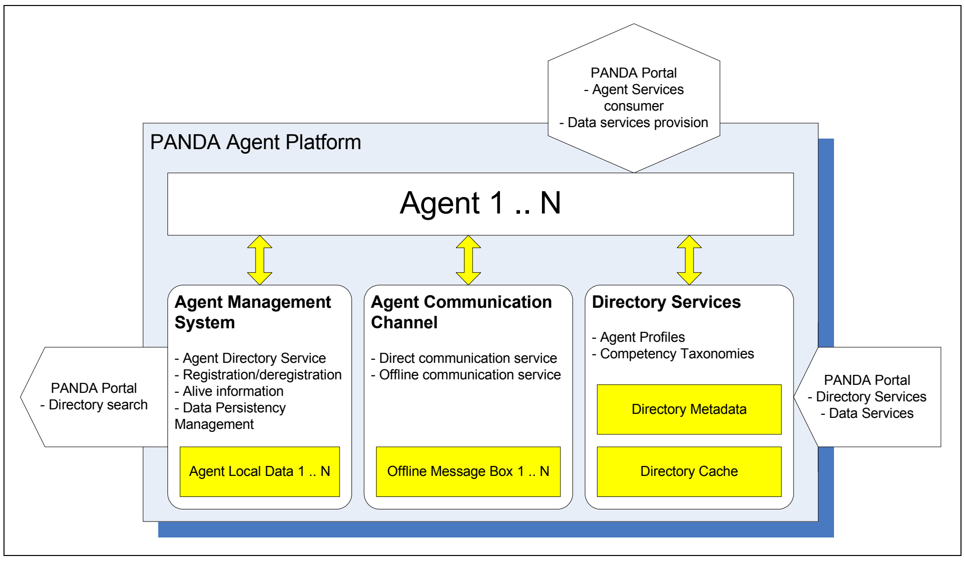


Figure 8 The structure of the PANDA Agent Platform (see online version for colours)



Note: There are also horizontal interactions with the PANDA Portal (transparent to agents) and vertical interactions for the integration of individual agents with other systems of the agent's owner.

The PANDA Partner Agent application is able to act in the community of agents as a coordinator and/or participant at the same time (this means it is able to start RBVO formation or participate in an RBVO formation initiated by another agent). It is empowered by the implementation of the RBVO Formation Protocol, rule engine and set of private business rules adjusted by the owner (human user). For the user interactions, it is equipped with the JAVA graphical user interface. In the case of server hosting, when direct application access is not possible, the agent can be controlled by the remote Graphic User Interface (GUI) using the *PANDA Agent Service*. This service is optionally deployed on each agent and enables one to access agent capabilities remotely. Although the service does not provide all the features like the application GUI (such as the CR editor, business rules editor, various RBVO filtering and sorting, *etc.*), it allows one to create any kind of custom user interface or integration of the agent in the internal business processes and applications of the company. This web service provides the following methods:

- submitCr – enables the submission of a collaboration request to the agent. The method also allows the definition of the set of business parameters for this CR
- getRbvoProposals – provides the list of proposed RBVOs for a particular CR
- getCrStatus – provides the status of a collaboration request
- confirmRbvo – enables the selection of an RVBO and signals the agent to close the collaboration request
- getParticipBizRuleParams – enables acquisition of the list of business rules parameters actually used on the participant's (collaborator's) side

- setParticipBizRuleParams – enables updating of the list of business rules parameters actually used on the participant's (collaborator's) side
- getObtainedNegotiations – enables acquisition of the list of negotiations that the agent participates in
- submitObtainedNegotiation – enables updating of a negotiation (that is in pending or empty status).

To have a complete picture, there is a possibility to integrate the agent with the private company data stores. There is a web service interface designed to enable the *Knowledge Processing Module* of the agent to ask an external service for the creation of proposals for incoming CRs. The idea is to substitute the Drools engine used by agents internally with any proposal creation mechanism. Using this interface, the agent is enabled by the possibility to create collaboration proposals based on real data and standard intracompany processes.

The implementation of the PANDA agent system has been performed in the JAVA programming language. The platform services have been deployed on the TOMCAT server beside the Portal and Central Services. The Partner Agents use native JAVA JAX-WS[6] implementation as the service container. Each agent runs as a separate JAVA application and is able to be deployed 'any-place', where JAVA 1.6SE or a better runtime environment is installed. It needs at least an outgoing internet connection (for web services calls) and optional incoming connection. The distribution (and also updating) of the agents is provided using Java Web Start Technology.[7]

6 Conclusion

The proposed protocol has been designed for RBVO formation, but it is also possible to deploy it to other domains of VOs which employ the concept of SLAs. The protocol allows for reflecting the conditions of real competitive environments as well as negotiation scalability and complexity and is a support for human-assisted negotiation.

The first phase of the protocol focuses on the multiround prenegotiation of the contract conditions between the partners. This phase is finished by a preliminary agreement or a participation refusing/rejection and can be fully or partially automated (agents negotiating on behalf of their owners). The second part contains pairwise multiround bargaining of the agreements. The result of this part is a set of SLAs or participation rejections. The third part is focused on RBVO dissolution and does not offer any special terminating conditions (*e.g.*, penalties, quality of service delivered) or RBVO execution/evolution control.

A possible improvement of the presented RBVO Formation Protocol consists in the adoption of features of the C-CNP Protocol in its decommitment and termination phases. In fact, the proposed RBVO Formation Protocol can be used instead of the contracting phase of the C-CNP. The decommitment phase of C-CNP or (the more complex) renegotiation phase of the RC-CNP should directly address the needs of the evolution (modification/perturbation) phase of the VO life cycle.

The RBVO Formation Protocol has been successfully deployed in the PANDA prototype and tested in the real demonstration of the system. There are about 38 partner agents utilising the protocol deployed on various sites (hosted on the central platform server or distributed on the user partners' servers or PCs in seven different geographical locations); 12 of them are able to play the role of coordinator and thus initiate RBVO formation. The partner agents are able to provide five different services with various constraints like languages, countries, industry domains, ERP module expertise, reputation, price and availability. In the worst case, there is a maximum of 24.3 million potential RBVOs for nonconstrained five-task CR if all of the companies offer all of the services. The RBVO Formation Protocol empowered by the IRAM algorithm and business logic captured by rules provides efficient (semi-)automated cooperation establishment and effective collaboration support.

Acknowledgements

The work is partially funded by the European Commission's FP6 programme within the projects PANDA (contract no. 027169) and ECOLEAD (contract no. FP6 IP 506958), and the Network of Excellence for Innovative Production Machines and Systems (No. 500273-2). It is also supported by the Czech Ministry of Education grant 6840770038.

Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The community is not liable for any use that may be made of the information contained herein.

References

Aubrey, B. (1991) 'Les conditions de possibilités du réseau', in I. Orgogozo (Ed.) *Les paradoxes du management*, Les Editions d'Organisation.

Bíba, J. and Vokřínek, J. (2006) 'Agent contracting and reconfiguration in competitive environments', *Cybernetics and Systems*, Austrian Society for Cybernetics Studies, Vol. 2, pp.527–532.

Bíba, J., Vokřínek, J. and Hodík, J. (2008) 'Renegotiable competitive contract net protocol', Research Report, Prague: CTU FEE, Department of Cybernetics, The Gerstner Laboratory, GL 194/08, ISSN: 1213-3000.

Camarinha-Matos, L.M. and Afsarmanesh, H. (1998) 'Virtual enterprises: life cycle supporting tools and technologies', in A. Molina, J.M. Sánchez and A. Kusiak (Eds.) *Handbook of Life Cycle Engineering: Concepts, Tools and Techniques*, Kluwer Academic Publishers, pp.535–571.

Camarinha-Matos, L.M. and Afsarmanesh, H. (2001) 'Virtual enterprise modeling and support infrastructures: applying multi-agent system approaches', in M. Luck, V. Mařík, O. Štěpánková and R. Trappl (Eds.) *Multi-Agent Systems and Applications*, *9th ECCAI Advanced Course ACAI 2001* and *Agent Link's 3rd European Agent Systems Summer School, EASSS 2001*, Prague, Czech Republic, 2–13 July, Selected Tutorial Papers, Springer, Vol. 2086 of Lecture Notes in Computer Science, pp.335–364.

Capó, J., Lario, F.C. and Ortiz, A. (2004) 'Proposal of an advanced model for supply chain management in construction industries, based on the virtual enterprise', in K-D. Thoben, K.S. Pawar and F. Weber (Eds.) *Proc. of the Tenth Int. Conf. on Information and Computation Economies ICE-2004*, Centre for Concurrent Enterprising, pp.395–402.

Esteban, L.S. (2007) 'Agent communication using web services, a new FIPA message transport service for Jade', *Multiagent System Technologies. 5th German Conference, (MATES-2007)*, 24–26 September, Leipzig, Germany: Springer, Vol. 9.

Faisst, W. (1997) 'Information technology as an enabler of virtual enterprises: a lifecycle-oriented description', *Proceedings of the European Conference on Virtual Enterprises and Networked Solutions*, Paderborn, Germany, April.

Fischer, K., Müller, J.P., Heimig, I. and Scheer, A-W. (1996) 'Intelligent agents in virtual enterprises', *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology (PAAM)*, Blackpool, UK, pp.205–223.

Gruber, M. and Nöster, M. (2005) 'Investigating structural settings of virtual organizations', in K.S. Pawar, F. Weber, K-D. Thoben and B. Katzy (Eds.) *Proc. of the 11th Int. Conf. on Concurrent Enterprising ICE-2005*, Centre for Concurrent Enterprising, June, pp.245–252.

Hodik, J., Vokrinek, J., Biba, J. and Becvar, P. (2007) 'Competencies and profiles management for virtual organizations creation', *Multi-Agent Systems and Applications V, 5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007*, Leipzig, Germany, 25–27 September, Springer-Verlag, ISBN: 978-3-540-75253.

Lomuscio, A.R., Wooldridge, M. and Jennings, N.R. (2003) 'A classification scheme for negotiation in electronic commerce', *Group Decision and Negotiation*, January, Vol. 12, No. 1, pp.31–56.

Mařík, V. and McFarlane, D.C. (2005) 'Industrial adoption of agent-based technologies', *IEEE Intelligent Systems*, Vol. 20, No. 1, pp.27–35.

Molina, A., Flóres, M. and Caballero, D. (1998) 'Virtual enterprises: a Mexican case study', in L.M. Camarinha-Matos, H. Afsarmanesh and V. Mařík (Eds.) *Intelligent Systems for Manufacturing: Multi-Agent Systems and Virtual Organizations, Proceedings of the BASYS – 3rd IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing*, Prague, Czech Republic, Kluwer, Vol. 130 of IFIP Conference Proceedings, August, pp.159–170.

Neubert, R., Langer, O., Görlitz, O. and Benn, W. (2001) 'Virtual enterprises – challenges from a database perspective', *ITVE '01: Proceedings of the IEEE Workshop on Information Technology for Virtual Enterprises*, Washington, DC: IEEE Computer Society, pp.98–106.

Ovcharenko, S., Alibhai, Z., Ng, C., Gruver, W.A. and Sabaz, D. (2006) 'Implementation of a wireless distributed intelligent system', in V. Mařík, W.A. Gruver, M. Pěchouček and L. Přreučil (Eds.) *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, Los Alamitos, CA: IEEE Computer Society, June, pp.207–212.

Overeinder, B.J., Verkaik, P.D. and Brazier, F.M. (2008) 'Web service access management for integration with agent systems', *Proceedings of the 2008 ACM Symposium on Applied Computing SAC '08*, Fortaleza, Ceara, Brazil, 16–20 March, New York, NY: ACM, pp.1854–1860.

Pěchouček, M., Doubek, J., Vokřínek, J. and Rehák, M. (2008) 'Incrementally Refined Acquaintance Model (IRAM) for request based virtual organization formation', *Distributed Human-Machine Systems*, Praha: CTU Publishing House, pp.182–187, ISBN: 978-80-01-04027-0.

Perugini, D., Jarvis, D., Reschke, S. and Gossink, S. (2007) 'Distributed deliberative planning with partial observability: heuristic approaches', *Integration of Knowledge Intensive Multi-Agent Systems 2007*, Waltham, MA: IEEE, pp.407–412, ISBN: 1-4244-0945-4.

Petersen, S.A., Divitini, M. and Matskin, M. (2001) 'An agent-based approach to modelling virtual enterprises', Special issue on 'Enterprise Modeling', *International Journal of Production, Planning and Control*, Vol. 12, No. 3, pp.224–233.

Preece, A. (2001) 'Supporting virtual organisations through knowledge fusion', *International Journal of Intelligent Systems in Accounting, Finance, and Management*, Vol. 10, pp.25–37.

Roberts, R., Svirskas, A. and Matthews, B. (2005) 'Request Based Virtual Organisations (RBVO): an implementation scenario', *Collaborative Networks and their Breeding Environments*, Springer, Vol. 186 of IFIP, pp.17–25.

Rocha, A.P. and Oliveira, E. (1999) 'An electronic market architecture for the formation of virtual enterprises', *PRO-VE 99 IFIP/PRODNET Conference on Infrastructures for Industrial Virtual Enterprises*, October.

Sandholm, T.W. and Lesser, V.R. (2001) 'Leveled commitment contracts and strategic breach', *Games and Economic Behavior*, April–May, Vol. 35, Nos. 1–2, pp.212–270.

Shao, J., Gray, W.A., Fiddian, N.J., Deora, V., Shercliff, G., Stockreisser, P.J., Norman, T.J., *et al.* (2004) 'Supporting formation and operation of virtual organisations in a grid environment', *Proceedings of the UK e-Science All Hands Meeting (AHM)*, pp.376–383.

Smith, R.G. (1980) 'The contract net protocol: high level communication and control in a distributed problem solver', *IEEE Transactions on Computers*, December, Vol. 29, No. 12, pp.1104–1113.

Tagg, R. (2001) 'Workflow in different styles of virtual enterprise', *ITVE '01: Proceedings of the IEEE Workshop on Information Technology for Virtual Enterprises*, Washington, DC: IEEE Computer Society, pp.21–28.

Trienekens, J.J.M., Bouman, J.J. and van der Zwan, M. (2004) 'Specification of service level agreements: problems, principles and practices', *Software Quality Journal*, March, Vol. 12, No. 1, pp.43–57.

Van Wijk, J., Geurts, D. and Bultje, R. (1998) 'Seven steps to virtuality: understanding the virtual organisation processes before designing ICT support', *Objects, Components and the Virtual Enterprise '98. An Interdisciplinary Workshop*, http://www.cs.tcd.ie/ Virtues/ ocve98/ proceedings/ index.html.

Vokřínek, J., Hodík, J., Bíba, J., Vybíhal, J. and Pěchouček, M. (2007) 'Competitive contract net protocol', *LNCS 4362 – SOFSEM 2007: Theory and Practice of Computer Science*, Berlin: Springer-Verlag, pp.656–668.

Wiendahl, H-P. and Lutz, S. (2002) 'Production in networks', *CIRP Annals – Manufacturing Technology*, Vol. 51, pp.573–586.

Wooldridge, M. and Jennings, N.R. (1999) 'The cooperative problem-solving process', *Journal of Logic and Computation*, August, Vol. 9, No. 4, pp.563–592.

Notes

1 The Foundation for Intelligent Physical Agents (FIPA) is an IEEE Computer Society standards organisation that promotes agent-based technology and the interoperability of its standards with other technologies, http://www.fipa.org.

2 http://www.panda-project.com/

3 Enterprise Resource Planning/Customer Relationships Management (ERP/CRM).

4 http://labs.jboss.com/drools/

5 For example, the one of the most used agent platforms JADE provides the web-services integration gateway WSIG for transparent integration of agents and WS, jade.tilab.com/doc/ tutorials/JADE_WSIG_Guide.pdf.

6 See http://java.sun.com/webservices/ for more details.

7 See http://java.sun.com/products/javawebstart/ for more details.

Incrementally Refined Acquaintance Model for Consortia Composition

Jan Doubek, Jiří Vokřínek, Michal Pěchouček, and Martin Rehák

Agent Technology Group, Gerstner Laboratory
Department of Cybernetics, Czech Technical University in Prague
duby.jan@seznam.cz, {vokrinek,pechouc}@labe.felk.cvut.cz

Abstract. This paper presents a specific contracting algorithm that contributes to the process of distributed planning and resource allocation in competitive, semi-trusted environments. The presented contraction algorithm is based on incrementally refined acquaintance models (IRAM) of the actor that provide the right set of approximate knowledge needed for appropriate task decomposition and delegation. This paper reports on empirical evaluation of the IRAM algorithm deployment in consortia formation domain.

1 Introduction

This work focuses on a technique for distributed consortium formation with limited knowledge sharing. The consortia formation is based on a negotiation between independent self-interested providers. The providers can offer several services and the goal is to find the best suitable composition of providers to cover required set of services.

The targeted domain organizes multi-party interaction in the environments that are:

– **non-centralized** and with flat organizational structure [R1] – the existence of a central coordination is minimal and the information about the skills of actors, resource availability, knowledge and goals is distributed,
– **multi-party involvement** [R2] – the final project cannot be implemented in isolation by a single actor, consortium composition can be initiated by several actors simultaneously,
– provides **partial knowledge sharing** [R3] – the actors in the environment are motivated to keep a substantial part of their private planning knowledge and resource availability information undisclosed.

Due to the presented requirements, the consortia cannot be evaluated centrally and the service allocation to the individual providers has to be negotiated. The goal is to minimize the interactions with the providers to the necessary minimum to reduce their private knowledge disclosure and simultaneously ensure the quality of the solution. To fulfill those demands we have designed presented algorithm and acquaintance model representation and provide experimental evaluation.

This algorithm contributes to the process of distributed planning and resource allocation in competitive, semi-trusted environments. The presented algorithm is based on incrementally refined acquaintance models (IRAM) – the model that the actor is maintaining about potential collaborators [1].

2 Problem Statement

The consortium composition can be represented as distributed state-space search through all the potential consortia in the environment with limited information sharing. Let us denote R as a requester agent, A_t as a set of services that R requests to fulfill the task t. Furthermore we have agent P_j, as a provider agent offering a certain set of services A_j^{max}, where $j \in \{1 \ldots n\}$ and

$$F_j(A_j) : \{A_j \subset A_j^{max}\} \tag{1}$$

is pricing function for agent P_j to provide set of services A_j. When asked agent P_j sends back just the price value.

The problem is then to acquire the optimal price

$$c(A_t) = \min \sum_{j=1}^{n} F_j(A_j) \rightleftharpoons \{A_j \subset A_j^{max}; \bigcup_{j \in \{1 \ldots n\}} A_j = A_t\} \tag{2}$$

In this paper we then focus to find optimal vector of sets $(A_1, \ldots, A_n)$ as a decomposition of task t, where the overall price $c(A_t)$ is minimal. Set of all vectors that satisfies task the condition $\bigcup_{j \in \{1 \ldots n\}} A_j = A_t$ will be referenced as a *Deal Space* - DS_t.

In our model we have made several assumptions:

– **Fixed price** – the price of a particular subset of services is fixed during algorithm run.
– **Tasks are independent** – a provider is capable of delivering same services during all negotiation even if he was contracted for some services in the previous tasks.
– **Non-increasing partial price** – a provider constructs a $F_j(A_j)$ as aggregated price from prices for individual services that are hidden to the requester. We assume the individual services price to be non-increasing in reference to increasing $|A_j|$.

3 IRAM-Based Consortium Formation

We have designed a straightforward decomposition mechanism that finds the optimal decomposition given the right objective function and a complete information about provider's resource availabilities. The decomposition algorithm is polynomial and easy to construct (see [2]). Its behavior, however, worsens strongly with lower quality of information about the provider's prices stored in the requestors' acquaintance models (containing a subset of the deal space). The most efficient approach in fully cooperative communities would be if the requestor queries all the providers and reconstructs the deal space for all services provided by all actors prior to computing the optimal a contract.

As this is not possible in the environment compliant with the requirements R1 and R3, the requestor needs to approximate such knowledge with only partially available information. We are proposing *incrementally refined acquaintance model* (IRAM) algorithm for handling partial knowledge sharing and private knowledge disclosure [1].

This approach has been evaluated and compared with the another method of provider prices estimation - the well-known Chebyschev Polynomials approximation method.

3.1 Acquaintance Model

The acquaintance model can have a number of forms [3], [4]. In this particular application the acquaintance model is understood as function that predicts actor responses to a particular *call-for-proposals* (`CFP`) type of message. We represent the *acquaintance model* (*am*) as a mapping from a set of $\mathcal{P}(A_j^{max})$ possible subsets asked from the provider P_j to a 1 dimensional real-value space representing cost $\mathcal{C}$.

$$\mathcal{F}_j^{am}\colon\mathcal{P}(A_j^{max}) \to \mathcal{C} \tag{3}$$

Let us discuss several properties of an acquaintance model. The *fixed point* is such a mapping among the actor, single service and a particular cost that is based on exact information acquired from the communication with the specific actor. In a fixed point as_x^σ

$$\mathcal{F}_j^{am}(as_x^\sigma) = f_j(s_x, |A_j|, pc_j) \text{ where } |A_j| = \sigma \tag{4}$$

where $f_j(s_x, |A_j|, pc_j)$ represents the price contribution of presence of service s_x in A_j to the total price of the entire set $F_j(A_j)$.

Provided that the fixed points of the acquaintance model are collected in a set $\Delta(\mathcal{F}_j^{am})$, we define the *size of the acquaintance model* $\delta(\mathcal{F}_j^{am})$ the amount of the fixed points in the acquaintance model as follows:

$$\delta(\mathcal{F}_j^{am}) = |\Delta(\mathcal{F}_j^{am})|. \tag{5}$$

Various approximation functions have been used in the acquaintance models, e.g. [2]. In our model we have selected the pairwise constant approximation. The **unknown** price of service s_y in subset with size $|A_j|$, equals to the closest bigger known fixed point in means of the size of the containing subset$|A'_j| : s_y \in A'_j$

$$\mathcal{F}_j(s_y, |A_j|, pc_j) = \mathcal{F}_j(s_y, |A'_j|, pc_j) \text{ if } |A_j| \lessdot |A'_j|, \tag{6}$$

provided that the symbol $\lessdot$ represent the smallest bigger value.

The *error of the acquaintance model* - $\epsilon(\mathcal{F}_j^{am})$ - represents how well does the acquaintance model capture real capability of the providers. Error of the acquaintance model is a dual quantity to the *quality of the acquaintance model.* There can be a number of ways how the error can be related to the quality. We only require that with a monotonic increase of quality the error decreases and vice versa.

We represent the error of the acquaintance model as a sum of the differences between the real costs and the information on costs provided by the acquaintance model.

$$\epsilon(\mathcal{F}_j^{am}) = \sum_{p,j} |\mathcal{F}_p^{am}(A_{p,j}) - f_p(A_{p,j}, pc_j)| \tag{7}$$

the index p goes through all possible subsets of A_j, and j goes through all partners.

As said before, the reason why we use the acquaintance models for contracting is that we are motivated by minimizing the unwanted knowledge disclosure during interaction (requirement R3). Each interaction represents disclosure of private information. By `CFP` the actors disclose their inability to perform a task as well as their intention to do so. By a response to `CFP` the agents disclose information about availability of particular resources. It is evident that with rising $\delta(\mathcal{F}_j^{am})$, the acquaintance model is more exact and thus provides better information (i.e. lower $\epsilon(\mathcal{F}_j^{am})$). Better acquaintance model managed to reduce communication (and thus private knowledge disclosure) during the negotiation between the actors. However, bigger $\delta(\mathcal{F}_j^{am})$ (and thus smaller $\epsilon(\mathcal{F}_j^{am})$) required substantial interaction during the acquaintance model construction phase where lots of unwanted information may have been disclosed.

The IRAM algorithm is balancing the size and the quality of the acquaintance models. In order to evaluate performance of the IRAM algorithm we have developed a reference algorithm that is working with a similar acquaintance model, constructed prior negotiation. Both algorithms are based on distributed state-space search using negotiation between actors. As a negotiation protocol, we use the *competitive contract-net protocol* [5], but any protocol that enables iterative contract negotiation can be used.

3.2 IRAM Algorithm

The run of this algorithm for one particular task t is started with the initiation phase. All providers are contacted for every single service and for maximal subset of services from task t. The IRAM model am is constructed using the closest bigger known fix-point approximation (see eq. 6). The model is represented by sets of prices for specific service and pricing function settings (see eq. 1). The price is set blank when is not known, and thus is calculated from other fixed points.

The algorithm constructs the Deal Space and evaluates it with the prices from $\mathcal{F}_j^{am}$. The cheapest consortium $Cons^{best}$ is selected and the providers are requested for appropriate services. Offered prices are then integrated into the IRAM model. The deal space is then reevaluated and the cheapest consortium is selected. If the new consortium is composed from fixed points (represents

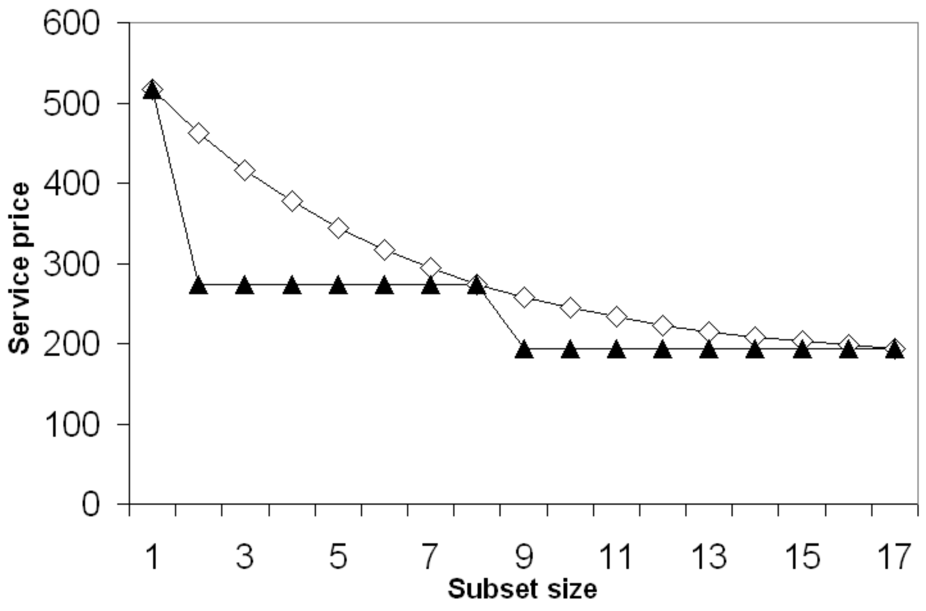


Fig. 1. IRAM model(labeled with triangles) for one service according to s_x, fixed points in 1,8,17. Approximated function is labeled by diamonds.

real price of the consortium – no price approximation), this we understand as optimum. Request for this consortium will lead to the exact same information and due to eq. 6 the algorithm has converge to the optimum. The phases of IRAM can be seen below.

Initialization. It is necessary to know at least two fixed points of acquaintance model for specific service from each provider, for proper functionality of IRAM algorithm. So if the algorithm have not these from previous contracts, it obtains them in the first iteration. Due to this fact the amount of communication is considerably higher regarding to following iterations. Preferably we choose the single service data and maximum provider coverage data (A_j^{max}). Single service provides us data needed for proportional price reconstruction necessary due to aggregated price. And the max coverage data gives us the lowest possible prices from provider needed for approximation.

Iteration Phases. The iteration phases represent processes that follow each other in further negotiation stage.

- Contacting the best known consortium given by acquaintance model
- Updating IRAM model by the received responses
- Reevaluating the acquaintance model
- Sorting the deal space by total consortium price
- Termination condition evaluation

Termination Condition. The algorithm is iterating (contacting and updating model) till the best evaluated consortium consists of fixed points only (e.g. no part of consortium has estimated evaluation)

The steps of IRAM algorithm can be seen in Figure 2.

```
1 Construct deal space $DS_t$.
2 Send CFP($s_i$) for all $s_i \in t$ to all providers $P$.
3 Update $am$ according to received responses.
4 Send CFP($A_p^{max}$) to all providers $P$.
5 Update $am$ according to received set of responses $A_p$.
6 Select $Cons^{best}$ from $DS_t$ evaluated by $am$.
7 If $Cons^{best} \subset \bigcup_{j \in P}(\Delta\mathcal{F}_j^{am})$ then terminate algorithm.
8 Send CFP($A_{p,j}$), where $\bigcup_{j=1\ldots n} A_{p,j} = Cons^{best}$ to providers $1\ldots n$.
9 Goto 5.
```

Fig. 2. IRAM algorithm steps

3.3 Properties of IRAM

The presented IRAM algorithm is sound and complete. The proof of completeness of the algorithm is made through conversion of whole idea to A^* algorithm [6], where the nodes of searched space are individual consortia from DS, and edges represent inclusion (or exclusion) of one provider to a consortium. This representation corresponds to *Coalition Structure graph* [7].

The heuristics of A^* is then based on acquaintance model approximation, all of the nodes are priced particularly by real prices (fixed-points) and by computed prices given by acquaintance model. The price of the consortium $Cons$ is represented by

$$c(Cons) = g(Cons) + h(Cons),$$

$$\text{where } g(Cons) = \sum_{A_j \subset \Delta(\mathcal{F}_j^{am})} (\mathcal{F}_j^{am}(A_j)) = \sum_{A_j \subset \Delta(\mathcal{F}_j^{am})} F_j(A_j),$$

$$\text{and } h(Cons) = \sum_{A_j \subset \mathcal{P}(A_j^{max})/\mathcal{P}(\Delta(\mathcal{F}_j^{am}))} \mathcal{F}^{am}(A_j). \tag{8}$$

The $g(Cons)$ represents price of the of subsets from $Cons$ that is known from previous negotiations (the fixed-points) and $h(Cons)$ is the unknown price of the subsets from $Cons$ estimated by acquaintance model.

The non-increasing individual pricing function assumption causes

$$s_x \in A_{j,1}; s_y \in A_{j,2}; s_y = s_x; |A_{j,1}| \leq |A_{j,2}|$$

$$\Rightarrow f_j(s_x, |A_{j,1}|, pc_j) \geq f_j(s_y, |A_{j,2}|, pc_j) \tag{9}$$

According to eq. 6 and 9 the $h(Cons)$ is always equal or lower then the real price of this subset, so $h(Cons) \leq h^*(Cons)$ and the eq. 8 is admissible heuristics of A^* algorithm. Since the IRAM is based on exploration of the best candidates evaluated by eq. 8 the algorithm provides the features of A^* algorithm [6].

3.4 Reference Algorithm

The presented approach has been empirically validated by comparison with the state-of-the-art algorithm with behavior similar to IRAM algorithm. Generally the IRAM method is used to approximate unknown pricing functions of partners and determine the direction of future negotiation. For the benchmarks purposes we have implemented a reference algorithm based on deployment of known Chebyshev polynomials [8], previously used for modeling of sellers production pricing function.

The *Chebyshev polynomials* exactly Chebyshev polynomials of the first kind are defined as

$$T_n(x) = \cos n \arccos x \tag{10}$$

Due to its orthogonality with respect to the weight $w(x) = (1-x^2)^{-1/2}$ in the interval [-1,1] are widely used for approximation, in most cases they are more effective than Taylor's. The Chebyshev polynomial state can be represented by the recursion formula: $T_{n+1} = 2xT_n(x) - T_{n-1}(x)$, where $T_0 = 1; T_1 = x$. Further information can be obtained in [9].

The approximation itself is made through computing weights (c_k). They represent the contribution of every Chebyshev polynomial to the resulting function. The complete approximation formula is

$$f(x) \approx \sum_{k=0}^{N-1} c_k T_k(x) - \frac{1}{2} c_0 \tag{11}$$

The defining weights are reconstructed from approximated known points $(x_k, f(x_k)), k = 1..M$ with formula

$$c_j = \frac{2}{M} \sum_{k=1}^{M} f(x_k) T_j(x_k) \tag{12}$$

The values of x_k should be mapped to $[-1, 1]$.

3.5 Implementation

As mentioned above, searched approximation method was selected according to initial environmental conditions similar or equal to IRAM's. In case of Chebyshev polynomials the conditions matched exactly. The implementation of the IRAM algorithm was just slightly modified in the field of approximation and all the other interfaces (like communication, consortium construction and evaluation) were left untouched. For a proper explanation of its deployment, let us explain the adaptation of Chebyshev polynomials to the environment. The first two samples (fixed points), needed in general for every approximation, are gathered from the initiation phase of negotiation (same as IRAM). The deal space is then evaluated from Chebyshev approximation of the pricing functions. The final condition is also same as in IRAM. If the same solution is evaluated as the best in two following iterations it is returned as result.

As mentioned previously, the only alteration (from the IRAM algorithm) in the optimal consortium search was the approximation process. Just like in IRAM there is a pricing function model for every particular service from every provider, this model consists of set of weights (eq. 11), the number of weights depends on the number of Chebyshev polynomials used. The number is also correlated with approximation quality (will be further described). The origin of the polynomials is the same for all approximations thus is stored as a constant. The exploiting of the incoming information takes place during the weight computation, which is performed as a result of collecting each new price information.

4 Experiments

The key contribution of the presented paper is in empirical evaluation of the presented algorithm. We will be analyzing the behavior of IRAM in relation to the reference algorithm presented above.

For presenting the contribution of IRAM we construct a market model that contains a set of four providers P, one simple *requester* that requests providers for set of 17 tasks $S = t_1 \dots t_{17}$ using IRAM and reference algorithm for comparison.

In our experiments we randomly generated 4 providers, where everyone of them was capable of delivering 14 services from total 18 services, this setting was chosen due to computation requirements. The max size of *acquaintance model* is $\delta_{max}(\mathcal{F}_j^{am}) = 65532$ possible proposals i.e. *fixed points*, and average *deal space* size in one task is $\langle |DS_t| \rangle = 8777$. The individual pricing functions were randomly generated as follows. We generate uniform distribution set of prices $UP = (bp_1 \dots bp_s)$ in defined range $(200, 600)$ for base price bp of every service from U. Then we create one random value in range $d_j \in (0.6, 1)$ for particular provider p_j that represent the discount in price according to the number of total services asked $|A_j|$. From discount is then computed margin value m_j

$$m_j = 1 + 0.05 * ((1 - d_j)/0.8) * ((1 - d_j)/0.8 + 1)/2. \tag{13}$$

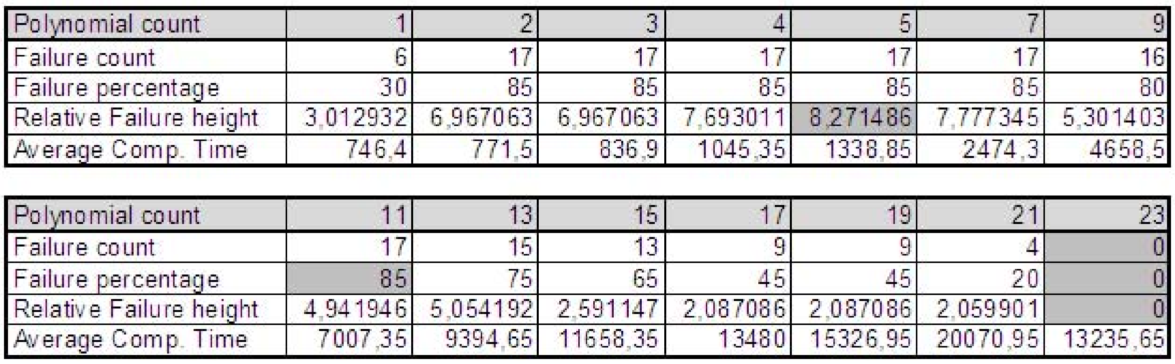
The price $f_{i,j}$ of single service s_i is then derived from base price bp_i and total service asked $|A_j|$ as follows.

$$f_j(s_i) = m_j bp_i d^{|A_j|-1} + 0.5 m_j bp_i \tag{14}$$

Then the pricing function corresponds to eq. 1 Every provider then responds only with this price when is asked for some services.

4.1 Quality of a Model

As shown in [8], the quality of a approximation rises with the number of polynomials used. It can be shown that for a approximation of a polynomial function of a certain degree d the number of Chebyshev polynomials needed for exact approximation is also d. However in our case the pricing functions (eq. 1) are little bit different to be specified exactly like a polynomial. Therefore we run quality tests to find the right Chebyshev polynomial count. The results are shown in figure 3.



Polynomial count	1	2	3	4	5	7	9
Failure count	6	17	17	17	17	17	16
Failure percentage	30	85	85	85	85	85	80
Relative Failure height	3,012932	6,967063	6,967063	7,693011	8,271486	7,777345	5,301403
Average Comp. Time	746,4	771,5	836,9	1045,35	1338,85	2474,3	4658,5

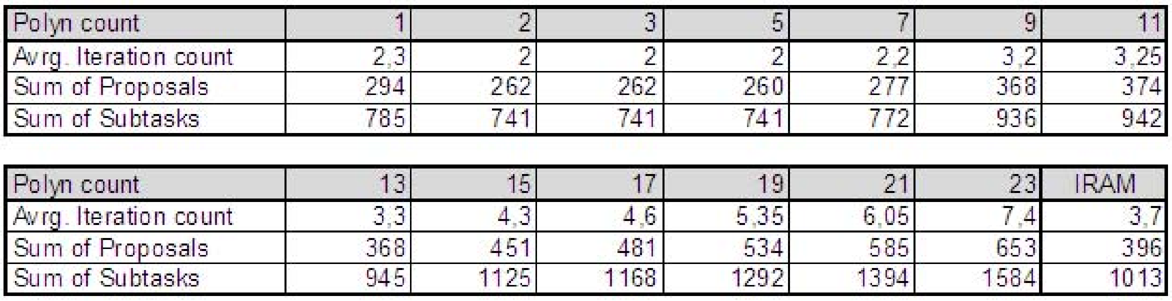
Polynomial count	11	13	15	17	19	21	23
Failure count	17	15	13	9	9	4	0
Failure percentage	85	75	65	45	45	20	0
Relative Failure height	4,941946	5,054192	2,591147	2,087086	2,087086	2,059901	0
Average Comp. Time	7007,35	9394,65	11658,35	13480	15326,95	20070,95	13235,65

Fig. 3. Chebyshev approximation performance with different polynomial count

The scenario of the test is performed by 6 partners with 17 services each from 34 possible services. They are gradually asked for 20 tasks composed of 9 services. The pricing functions specifications are the same as usual.The size of the deal space is then according to the setting 19683. There is a set of tests on the same setting just with different count of Chebyshev polynomials used. Measured variables was *failure count* (finding the wrong optimum), *failure percentage*, *average relative failure* height (to the price of optimum) and *computational time* (on the same hardware setting in ms). The table shows us that the approximation has interesting numbers in one polynomial case, which represents the linear approximation with one straight line. The other results worsens with rising polynomial number until the peak at 11 polynomial for failure count and 5 for average relative failure height. The computational time is rising due to computing of higher and higher powers. By the polynomial count of 23 we can se ideal approximation of our pricing functions represented by 0 failures. So this setting we can use as a proper benchmark for IRAM in this particular environment.

4.2 Benchmarking IRAM vs. Chebyshev

Finally we can unveil the key comparison of Chebyshev method and IRAM. For the first set of tests we simply use the setting described in previous section in order to illustrate major differences. The measured variables are *average Iteration count*, *sum of asked proposals* and *sum of asked services*. The Figure 4 providing measured variables with IRAM's results in the last column, clearly shows that with the same level of quality (zero failures) the shared information (gathered and paid) from partners is 50 % bigger in negotiation than using Chebyshev approximation. To be equal in information requirements we have to accept 85 % chance to have 5 % failure with Chebyshev of the order 11. Just for clear comparison of the *average computational time* for one IRAM negotiation was 859,8 ms comparing to 13235,65 ms of Chebyshev of the order 23. The time results have just relative information value because the implementation of Chebyshev polynomial computation can be slightly upgraded, which is not the key object of this paper. For the next experiments we will use the Chebyshev polynomial of the order 11 and Chebyshev polynomial of the order 23 for benchmarking with IRAM's results.



Polyn count	1	2	3	5	7	9	11
Avrg. Iteration count	2,3	2	2	2	2,2	3,2	3,25
Sum of Proposals	294	262	262	260	277	368	374
Sum of Subtasks	785	741	741	741	772	936	942

Polyn count	13	15	17	19	21	23	IRAM
Avrg. Iteration count	3,3	4,3	4,6	5,35	6,05	7,4	3,7
Sum of Proposals	368	451	481	534	585	653	396
Sum of Subtasks	945	1125	1168	1292	1394	1584	1013

Fig. 4. IRAM vs. Chebyshev comparison

Another key comparison which can be shown on multiple task solving scenarios, is the progress of the harvested data utilization. Due to incrementally better and larger model sizes we can obtain the solution with lower negotiation requirements. Results of this experiments are shown in figures 5 and 6. In the first one there is the number of fixed points requested from the partners in every negotiation. This value has the meaning of shared information among the partners. We can see a notable computational overhead needed for Chebyshev of the order of 23 in the beginning of the negotiations. This trend then gets significantly better than IRAM numbers with increasing number of task solved. This is caused by bigger (thus better) Chebyshev of the order 23 model. The Chebyshev of the order 11 has almost the same data. It was chosen because of its equal data requirements as IRAM. On this particular data we can see progressive IRAM's character in the single task formation problem, i.e. the model is build from scratch. In the first step of the graph IRAM outperforms Chebyshev by 50 %. In unknown environments or in highly dynamic cases this capability become very useful. In the second figure we shown the sizes of each model (see eq. 5) and how this size continually grows with the number of negotiations.

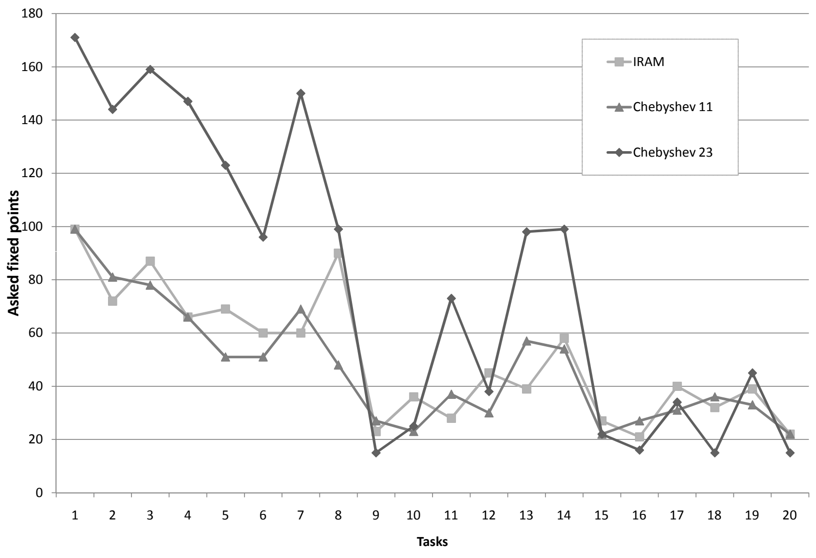


Fig. 5. IRAM vs. Chebyshev 11 and Chebyshev 23 asked fixed points in particular negotiations

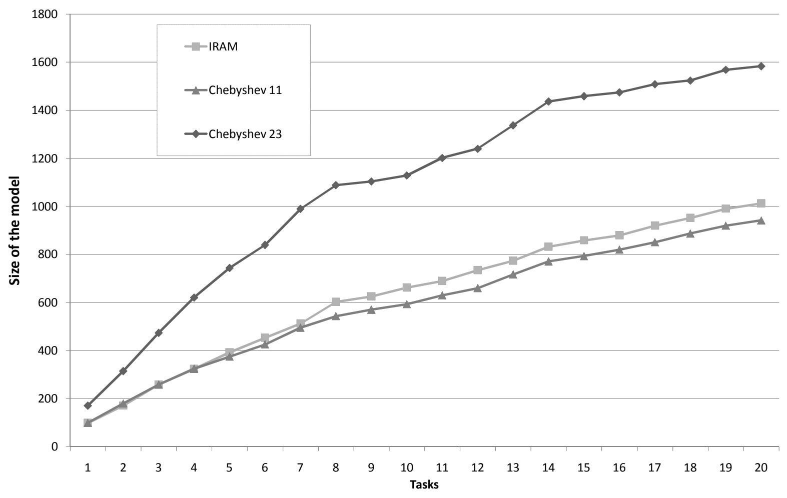


Fig. 6. IRAM vs. Chebyshev 11 and Chebyshev 23 in size of the models

Again, we can see the overhead needed by Chebyshev of the order 23. It comes with the robustness of 23 Chebyshev polynomials and its requirements for initial data to construct the proper approximation. Trends for all of the algorithms are almost the same. With an increasing number of negotiations they tend to converge to certain bound where the models have mapped all interesting areas of partner's pricing functions. In those regions IRAM shows significantly lower need for collected information to provide the best solution.

As we can see, the deployment of IRAM algorithm in this type of domains brings significant improvement in performance, not just in information sharing field but with its simple implementation even in computational requirements.

5 Conclusion

The paper also presents a specific algorithm for distributed task delegation and resource allocation in semi-trusted multi-actor communities - the Incrementally Refined Acquaintance Model (IRAM). This algorithm is based on incrementally maintained social knowledge of the service requestor about the service providers. The novelty of the presented approach is in the fact that social knowledge is used even if very imprecise and it is gradually refined by means of unsuccessful attempts to contract.

The presented IRAM algorithm allows consortia composition with respect to defined requirements, mainly non-centralized approach and minimization of private knowledge disclosure. In environments with a certain degree of dynamics, IRAM's decent information requirement in initiative phase of building the model brings it a significant benefit in comparison with classical approximation approaches (Chebyshev). It can be outperformed in longer sets of negotiations due to more voluminous models representing the opponents.

High dynamics of the environment causes devaluation of acquaintance model and it can lead to incorrect solution. In such dynamic environment, the IRAM algorithm has to start building the model from scratch for every new task. In one deal scenario, the presented IRAM algorithm still provides quick convergence to the optimum with low communication and thus low private knowledge disclosure. Another option is limit the validity of the information obtained and reconstruct part of the model only.

Acknowledgment

The work is funded by the European Commission's FP6 programme within the projects Collaborative Process Automation Support using Service Level Agreements and Intelligent Dynamic Agents in SME clusters (PANDA), IST-No.027169. The research is also partly funded by the Ministry of Education, Youth and Sports of the Czech Republic grant No. MSM 6840770038 and No. 1M0567.

Any opinions expressed in this paper are those of the author(s)/organization and do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

References

1. Pěchouček, M., Mařík, V., Bárta, J.: Role of acquaintance models in agent's private and semi-knowledge disclosure. Knowledge-Based Systems 19, 259–271 (2006)
2. Pěchouček, M., Lerch, O., Bíba, J.: Iterative query-based approach to efficient task decomposition and resource allocation. In: Klusch, M., Rovatsos, M., Payne, T.R. (eds.) CIA 2006. LNCS (LNAI), vol. 4149, pp. 258–272. Springer, Heidelberg (2006)
3. Cao, W., Bian, C.-G., Hartvigsen, G.: Achieving efficient cooperation in a multi-agent system: The twin-base modeling. In: Kandzia, P., Klusch, M. (eds.) Cooperative Information Agents. LNCS (LNAI), vol. 1202, pp. 210–221. Springer, Heidelberg, Heidelberg (1997)
4. Mařík, V., Pěchouček, M., Štěpánková, O.: Social knowledge in multi-agent systems. In: Luck, M., Mařík, V., Štepankova, O. (eds.) Multi-Agent Systems and Applications. LNCS (LNAI). Springer, Heidelberg (2001)
5. Vokřínek, J., Hodík, J., Bíba, J., Vybíhal, J., Pěchouček, M.: Competitive contract net protocol. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 656–668. Springer, Heidelberg (2007)
6. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality af A*. J. ACM 32(3), 505–536 (1985)
7. Sandholm, T.: Distributed Rational Decision Making. In: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 201–258. MIT Press, Cambridge (1999)
8. Saha, S., Biswas, A., Sen, S.: Modeling opponent decision in repeated one-shot negotiations. In: AAMAS 2005: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 397–403. ACM, New York (2005)
9. Rivlin, T.: Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory. Wiley-Interscience, Chichester (1974)

Web Intelligence and Agent Systems: An International Journal 9 (2011) 123–133
DOI 10.3233/WIA-2011-0210
IOS Press

Plan representation and execution in multi-actors scenarios by means of social commitments

Antonín Komenda [*], Jiří Vokřínek and Michal Pěchouček
Gerstner Laboratory - Agent Technology Center, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Technická 2, 16627 Praha 6, Czech Republic
E-mail: {antonin.komenda, jiri.vokrinek, michal.pechoucek}@agents.felk.cvut.cz

Abstract. We present an approach to plan representation in multi-actor scenarios that is suitable for flexible replanning and plan revision purposes in dynamic non-deterministic multi-actor environments. The key idea of the presented approach is in representation of the distributed hierarchical plan by social commitments, as a theoretically studied formalism representing mutual relations among intentions of collaborating agents. The article presents a formal model of a recursive form of commitments and discusses how it can be deployed to a selected hierarchical planning scenario. The decommitment rules definition and their influence on the plan execution robustness and stability is also presented. The approach was verified and evaluated in a simulated environment. The experimental validation confirms the performance, stability, and robustness of the system in complex scenarios.

Keywords: Distributed planning, commitments based planning, social commitment, decommitment rules, non-deterministic environment

1. Introduction

Cooperation between intelligent agents is usually established by means of negotiation resulting in a set of obligations for the participating agents that lead onwards to achievement of a common goal agreed to by the agents. Wooldridge and Jennings formalize the obligations by describing the cooperative problem solving by means of *social commitments* [20] - the agents commit themselves to carry out actions in the social plan leading onwards to achievement of their joint persistent goal [10].

The problem of distributed planning (DP) has been often discussed in the AI planning and multi-agent research communities recently (e.g. [3], [1], [5], [18]). Distributed planning has been viewed as either (*i*) planning for activities and resources allocated among distributed agents, (*ii*) distributed (parallel) computation aimed at plan construction or (*iii*) plan merging activity. The classical work of Durfee [3] divides the planning process into five separate phases: task decomposition, subtask delegation, conflict detection, individual planning and plan merging.

The distributed planning approach proposed in this paper does not provide constructive algorithms for dealing with either of the phases. Instead we propose a special mechanism for plan execution in distributed, multi-actor environment. As such it will affect all the phases of the Durfee's distributed planning architecture.

While classical planning algorithms produce a series of partially ordered actions to be performed by individual actors, we propose an extension of the product (but also an object) of the planning process so that it provides richer information about the context of execution of the specific action. The context shall be particularly targeted towards mutual relation between the actions

[*]Corresponding author.

to be performed by individual actors and shall be used mainly for replanning and plan repair purposes.

The planning problem we are trying to deal with can be informally understood as the task of solving a classical HTN (hierarchical task network) planning problem, defined by an initial partially ordered (causally connected) series of goals, by a set of admissible operators (defined by their preconditions and effects) and methods suggesting a decomposition of a goal into a lower-level planning problem. The plan can be sought for by an individual actor or in collaboration of multiple actors (sharing knowledge and resources). The product of planning is a set of partially ordered terminal actions, allocated to individual actors who agreed to implement the actions under certain circumstances. These circumstances are expressed by specific commitments including the following pieces of information:

- *commitment condition* that may be (*i*) a specific situation in the environment (such as completion of some precondition) or (*ii*) a time interval in which the action is to be implemented no matter what the status of the environment is or (*iii*) a combination of both.
- *decommitment conditions* specifying under which condition the actor is allowed to recommit from the commitment once the task is finished (e.g. notification) or once the task cannot be completed (e.g. a failure)

For long, multi-agent research community has been providing interesting results in the formal work in the field of agents' social commitment, as specific knowledge structures detailing agents individual and mutual commitments. The presented research builds on and extends this work.

The article is an extended and modified version of the original work [7] published on the International Multiconference on Computer Science and Information Technology (IMCSIT/ABC 2008).

The work is structured as follows. In the section 2, the formal description of commitments by Wooldridge is extended, a recurrent notation formalizing the commitments is presented and its use for distributed planning purposes is shown using a scenario for verification. The section 4 gives a brief overview of the most relevant works to our approach. Finally, the last section concludes the paper.

2. Commitments for Planning and Re-planning

As stated in the introduction, a social commitment is a knowledge structure describing an agent's obligation to achieve a specific goal, if a specific condition is made valid and how it can drop the commitment if it cannot be achieved. The commitment does not capture description how the committed goal can be achieved. Individual planning for a goal achievement, plan execution and monitoring is a subject of agents internal reasoning processes and is not represented in the commitment.

In the context of the planning problem defined in the Introduction, we understand the agent's specific goal (to which it commits) as an individual action, a component of the plan, which resulted from the given planning problem. While typical action in a plan contains only a precondition and an effect, in this paper we will describe how its representation can be extended so that the commitment-related information is included.

Michael Wooldridge in [21] defines the commitments formally as follows:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda),\\ &\quad \lambda = \{(\rho_1,\gamma_1),(\rho_2,\gamma_2),\ldots,(\rho_k,\gamma_k)\},\end{aligned} \tag{1}$$

where A denotes a committing actor, ψ is an activation condition, φ is a commitment goal, and λ is a convention. The convention is a set of tuples (ρ,γ) where ρ is a decommitment condition and γ is an inevitable outcome. The convention describes all possible ways how the commitment can be dropped. Generally speaking, the actor A has to transform the world-state in such a way that the φ goal becomes true if ψ holds and any γ has not been made true yet. The actor is allowed to drop the commitment if and only if $\exists i : \rho_i$ which is valid. A decommitment is allowed provided that γ_i is made true. A formal definition in modal logic (working with the models of mental attitudes like Believes, Desires, Intentions, [14], and temporal logic where the operator AG denotes an the inevitability and operator $\curvearrowleft$ denotes the temporal until) follows as defined in [21]:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda) \equiv\\ &\quad ((\mathsf{Bel}\ A\ \psi) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \varphi)\\ &\qquad \wedge(((\mathsf{Bel}\ A\ \rho_1) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \gamma_1))) \curvearrowleft \gamma_1)\\ &\qquad \ldots\\ &\qquad \wedge(((\mathsf{Bel}\ A\ \rho_k) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \gamma_k))) \curvearrowleft \gamma_k)\\ &\quad) \curvearrowleft \bigvee_i \gamma_i).\end{aligned} \tag{2}$$

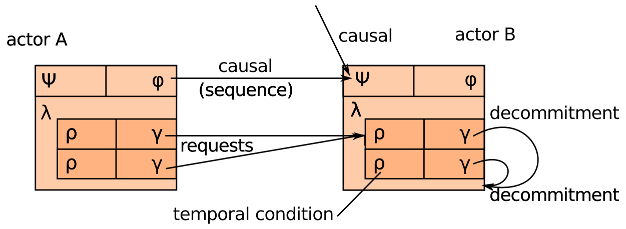


Fig. 1. Commitments and bindings - the actor A's commitment influences the actor B's commitment using the causal (sequential) link, the link is described using the ψ and φ clauses (e.g. $\psi =$ `building-is-ready(B)` and $\varphi =$ `ready(B)`). The actor B's commitment is influenced by external causality too. The actor B's commitment can be decommitted in two cases: either the *temporal condition* ρ becomes true or one of the actor A's rules *requests* the decommitting. The decommitment request is triggered by one of the actor A's ρ conditions.

This definition is used in a declarative way. Provided that whatever the agent does during a specific behavior run complies with the above defined commitment, the expression 2 is valid throughout the whole duration of the run.

One of the goals of the research described in this paper was to provide a formalism for networked commitments to be used for replanning. As clearly stated in the introduction, the commitment conditions can represent variable bindings among preconditions and effects of the individual commitments achieved either by monitoring the environment status or by inter-agent communication (e.g. reception of a specific trigger message). Such representation would be very inflexible in practical applications as it would either need the agents to do nothing and wait for an inhibiting event to happen or risk that once an inhibiting event happens the agent will be busy performing other commitments. Therefore the agents may want to engage in booking and the commitment's precondition would contain fixed time when the commitment is supposed to be adopted. The most flexible approach would be a combination of both - inhibition event and preliminary booked time window, specifying when the inhibiting event is likely to happen. Let us assume that this is the case in the remainder of the paper.

In the distributed plan execution a failure may occur. The indirect impact of this failure may be e.g. a situation where the arranged inhibition event will not happen in the preliminary booked time window. Such occurrence may invoke replanning and allow some agents to e.g. drop unnecessary commitments. This is the reason why the commitments shall not be linked one with other not only via preconditions but also by means of variable bindings among individual agent's decommitment rules. Using these bindings, we can describe the causal sequentiality of the commitments and requests for particular decommitments (Figure 1).

While we will be generalizing on the process of decommitment later in the paper, let us work for now with the specific particular decommitment case suggested in the previous paragraph. Let us assume one agent A forcing decommitment of the other agent's B commitment by means of setting a value of a variable contained in the other agent's commitment. The agent A contains a commitment with a decommitment rule in the form $\langle \rho, v\rangle$ and the agent B contains a commitment with a decommitment rule in the form $\langle v, \mathtt{decommit}(B)\rangle \in \lambda_A$.The request is started by ρ precondition of the actor A (e.g. decommitting the A's commitment). Thus the actor A intends to make the variable v valid. This causes the agent B to intend to decommit by intending the variable `decommit`(B) to be valid (see Figure 1).

This clear example uncovers two needed extensions of the classical social commitment model: (*i*) recurrence of the commitment form – enabling a possibility to disable (decommit) a decommitment request and (*ii*) explicit termination condition – describing termination without any intentional part.

2.1. *Commitment Recurrent Form*

The original Wooldridge definition of a commitment makes a clear distinction between the commitment subject (φ) and the set of mini-goals in the commitment convention (γ). While there is a mechanism for the agent to drop φ, a once adopted mini-goal γ cannot be decommitted. Due to high dynamism and uncertainty of the target scenario, we assume the replanning and plan repair mechanisms to be substantially more complex. We require that the mechanism would allow the agent to try out several different decommitment alternatives, based on the current properties of the environment. The set λ allows listing various different decommitment rules, while no mecha-

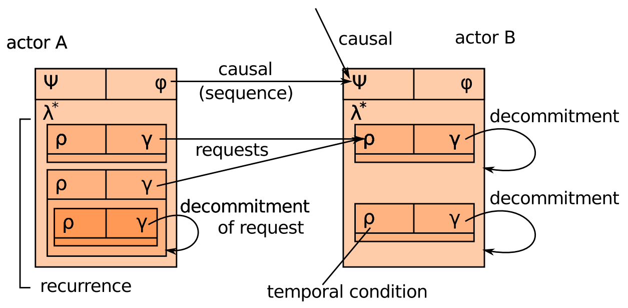


Fig. 2. Commitment and its λ^* commitments (Figure 1). is extended by one *decommitment of request* which can be decommitted if the most inner ρ condition becomes true. Decommitting of the request causes the actor B's commitment cannot be decommitted by the actor A's convention goal any more. Here the recurrent form enables the nesting of the inner commitment.

nism have been specified how different decommitment alternatives are tried out.

That is why we propose generalization of the commitment so that each goal in the commitment structure can be treated equally. Let us introduce the recurrent form of a commitment:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda^*), \lambda^* = \\ &\quad\{(\mathsf{Commit}\ x_1\ \rho_1\ \gamma_1\ \lambda_1^*), \\ &\quad(\mathsf{Commit}\ x_2\ \rho_2\ \gamma_2\ \lambda_2^*), \ldots, \\ &\quad(\mathsf{Commit}\ x_k\ \rho_k\ \gamma_k\ \lambda_k^*)\},\end{aligned} \tag{3}$$

which enables the nesting of the commitments (Figure 2).

The formula 3 extends the definition in 2 not only by inclusion of a set of decommitment rules in each of the individual decommitment rules. It also allows the newly adopted commitments to be assigned to different actors. The delegation kind of decommitment between two agents A and B would have the following form:

$$(\mathsf{Commit}\ A\ \psi\ \varphi\ \{(\mathsf{Commit}\ B\ \rho\ \varphi\ \emptyset)\}), \tag{4}$$

representing that agent A can drop the commitment towards φ provided that ρ is valid and provided that B accepts a commitment towards φ on A's behalf.

This form is very expressive in the sense of the description of exceptional states. It allows us to have a branched chain of individual nested commitments for each individual situation. The recurrent nature allows us to describe an arbitrarily complex protocol using only one knowledge base structure. The definition of the commitment in the recursive form simplifies to:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda^*) \equiv \\ &((\mathsf{Bel}\ A\ \psi) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \varphi) \wedge \bigwedge_j \lambda_j^*) \curvearrowleft \bigvee_i \gamma_i),\end{aligned} \tag{5}$$

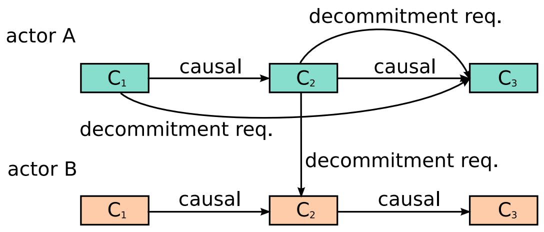


Fig. 3. Commitment graph – the *causal* links define the sequentiality of the commitments of each actor. The commitment C_3 of the actor A can be decommitted by both C_1 and C_2 commitments. The C_2 commitment of the actor B can be decommitted by actor A using the decommitment request.

where the $\bigwedge_j \lambda_j^*$ part acts as the convention set in the original definition in formula 2.

With the help of the proposed form, we can transform the set of mutually interconnected commitments into a graph notation. The formalization is convenient especially due to a broad field of well-known principles and algorithms (e.g. dead-lock detecting, known complexity, extensive transformations and others). Additionally, we will propose a violation solving algorithm (in Section 2.3), based on graph representation of the commitments.

Vertices of the *commitment graph* (Figure 3) represent particular commitments and edges describe intercommitment bindings (causality and decommitment requests). Formally, the commitment graph is defined as follows:

$$\begin{aligned}&\vec{G} = (V, E), \\ &M : V(\vec{G}) \to \mathcal{P}((\mathsf{Commit})), \\ &W : E(\vec{G}) \to \mathcal{P}((\mathsf{Commit})) \times \mathcal{P}((\mathsf{Commit})),\end{aligned} \tag{6}$$

where $\vec{G}$ is an oriented graph with vertices V and edges E. Additionally, the functions M and W map the commitments and their bindings to the vertices and edges. The set $\mathcal{P}((\mathsf{Commit}))$ stands for all possible commitments.

2.2. Decommitment Rules

We require the agents carrying out intelligent planning and re-planning by means of social commitments to be able to perform at least basic reasoning about the decommitment rules attached to the particular commitments. This is needed at the time of re-planning, when an agent needs to decide which decommitment rule (i.e. a new commitment) to adopt, provided that conditions for more than one of them are satisfied. Similarly, agents, when they negotiate about who will accept which commitment, shall be able to analyze not only properties of the goal and costs associated with the goal completion process but also the various decommitment rules when considering the likelihood of the particular failure to occur. Ideally, the agent shall be able to estimate costs of each decommitment rule. However, with the lack of information about the dynamics of the environment, we will be only able to partially order the decommitment rules.

Before analyzing the decommitment rules, we have to describe a basic decommitment frame, which is based on the type of the base commitment. The types of the commitment mutuality are the following:

- *Individual commitments* (IC) - commitments that do not involve other agent than the agent itself. These commitments shall be used if the impact of a failure within the multi-agent community shall be minimized. Individual commitments shall represent several other ways how an agent can accomplish a given task.
- *Minimal social commitment* (MSC) - is the classical type of decommitment, where the agent is required to notify the members of the team about its inability to achieve the commitment.
- *Joint commitments* (JC) - these commitments provides mutually linked commitments (of several agents) via decommitment rules. In a replanning situation the joint commitments proactively assure that the cost of the failure is minimized. An example of the use of a joint commitment is a decommitment of another agent's linked commitment as explained in the Section 2

Starting with the IC, each decommitment rule set (corresponding to Wooldridge's commitment convention) must contain two basic rules, which ensure the rationality of the agent's decision making process. These rules are based on the definition of the *open-minded commitment* defined in [21]:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \varphi) \equiv \\ &\quad \mathsf{AG}((\mathsf{Int}\ A\ \varphi) \curvearrowleft ((\mathsf{Bel}\ A\ \varphi) \vee \neg(\mathsf{Bel}\ A\ \mathsf{EF}\varphi))),\end{aligned} \tag{7}$$

where the operator EF denotes future possibility. Thus in each and every commitment the initial decommitment in the λ^* set should be the success triggering commitment

$$(\mathsf{Commit}\ A\ false\ (\mathsf{Bel}\ A\ \varphi)\ \emptyset) \tag{8}$$

and the λ^* set should be always closed with a fail-safe trigger turning a violated commitment eventually off

$$(\mathsf{Commit}\ A\ false\ \neg(\mathsf{Bel}\ A\ \mathsf{EF}\varphi)\ \emptyset), \tag{9}$$

which is used provided that no other decommitment rule can be used and the parent commitment became unrealizable.

The triggering decommitment condition ρ in rules (8) and (9) cannot become true (as $\rho = false$), which means the agent will never intend the decommitment outcome γ according to the definition (5)

$$\begin{aligned}&((\mathsf{Bel}\ A\ \rho) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \gamma))) \curvearrowleft \gamma \\ &(false \quad\ \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \gamma))) \curvearrowleft \gamma,\end{aligned} \tag{10}$$

nevertheless the rule can drop the commitment using the drop-out part

$$\bigvee_i \gamma_i. \tag{11}$$

This principle can be used for any drop-out rule with no need for explicit intention.

Let us introduce three different types of decommitment rules based on the definition in [17]:

- *Full decommitment* (Fd) - the basic decommitment strategy is dropping the commitment. Under defined circumstances the agent is completely released from the commitment. The full decommitment decommits the original commitment if and only if the commitment goal φ is unrealizable.
- *Delegation* (D) - by using this type of commitments the agent shall be able to find another agent who will be able to complete its commitment

on the original agent's behalf. It is possible that such a commitment will contain unbound variables representing the need to search for an agent suitable for delegation. Delegation decommits the original commitment if and only if the commitment goal φ is unrealizable and the new commitment on the other agent's side is formed.

- *Relaxation* (R) - is a special decommitment, where the original commitment is replaced with a new commitment with a relaxed condition and/or goal. The new commitment must be consistent with all other bound commitments. Provided that the bound commitment is of another agent, the relaxation must be negotiated. The agent being addressed tries to fit the requested relaxed commitment into its knowledge base and eventually use some other decommitment rules of other commitments to change it and fulfill the request. Relaxation decommits the original commitment if and only if the commitment goal φ is unrealizable, the negotiated relaxation conditions hold and the relaxed commitment is formed.

The ordering and presence of particular decommitment rules between the two basic decommitments (success and violation) has a non-trivial impact on the robustness of the execution (especially in an overloaded system) as shown in [17]. The best results for heavily stressed systems can be reached by the ordered set of (D, R, Fd) decommitment rules. On the other hand, the best set for more vacant systems seems to be (R, D, Fd).

For commitments of the MCS type, the rule set has to contain a notification sub-commitment which informs other agents about the state of the parent commitment. The most complex commitments (JC) contain arbitrary number of mutual sub-commitments, which can act as multi-agent coordination messages, synchronization protocols, joint actions and others.

2.3. *Violation Solving Algorithm*

A graph notation can be used to describe the process of successive solving of exceptional states. The process is based on traversing through the commitment graph. The traversing starts with the first violated commitment. One of the decommitment rules is triggered (depending on the violation type). As the decommitment rule is a commitment it invokes an intention of the agent to terminate the commitment. In the case that the intention is a decommitment request, the process passes on the requested commitment (decommitment rule respectively) and starts one of the decommitment rules on the side of the requested commitment. Provided that the decommitment rule terminates the commitment without a need to request other decommitments, the process ends here and the violation is fixed.

The algorithm written in BDI pseudocode follows:

```
Input: Vertex $v$ of a commitment graph $\vec{G}$
  representing violated commitment $C$.
Output: Updated graph with solved violation.
function solve($v$, $C$, $\vec{G}$) begin
  $D$ := find appropriate decommitment rule in $C$
  run $D$ begin
    $(\mathsf{Int}\ A\ \gamma(D)) \Rightarrow$
    if $\gamma(D)$ is request then
      solve($M^{-1}(\gamma(D)(M(v)))$, $\gamma(D)(C)$, $\vec{G}$)
    else
      fix subgraph induced by vertex $v$
        from graph $\vec{G}$
    end
  end
end
```

3. Evaluation

The approach presented in this article has been evaluated in a realistic simulation scenario for distributed planning and coordination in non-deterministic environments [9], [8], [19] emphasizing mixed-initiative planning and decision making. Figure 4 shows the scenario takes place on an island inspired by the Pacifica Suite of Scenarios[1]. (Pacifica domain for DARPA planning initiative [12], [13] and other experiments). The Pacifica scenarios adopt the concept *Go places, do things*, a tasking base that allows a range of missions to be designed for experiments. On the island, there are cities and a network of roads connecting them, but off-road movement is also allowed. There are also several seaports and airports. Scenario actors are of various types: ground units, armored units, aerial or sea units, civilian and hostile units (see Figure 4).

The scenario takes place in a hostile environment with limited information visibility and sharing. Due to this, the environment provides non-deterministic behavior from the point of view of a single unit. There are heterogeneous independent self-interested units in

[1]http://www.aiai.ed.ac.uk/oplan/pacifica

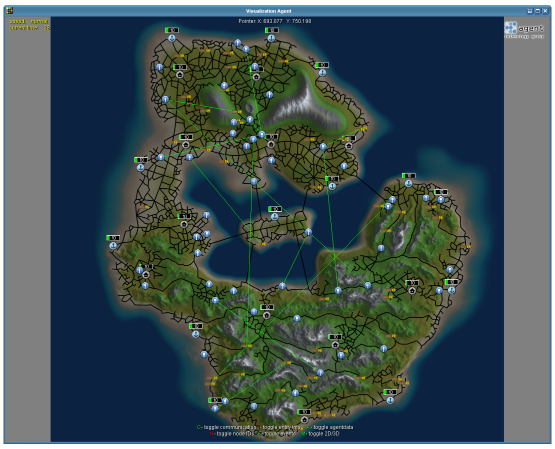


Fig. 4. Scenario island screenshot

the scenario that commit to the shared/joint goals. To fulfill the desired strategic mission (designated in form of high-level strategic tasks by the human operators) in such an environment, the units provide complex cooperative actions on several levels of planning and control.

The aim is to fulfill strategic goals defined by mid- and long-term planners on the strategic level (fulfilling the mission) for each type of unit. The generation of strategic plans is provided by a set of *commanders* that are responsible for each type of *field units* - ground, aerial, sea and armored. The number and specialization of commanders reflects the desired scenario setting. The field units are dedicated to a particular commander and receive strategic goals from it. The hierarchical structure of the tactical planners then creates tactical plans for each field unit (tactical planners are part of each unit's tactical layer). Tactical plans are confronted with the developed multi-agent simulation and adapted to the actual feedback provided by the simulation in real-time. Execution of the plan of the individual unit is simulated and integrated with the environment feedback from the simulation engine.

There are ground units which serve as *Transporters* (can provide faster transportation of other unit(s), material or civilians), *Construction units* (can repair damages or assemble/disassemble stationary units) and *Medical units* (provide medical care for other units or some rescue operations). *Armored units* are used for protection of other units or securing an area or a convoy. *Aerial units* – UAVs with an extended visibility range and *Sea units* for transportation over the sea.

The problem of automated planning in such a world can be described by several more or less separable problems which need to be solved in order to be able to plan in a dynamic non-deterministic environment. The overview of the problems follows:

– **Distributed planning** – Planning in such an environment is realizable only as a distributed process. This affirmation is supported by several facts: the objects of planning are naturally distributed in the world, the robustness of the planning process is a key issue, and finally, each entity has to hold its own private knowledge of its capabilities in the form of a planning domain.
– **Distributed resource allocation** – An integral part of the planning process is resource allocation both of entities acting in the world and of static resources. The allocation process must be appropriately integrated with the planning system and similarly, the planning process has to be robust with respect to the aforementioned constraints of the environment.
– **Distributed plan execution and synchronization** – The distributed plan consisting of several personal plans has to be executed by the entities. The plan has to be robust enough to be able to minimize its volatility and does not need to be completely replanned in case of any non-determined effect.

Planning and control of activities of individual units and actors in the scenario is loosely structured into three levels of detail. We recognize several layers of coordination and control:

– **Strategic layer:** The actors use aggregated meta-information from the tactical layer. This layer provides an overall strategic plan for middle- and long-term horizon. High-level planning and peer-to-peer coordination among the actors is possible (while non-transparent to the tactical level).
– **Tactical layer:** On this layer, the units use aggregated information from the individual layer, the information obtained through communication with each other and the information obtained from the strategic layer. The units and actors use classical planning and cooperation methods and can create new goals or adapt the goals received from the strategic level.
– **Individual layer:** On this layer, the units should perform reactive behavior based on the obtained information and current goals.

The suggested coordination is hierarchical with respect to the type of unit, area of operation and visibility. Three-layer architecture enables to separate middle- and long-term strategic planners from the real-time planning and control on the tactical and individual level. The strategic planner can utilize advanced planning methods using aggregated meta-data from the whole system. On the other hand, the tactical planner has to provide real-time response and it uses limited information provided by an individual layer of the respective unit. On the tactical level, local cooperation and information sharing of the field units is provided. The local cooperation include algorithms e.g., formation planning and holding, rendezvous point designation, target selection and others.

Each agents is using all or a subset of all three layers. Each layer produces particular commitments and these commitments define the plan. The commitments on the individual layer describe which actions should be executed (simply by defining that $\varphi = a$, where a is an action). Moreover, the commitments on all layers describe possible success and decommitment rules (including delegation policies and relaxation intervals). As each layer works with its own level of abstractions, the commitments for each layer can vary. Additionally, the commitments are inter-linked always only between two neighbouring layers. The joint commitments describe mutual commitments of two or more agents (typically used for plan synchronization).

The strategic layer uses the HTN I-X planner [15] and a distributed resource-allocation algorithm. The planner uses an abstract sub-domain derived from the scenario domain and produces an abstract plan. This plan is instantiated using negotiation about the resources (Figure 5). The negotiation is based on well-known contract-net protocol. The initiator agent calls for proposals and other agents reply with their offers valuated by length of the sub-plans. The initiator picks the most suitable (shortest) sub-plan and informs the offering agent. This agent, based on the acceptance, commits itself to the sub-plan goal(s) and thus helps the initiator agent to fulfill its plan. The process is recursive and produces dynamically agent hierarchy necessary for the fulfillment of the top-most goal.

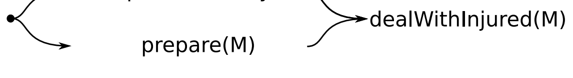


Fig. 5. Instantiated strategic plan - the medic unit M was requested by the commander agent to fulfill a task: deal with the injured in city A, and it negotiated the transport with the transport unit T.

The instantiated plan is converted into commitments (Figure 6). The conversion process creates a commitment according to the particular plan action ($\varphi = a$) and according to forward causality links of the plan.

The commitments of the tactical layer are based on strategic commitments. The layer uses negotiation to form the most suitable mutual commitments. The constraints for negotiation respect the particular needs of the agents. The tactical commitments also define decommitments to the strategic layer and they can additionally refine strategic commitments too. They are much more refined than the strategic commitment in the sense of spatio-temporal constraints, and particular world states. The tactical commitments are most enriched by the λ^* commitments. Thus, the most important part of the decommitting / replanning process is done by this layer.

An example of the tactical negotiation can be: A transport unit T is planning the tactical commitment $moveto(l_1, l_2)$, it can find out it needs support from another unit. In this case, a negotiation process must find an appropriate support unit S_p that proposes the most complying commitment (e.g. in terms of temporal constraints). If such a unit is found the JC is established, planned, and connected to other commitments in the knowledge base.

And finally, the individual layer plans commitments for later execution. These commitments copy the tactical commitments, but some of these can be omitted (e.g. $atPosition$ in the Figure 6). Each individual commitment contains a decommitment request only to its parent commitment (from the tactical layer).

During the execution of the plan the commitments are processed. The commitment can evolve (Section 2.2) according to the plan or due to unexpected environment interactions. The stability and performance of the various types of decommitment rules and their combinations has been studied in [17]. The experimental evaluation proved significant improvement of plan execution performance and stability when using particular combinations of decommitment rules. The success rate of commitment execution and available resources utilization significantly increases with the size of the decommitment rule set. A detailed analysis of the relaxation decommitment rule and its utilization for the presented scenario can be found in [16].

4. Related Work

Formalization of commitments has been extensively studied in the past using various formalisms, most of

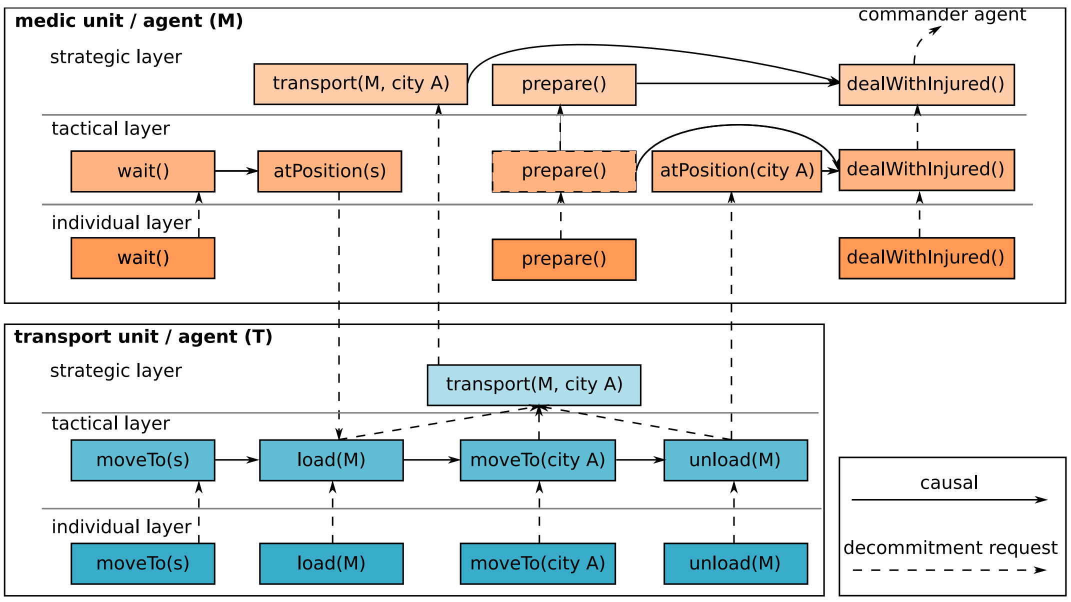


Fig. 6. Commitment bindings of multi-layer architecture for two units – the medic unit M is committed to fulfill a task: deal with the injured in city A, and the transport unit T is committed to transport the medic unit M to city A. The figure shows the directions of the potential decommitment propagation among the layers of actors.

all building on and extending the BDI framework when describing obligations the agents adopt. Fasli [6] distinguishes two classes of obligations - general and relativized - and adoption of a social commitment by an agent is described as an adoption of a role. Thus, the agent promises its coherence with a (behavior) norm defined by the commitment. The framework extends BDI into a many-sorted first order modal logics to add concepts of obligations, roles and social commitments while it also uses branching temporal components from Computational Tree Logics (CTL) [4]. Besides strategies for adoption of social commitments by the agents the framework also defines strategies regarding conditions for a successful de-commitment from the agent's obligations.

Another formal representation of commitments considering temporal account has been introduced in [11]. CTL [4] has been extended to capture features not being usually considered in common approaches (but relevant for realistic environments), namely time intervals considered in commitments satisfaction, "maintenance" manner of commitments next to "achieve" manner of commitments and vague specification of time. Commitments have been formally defined using Backus-Naur Form as an n-tuple (Commit id, x, y, p) where the commitment identified uniquely by its id and the interpretation is that x commits to y to make the condition p become true. The formal framework uses event calculus and defines operations $create(x, C)$, $cancel(x, C)$, $release(y, C)$, $assign(y, z, C)$, $delegate(x, z, C)$ and commitment $discharge(x, C)$ above the commitments as well as new predicates $satisfied(C)$ and $breached(C)$ which evaluate the status of the commitments. The past is considered linear while the future is branching. When created, the commitment is neither satisfied nor breached (the satisfaction of commitments is applied three-value logics). A commitment once satisfied or breached remains satisfied or breached once and for ever since the time.

Evolution of commitments in teamwork has been studied by Dunin-Keplicz [2]. Teamwork is explicitely represented using BDI framework by introducing a concept of a collective intention resulting in a plan-based collective commitment established within a group of agents adopting it. The teamwork consists of four consecutive stages - *(i) potential recognition*, *(ii) team formation*, *(iii) plan formation* and *(iv) team action*. The collective commitment based on a social plan (the collective intention) splits into sub-actions expressed as pairwise social commitments between agents. Establishment of the collective commitment consists in a consecutive execution of social actions defined at the particular stages: *(i)* `potential-recognition` $\rightarrow$ *(ii)* `team-formation` $\rightarrow$ *(iii)* `plan-generation` executed as `task-division` → `means-end-analysis` → `action-allocation` and *(iv)* `team-action` implemented as execution of respective actions allocated to each agent in the former stage. Naturally, the above-mentioned social actions are hierarchically bound from the first to the last stage. Dynamically evolving environment may cause unfeasibility of the allocated actions during the team action which results in a need for evolution of the collective commitment accordingly. In such a case, the maintenance of the collective commitment is achieved by invoking reconfiguration at the `action-allocation` level progressing upwards to higher levels of the hierarchy of social actions, possibly up to the `potential-recognition`. Finally, the collective commitment is adapted (another potential for the teamwork recognized) or dropped. The hierarchical manner of the reconfiguration allows for minimization of changes necessary to perform in order to adapt the collective commitment. The communication necessary for the reconfiguration is explicitly involved and formalized in the framework. Adaptation of the commitment is motivated by persistency of the joint intention which differs given a chosen intention strategy (*blind*, *single-minded* and *open-minded*). For the sake of not making the presented multi-modal logical framework even more complex and less tractable, temporal aspects of the cooperation are assumed to be expressed in a procedural way rather than by employing temporal and dynamic elements among the modalities used.

5. Conclusion

This article dealt with the problem of distributed planning used for replanning and plan repair processes. The classical work on commitments has been extended towards commitment recurrence for flexible and more expressive representation of replanning alternatives. Similarly, the termination condition has been defined as a specific type of commitment. The various types of commitments were classified according to the impact they may have on the other collaborating actors. This classification enables the agents to make the right decisions during the decommitment process.

Based on the social commitment representation we have defined and formalized basic decommitment rules for open-minded commitments representation. The evaluation of the presented approach has been made on an experimental realistic scenario and deployed in a multi-agent system for commitment-based distributed planning. The combinations of particular rules provide a complex decommitment behavior and significantly improve plan execution performance and stability. The success rate of commitment execution and available resources utilization significantly increases with the size of the decommitment rule set. Different rule combinations have to be chosen for different application scenarios.

Acknowledgement

The presented effort was sponsored by Czech Ministry of Education grant 6840770038, supported by the European Research Office of the US Army under grant number W911NF-08-1-0041, and by Internal Grant of Czech Technical University under number CTU0802913.

The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

References

[1] M. E. DesJardins and M. J. Wolverton. Coordinating a distributed planning system. AI Magazine, 20(4):45-53, 1999.

[2] B. Dunin-Keplicz and R. Verbrugge. Evolution of collective commitment during teamwork. Fundamenta Informaticae, vol. 56, no. 4, pp. 563-592, August 2003.

[3] E. H. Durfee. Distributed problem solving and planning. In G. Weis, editor, A Modern Approach to Distributed Artificial Intelligence, chapter 3. The MIT Press, San Francisco, CA, 1999.

[4] E. Emerson and J. Srinivasan. Branching time temporal logic. In Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop, ser. LNAI, vol. 354. Springer- Verlag, 1988, pp. 123-172.

[5] E. Ephrati and J. S. Rosenschein. A heuristic technique for multiagent planning. Annals of Mathematics and Artificial Intelligence, 20(1-4):13-67, 1997.

[6] M. Fasli. On commitments, roles, and obligations. In CEEMAS 2001, ser. LNAI, B. Dunin-Keplicz and E. Nawarecki, Eds., vol. 2296. Springer-Verlag Berlin Heidelberg, 2002, pp. 93-102.

[7] A. Komenda, M. Pechoucek, J. Biba, and J. Vokrinek. Planning and re-planning in multi-actors scenarios by means of social commitments. In Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT/ABC 2008), volume 3, pages 39-45. IEEE, october 2008.

[8] A. Komenda, J. Vokrinek, M. Pechoucek, G. Wickler, J. Dalton and A. Tate. I-Globe: Distributed Planning and Coordina-

tion of Team-oriented Activities. In proceedings of Knowledge Systems for Coalition Operations 2009.

[9] A. Komenda, J. Vokrinek, M. Pechoucek, G. Wickler, J. Dalton, A. Tate and D. Pavlicek. Distributed Planning and Coordination in Non-deterministic Environments. Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10-15, 2009, Budapest, Hungary, pp. 1401-1402.

[10] H. Levesque, P. Cohen, and J. Nunes. On acting together. In AAAI-90 Proceedings, Eighth National Conference on Artiffcial Intelligence, volume 2, pages 94-99, Cambridge, MA, USA, 1990. MIT Press.

[11] A. Mallya, P. Yolum, and M. Singh. Resolving commitments among autonomous agents. In Advances in Agent Communication. International Workshop on Agent Communication Languages, ACL 2003, ser. LNCS, F. Dignum, Ed., vol. 2922. Springer-Verlag, Berlin, Germany, 2003, pp. 166-82.

[12] G. A. Reece and A. Tate. The pacifica neo scenario. technical report arpa-rl/o-plan2/tr/3. Technical report, Artificial Intelligence Applications Institute, University of Edinburgh, Scotland, March 1993.

[13] G. A. Reece, A. Tate, D. I. Brown, M. Hoffman, and B. R. E. The precis environment. In Proceedings of National Conference on Artificial Intelligence (AAAI-93) ARPA-RL Planning Initiative (ARPI) Workshop, Washington, DC, 1993.

[14] M. P. Singh, A. S. Rao, and M. P. Georgeff. Multiagent Systems A Modern Approach to Distributed Artiffical Intelligence, chapter Formal Methods in DAI: Logic Based Representation and Reasoning, pages 201-258. MIT Press, Cambridge, MA., 1999.

[15] A. Tate. Intelligible ai planning. In M. Bramer, A. Preece, and F. Coenen, editors, Research and Development in Intelligent Systems XVII (Proc. 20th ES), pages 3-16. Springer, 2000.

[16] J. Vokrinek, A. Komenda, and M. Pechoucek. Relaxation of social commitments in multi-agent dynamic environment. In Proceedings of International Conference on Agents and Artificial Intelligence (ICAART09). Springer, 2009.

[17] J. Vokrinek, A. Komenda, and M. Pechoucek. Decommitting in multi-agent execution in non-deterministic environment: Experimental approach. In AAMAS '09: Proceedings of the eight international joint conference on Autonomous agents and multiagent systems, pp. 977-984, 2009.

[18] M. M. de Weerdt, A. Bos, J. Tonino, and C. Witteveen. A resource logic for multi-agent plan merging. Annals of Mathematics and Artificial Intelligence, special issue on Computational Logic in Multi-Agent Systems, vol. 37, no. 1-2, pp. 93-130, Jan. 2003.

[19] G. Wickler, A. Komenda, M. Pechoucek, A. Tate and J. Vokrinek. Multi-Agent Planning with Decommitment. In proceedings of Knowledge Systems for Coalition Operations 2009.

[20] M. Wooldridge and N. Jennings. Cooperative problem solving. Journal of Logics and Computation, 9(4):563-594, 1999.

[21] M. Wooldridge. Reasoning about Rational Agents. Intelligent robotics and autonomous agents. The MIT Press, 2000.

Decommitting in Multi-agent Execution in Non-deterministic Environment: Experimental Approach

Jiří Vokřínek
Agent Technology Center
Czech Technical University in Prague
vokrinek@agents.felk.cvut.cz

Antonín Komenda
Agent Technology Center
Czech Technical University in Prague
komenda@agents.felk.cvut.cz

Michal Pěchouček
Agent Technology Center
Czech Technical University in Prague
pechoucek@agents.felk.cvut.cz

ABSTRACT

The process of planning in complex, multi-actor environment depends strongly on the ability of the individual actors to perform intelligent decommitment upon specific changes in the environment. Reasoning about decommitment alternatives during the planning process contributes to flexibility and robustness of the resulting plan. In this article we formally introduce and discuss three specific decommitment rules: (i) relaxation, (ii) delegation and (iii) full decommitment. We argue that appropriate selection, setting and preference ordering of the decommitment rules contributes to robustness (measured as a number of failures) of the overall plans. The presented claims are supported by empirical experiments.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Intelligent agents*; I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Plan execution, formation, and generation*

General Terms

Measurement, Performance, Reliability, Experimentation, Verification

Keywords

social commitment, decommitment rule, commitments based planning, non-deterministic environment

1. INTRODUCTION

The process of planning in complex, multi-actor environment depends strongly on the ability of the individual actors to perform intelligent decommitment upon specific changes in the environment. Reasoning about decommitment alternatives during the planning process contributes to flexibility and robustness of the resulting plan.

Multi-agent research community has provided a viable formalism for representing agents commitment towards their individual as well as joint intentions. Wooldridge and Jennings have formalized such mental attitude of the agents by means of special knowledge structures, referred to as *social commitments* [13].

Cite as: Decommitting in Multi-agent Execution in Non-deterministic Environment: Experimental Approach, Jiří Vokřínek, Antonín Komenda, Michal Pěchouček, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 977–984

Social commitment is a knowledge structure describing agent's obligation to achieve or maintain a specific goal, under specific conditions. The commitment does not capture description how the committed goal can be achieved. An important knowledge component in the commitment – *decommitment rule*, sometimes referred to as a *convention* [12] – provides information about how and under which circumstances the commitment can be dropped. Reasoning activities of a rational agent such as individual planning for a goal achievement, plan execution and monitoring as well as replanning, plan reparation or plan merging and plan coordination are supported by the information encoded in the social commitments.

In the context of classical planning, agents deliberate about primitive (or compound) actions, components of the plan, in order to form appropriate ordering (or decomposition) representing a result of a specific planning problem. While a typical action in a plan contains only a set of preconditions and resulting postconditions, we suggest extending the action representation with the commitment-related information. A planning agent will not only reason about preconditions and effects of an action but also about how much and in which way it can rely on someone implementing the given action. This capability is critical for agents to be able to coordinate their actions and to perform *multi-agent planning* in the sense of forming plans (*i*) by interaction among multiple autonomous agents and (*ii*) to be executed by multiple autonomous agents.

The product of planning with commitments is a set of partially ordered terminal actions, allocated to individual actors who agreed to implement the actions under certain circumstances, clustered into two categories :

- *commitment condition* that may be (*i*) a specific situation in the environment (such as completion of a particular precondition) or (*ii*) a time interval in which the action has to be implemented no matter what the status of the environment is or (*iii*) a combination of both.
- *decommitment conditions* specifying under which condition the actor is allowed to decommit from the commitment once the task is finished (e.g. notification) or once the task cannot be completed (e.g. a failure)

Planning with commitments results in a plan where agents commit themselves to carry out actions leading towards achievement of e.g. a joint persistent goal [5]. Planning strategies used for establishing such a plan are usually based on the delayed-commitment principle [1]. Other approach uses eager-commitment strategy as showed in [11].

In this paper we formally introduce and discuss three specific decommitment rules: (i) relaxation, (ii) delegation and (iii) full decommitment. We argue that appropriate selection, setting and preference ordering of the decommitment rules contributes to robustness (measured as a number of failures) of the overall plans. The particular focus of this contribution is the preference ordering of the decommitment rules in various non-deterministic environments. Besides formal specification of the decommitment rules, the key research contribution lies in experimental analysis of the use of the various decommitment strategies and their mutual dependence.

The article is structured as follows. Section 2 gives a brief overview of works most relevant to our approach. In Section 3 we adapt the formal description of commitment by Wooldridge and introduce a particular commitment convention to improve the flexibility of the commitments. The verification and experimental evaluation is presented in Section 4. Finally, the last section concludes the paper.

2. RELATED WORK

Formalization of commitments has been extensively studied in the past using various formalisms, most of all building on and extending the BDI framework [9] when describing obligations the agents adopt. Fasli [4] uses a model based on branching temporal components from Computational Tree Logics (CTL) [3], which is much more expressive than our model based on temporal intervals, however it is too complex for experimental deployment. Fasli also defines strategies regarding conditions for a successful decommitment from the agent's obligations, which in several aspects correlate with Wooldridge's convention [12] and thereby with our decommitment rule set.

Another formal representation of commitments considering temporal account has been introduced in [7]. CTL has been extended to capture features usually not considered in common approaches (but relevant for realistic environments), namely time intervals considered in commitments satisfaction, *maintenance* type of commitments next to *achieve* type of commitments and vague specification of time. However, most of these aspects can be also captured using BDI and Wooldridge's social commitment framework showed in [12] in combination with an appropriate temporal model.

The uncertainty in agent's commitments has been studied in [14]. The authors have extend the commitment with *"... uncertainty by explicitly describing the possibility of future modification/revocation of the commitment ..."*. The paper concentrated on the uncertainty in the quality of the commitment fulfilment (quality of service) rather than on decommitting conditions.

The last but not least is an extensive related research field of planning in dynamic and/or uncertain environment. There is a wide range of approaches, for example probabilistic planning – MAXPLAN [6], contingency planning – system CIRCA [8], planning under uncertainty [2]. These approaches focus on creating plan alternatives to avoid uncertainty, in contrary to the commitment based planning where the emphasis is put on individual commitments and its decommitment strategies.

3. DECOMMITMENT RULES

As stated in the Introduction, the targeted topic of the research proposed in this paper is the impact of the decommitment rules on the plan execution. In this section, we provide formalization for the three most commonly used decommitment rules.

We require the agents that perform intelligent planning and replanning by means of social commitments to be able to perform at least basic reasoning about the decommitment rules attached to the particular commitments. This is needed at the time of replanning, when an agent needs to decide which decommitment rule (i.e. a new commitment) to adopt, provided that conditions for more than one of them are satisfied. Similarly, when negotiating about who will accept which commitment, the agents shall be able to analyze not only properties of the goal and costs associated with the goal completion process but also the various decommitment rules when considering likelihood of the particular failure to happen. Ideally, the agent shall be able to estimate costs of each decommitment rule. In the scope of this paper we are not addressing the agents' decision making, but we focuss on the performance and usability of several decommitment strategies settings during execution of the commitments in dynamic environment.

Michael Wooldridge in [12] defines the commitments formally as follows:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda),\\ &\quad\lambda = \{(\rho_1,\gamma_1),(\rho_2,\gamma_2),\ldots,(\rho_k,\gamma_k)\},\end{aligned} \tag{1}$$

where A denotes a committing actor, ψ is an activation condition, φ is a commitment goal, and λ is a convention. The convention is a set of decommitment rule tuples (ρ,γ) where ρ is a decommitment condition and γ is an inevitable outcome. The convention describes all possible ways how the commitment can be dropped. Generally speaking, the actor A has to transform the world state in such a way that the φ goal becomes true if ψ holds and any γ has not been made true yet. The actor is allowed to drop the commitment if and only if $\exists i : \rho_i$ which is true. A decommitment is fulfilled provided that γ_i is made true. A formal definition in modal logic (working with the models of mental attitudes like Believes, Desires, Intentions, [9], and temporal logic where the operator AG denotes inevitability and operator $\curvearrowleft$ denotes the temporal until) follows as defined in [12]:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda)\equiv\\ &\quad((\mathsf{Bel}\ A\ \psi)\Rightarrow\mathsf{AG}((\mathsf{Int}\ A\ \varphi)\\ &\qquad\wedge(((\mathsf{Bel}\ A\ \rho_1)\Rightarrow\mathsf{AG}((\mathsf{Int}\ A\ \gamma_1)))\curvearrowleft\gamma_1)\\ &\qquad\ldots\\ &\qquad\wedge(((\mathsf{Bel}\ A\ \rho_k)\Rightarrow\mathsf{AG}((\mathsf{Int}\ A\ \gamma_k)))\curvearrowleft\gamma_k)\\ &\quad)\curvearrowleft\bigvee_i\gamma_i).\end{aligned} \tag{2}$$

This definition is used in a declarative way. Provided that whatever the agent does during a specific behavior run complies with the above defined commitment, the expression 2 is valid throughout the whole duration of the run.

The structure of our commitments is based on this definition and the decommitment rule set is in detail discussed in the next section.

We have recognized three main types of decommitment usually used in commitments: (i) *Full Decommitment* for dropping the commitment, (ii) *Delegation* of the commitment to another agent, and (iii) *Relaxation* of the time frame of the commitment. The decommitment condition for each decommitment strategy is defined to enable flexibility of the commitment under various circumstances. During the planning process, the preference relation over the decommitments is defined as a part of the decommitment rule set. The decommitment rules are unordered according to the definition (1) and thus we must slightly change the definition of the commitment:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda),\\ &\lambda = \big((\rho_1,\gamma_1),(\rho_2,\gamma_2),\ldots,(\rho_k,\gamma_k)\big),\end{aligned}\tag{3}$$

where the ordering is fixed and the rules are processed in a specific order. The processing of the rule means that dropping a part of the commitment definition in (2)

$$\bigvee_i \gamma_i \tag{4}$$

simplifies to

$$\gamma^*, \tag{5}$$

where γ^* is the inevitable outcome of an active decommitment rule. There is only one active rule for each commitment and the rules are switching from the first rule to the last one. The switch is performed only if γ^* is not realizable and ρ of the next active rule holds. The switching process uses defined fixed ordering of the rules to determine the correct succeeding rule.

According to our understanding, each decommitment rule set (corresponding to Wooldridge's commitment convention) must contain two basic rules, which ensure the racionality of the agent's decision making process. These rules are based on the definition of the *open-minded commitment* defined in [12]:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \varphi) \equiv\\ &\mathsf{AG}((\mathsf{Int}\ A\ \varphi) \curvearrowleft ((\mathsf{Bel}\ A\ \varphi) \vee \neg(\mathsf{Bel}\ A\ \mathsf{EF}\varphi))),\end{aligned}\tag{6}$$

where the operator EF denotes future possibility. Thus in each and every commitment the initial rule should be the success rule

$$(false, (\mathsf{Bel}\ A\ \varphi)) \tag{7}$$

and the decommitment rule set should be always closed with a fail-safe rule turning a violated commitment eventually off

$$(false, \neg(\mathsf{Bel}\ A\ \mathsf{EF}\varphi)), \tag{8}$$

which is used provided that no other rule can be used and the commitment became unrealizable.

The decommitment condition ρ in rules (7) and (8) cannot become true (as $\rho = false$), which means the agent will never intend to the decommitment rule outcome γ according to the definition (2)

$$\begin{aligned}&((\mathsf{Bel}\ A\ \rho) &\Rightarrow\ &\mathsf{AG}((\mathsf{Int}\ A\ \gamma))) \curvearrowleft \gamma\\ &(false &\Rightarrow\ &\mathsf{AG}((\mathsf{Int}\ A\ \gamma))) \curvearrowleft \gamma,\end{aligned}\tag{9}$$

nevertheless the rule can drop-out the commitment using the drop-out part

$$\bigvee_i \gamma_i \text{ or } \gamma^* \tag{10}$$

respectively. This principle can be used for any drop-out rule with no need for explicit intention.

Definition 3.1. *Each commitment can be decommitted if the commitment goal φ is achieved (the commitment succeeded) or if the commitment goal φ can not be achieved any more (the commitment is violated).*

The formal definition of the decommitment rule set in each commitment follows

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda),\\ &\lambda = \big((false, (\mathsf{Bel}\ A\ \varphi)),\\ &\qquad \ldots \text{investigated decommitment rules} \ldots,\\ &\qquad (false, \neg(\mathsf{Bel}\ A\ \mathsf{EF}\varphi))\big),\end{aligned}\tag{11}$$

where the three investigated rules are injected between two basic rules and thereby the last violation rule can be avoided. The rate of avoidance is one of the experimental metrics and is discussed in Section 4.

Three proposed decommitment rules can be defined using the adopted formalism as follows:

Definition 3.2. *Full decommitment decommits the original commitment if and only if the commitment goal φ is unrealizable.*

Definition 3.3. *Delegation decommits the original commitment if and only if the commitment goal φ is unrealizable and the new commitment on the other agent's side is formed.*

Definition 3.4. *Relaxation decommits the original commitment if and only if the commitment goal φ is unrealizable, the negotiated relaxation conditions hold and the relaxed commitment is formed.*

In the three following subsections, we describe the rules in more details and we formalize them using a temporal model, based on the duration time interval of the commitment being the only constraint of the commitment goal φ. This model is suitable for commitment-based planning, since the plan is a (partially) ordered list of temporally successive commitments.

3.1 Full Decommitment

The basic decommitment strategy is dropping the commitment. Under defined circumstances the agent is completely released from the commitment.

Let the commitment time interval $T_\varphi = \langle t_s, t_e \rangle$, where t_s is the starting time and t_e is the ending time of the commitment time interval. The commitment duration is defined as $t_d = t_e - t_s$ Let the commitment goal condition φ contain only defined temporal properties, then the decommitment rules can be described as:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda),\\ &(t_s^{est} > t_s \Rightarrow update(t_s, t_s^{est}), false) \in \lambda,\\ &(t_e^{est} > t_e, t_e^{est} > t_e) \in \lambda,\\ &t_e \in \varphi,\end{aligned}\tag{12}$$

where t_s^{est} and t_e^{est} are estimations of the real start and end of the activity. The first part of the rule describes continuous adjustment of the commitment's start time in the case the agent is forced to postpone its execution (which may not affect the end time condition and can not affect other commitments). The second part reflects Definition 3.2.

The t_s, t_e are the parameters of the commitment negotiated and fixed at the planning (contracting) time. The estimates t_s^{est} and t_e^{est} are continuously updated and can vary over time.

3.2 Delegation

By using this type of the decommitment rule the agent shall be able to find some other agent who will be able to complete its commitment on the original agent's behalf. It is possible that such a commitment will contain unbound variables representing the need to search for an agent suitable for delegation. The basic idea is to find an agent that is able to undertake the commitment under circumstances when the decommitment condition (which is true in case of the original agent) became false, so the new agent is able to fulfill the commitment. The delegated commitment can contain a new set of decommitment rules.

Formally we can use the same variables as in full decommitment, but we are using

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda),\\ &\quad(t_s^{est} > t_s \Rightarrow \mathit{update}(t_s, t_s^{est}), \mathit{false}) \in \lambda,\\ &\quad(t_e^{est} > t_e, (\mathsf{Commit}\ B\ \psi\ \varphi\ \lambda)) \in \lambda,\\ &\quad t_e, B \in \varphi,\end{aligned} \tag{13}$$

where B is the other agent undertaking the commitment.

3.3 Relaxation

Relaxation is a special decommitment, where the original commitment is replaced with a new commitment with relaxed condition and/or goal. In the scope of this text we are focusing on the relaxation of the commitment time interval for the sake of simplicity. The commitment time interval is usually captured by the commitment subject φ and specifies the time frame booked for the commitment execution. The temporal uncertainty can be a part of the commitment subject definition (and thus the whole commitment has to be renegotiated in case of any change) or, more preferable, it can be included in the commitment as an instance of a decommitment rule.

According to Definition 3.4, the decommitment rule can be then described as:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda),\\ &\quad(t_s^{est} > t_s \Rightarrow \mathit{update}(t_s, t_s^{est}), \mathit{false}) \in \lambda,\\ &\quad((t_s^{est} < t_s) \wedge (t_s^{est} \in T_s^{rlx}),\\ &\qquad(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda) \wedge \mathit{update}(t_s, t_s^{est}) \in \lambda,\\ &\quad((t_e^{est} > t_e) \wedge (t_e^{est} \in T_e^{rlx}),\\ &\qquad(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda) \wedge \mathit{update}(t_e, t_e^{est}) \in \lambda,\\ &\quad t_s, t_e \in \varphi,\end{aligned} \tag{14}$$

where T_s^{rlx} and T_e^{rlx} are the agreed relaxation intervals (negotiated relaxation conditions) for the start and end time. The T_s^{rlx} and T_e^{rlx} is an extended set of parameters negotiated and fixed at the planning time and the $\mathit{update}(t_s, t_s^{est}$ part changes the temporal parameters of the newly forming commitment to relaxed values.

3.4 Impact of Decommitment Rules

The impact of the particular rules is discussed in Section 4.2.1. We assume the complex combination of the decommitment rules provides non-trivial behavior and should improve the performance of the commitments' execution in non-deterministic environments under stress conditions (the system is overloaded). Let us postulate the following hypothesis:

HYPOTHESIS 3.1. *A proper combination of the three defined decommitment rules, i.e. relaxation stated in Definition 3.4, delegation stated in Definition 3.3, and full decommitment stated in Definition 3.2 improves the commitment execution stability in the non-deterministic environment and preserves the utilization of resources and should increase the commitment's execution success rate.*

Figure 1: Scenario island screenshot

Each of the presented rules provides different impact on the agent's current state. For example, relaxation helps to maintain the commitment execution, delegation effectively unblocks the agent's resources and full decommitment releases the agent's resources by dropping the commitment. We expect a combination of the decommitment rules to emerge in a self-adaptation pattern that should lead to some sort of a real-time commitment execution optimization.

The decommitment rules introduced in this chapter have been implemented in the commitment-based planning system and experimentally evaluated. The next section describes the experimental scenario and discusses the influence of the rules on stability of the commitments execution and decommitment flexibility.

4. EXPERIMENTS

The decommitment rules temporally formalized in Section 3 have been deployed in a realistic simulation scenario based on an island inspired by the Pacifica Suite of Scenarios[1]– Fig. 1. The scenario simulates limited information visibility and information sharing. Due to this, the environment provides non-deterministic behavior from a single unit's point of view. There are heterogenous independent self-interested units in the scenario that commit to the goals. During the execution of a plan the commitments are processed. The commitment can evolve according to the plan or due to unexpected environment interactions. Monitoring of the commitments is triggered by a change of the world, e.g. a tick of the world timer, movement of a unit, a change of a world entity state, etc. The process evaluates all commitments in the actor's knowledge base. The value of the commitment defines the commitment state and can start the decommitting process.

For the experimental evaluation purposes we have designed a multi-actor transport scenario, where individual agents provide non-accurate estimates of the transportation time (the execution time may differ because of unexpected events such as unit breakdowns, path changes, etc.). We combine decommitment strategies introduced in Section 3. The influence of the selection of strategies and ordering is analyzed by a series of experiments.

In the experimental scenario, there is a set of *resource agents* able to provide a unified resource to the *requestor agent*. The requestor agent introduces a set of tasks and allocates it to the resource agents. The allocation is done by the planning process that takes tasks one by one and finds the best resource agent for its execution. The planning pro-

[1]http://www.aiai.ed.ac.uk/oplan/pacifica

cess is based on the well-known contract-net-protocol [10] and provides an almost even distribution of tasks across the resource agents. During planning, the appropriate decommitment rules are set according to the experiment settings (see Section 4.2).

The experiments have been evaluated by the simulation, where the results have been aggregated from 10 runs for each experiment setting. For each run, random values of configuration variables have been generated. Each agent recomputes the parameters for the next ongoing commitment according to the current state and executes decommitment rules when necessary.

The decommitment rules execution differs for each commitment according to the experiment setting. The decommitment rules have been set according to the definitions in Section 3. A detailed description of the implementation follows:

Basic decommitment rules – according to Equation 11, we can use the temporal model formerly proposed; the goal φ is achieved if the commitment execution is finished not later than t_e and the commitment is violated if the execution is finished later than t_e (the goal φ cannot be achieved and the resource time frame is wasted).

Full decommitment – according to Equation 12; the t_e^{est} is computed during the simulation. If this rule applies the commitment is removed from the plan.

Relaxation – according to Equation 14; this rule can be applied before the commitment execution (relaxation of the start of the commitment t_s within T_s^{relax}) or during the commitment execution (relaxation of the end of the commitment t_e within T_e^{relax}).

Delegation – according to Equation 13; the t_e^{est} is computed during the simulation. If this rule applies the agent tries to find another agent which is able to undertake the commitment (the original commitment is delegated with no decommitment rules except the basic ones). The delegation is based on negotiation between agents, where each agent bids for the commitment undertaking. If there is no winning bidder the new commitment is not formed and the decommitment condition ρ remains true.

The delegation algorithm is based on contract-net-protocol. Each agent prepares the bid for the commitment delegation based on it's current state and commitment parameters. The bid computation is the following:

1. Let t be the current simulation time and ${}^D t_s$, ${}^D t_e$, ${}^D t_d$ parameters of the commitment D that has to be delegated.
2. If there is an active current commitment C in time t, compute
$$ {}^D t_e^{est} = {}^C t_e^{est} + {}^D t_d, $$
else set
$$ {}^D t_e^{est} = t + {}^D t_d. $$
3. If there is a next commitment N in the plan, compute
$$ t_{limit} = {}^N t_s^{est}, $$
else set
$$ t_{limit} = \textit{positive infinity}. $$
4. If the next commitment N contains a relaxation rule, recompute the estimate as
$$ t_{limit} = \max(T_s^{rlx}). $$
5. Compute the bidding value
$$ bid = t_{limit} - {}^D t_e^{est}. $$
6. If $bid < 0$ reject delegation, else send the bidding value *bid*.

The negative bidding value means that commitment D cannot be inserted to the agent's plan without breaking consequent commitments. Only the potential relaxation of the first consequent commitment is taken into account when estimating impact of delegation on the agent's plan. This approach doesn't affect the risk of subsequent commitments violation and has linear complexity, but it reduces the possibility of the delegation with comparison to the more complex methods operating with the subsequent commitments' rule sets.

4.1 Scenario Setup

The experiments show the influence of the selected decommitment rules and their order on flexibility, robustness and execution stability in the non-deterministic stressed environment.

In the experiments, the environment dynamics is simulated by non-deterministic prolonging of the activities. This dynamics is not taken into account by agents during the planning process. The prolonging events are generated for each agent individually using uniform distribution with mean value $\bar{e} = 15000$ time units and variance $\sigma^2 = 8333$ time units. Each experiment has been performed on a sequence of 10 randomly generated runs. The duration of the prolonging event varies from 0 to 14000 to evaluate the system behavior under different stress conditions and it is referred to as *repair time* t_r[2]. The system is critically overloaded when $t_r = 15000$, where the repair time is equal to prolonging events' meantime and the execution of commitments fails.

To enable the possibility of delegation rule execution we introduce vacant resources - the agents with no plans that are joining the system during execution phase and they are able to undertake delegated commitments. The number of *vacant agents* is set to 5 which produces 10% of overall free resources.

Each agent has a plan containing 100 commitments. The duration of the commitment execution t_d (ideally with no prolonging events) is randomly generated with uniform distribution from 5000 to 15000 time units. The start time of the commitment t_s is set to the earliest possible time of the winning resource agent and the end time is set to

$$ t_e = t_s + t_d, \tag{15} $$

which makes 100% load of the agents in ideal conditions (with no prolonging events taken into account). The relaxation intervals are set to

$$ T_s^{rlx} = \langle 0.7 \times t_s, 1.3 \times t_s \rangle, T_e^{rlx} = \langle 0.7 \times t_e, 1.3 \times t_e \rangle \tag{16} $$

[2]The individual agent load can be computed as $(1 + \frac{t_r}{\bar{e} - t_r}) \times$ 100% and is varying from 0 to 1500%.

that makes 30% relaxation intervals. The commitment execution is non-interruptible, so if the decommitment rule applies after the commitment execution is started the resource is blocked for the whole t_d.

When we enable the prolonging events the overall system performance is very stressed. In the experiments we focus on the qualitative results of presented decommitment strategies rather than fine tuning commitment parameters according to current experimental settings. The evaluation and discussion is presented in the next chapter.

4.2 Results

This chapter summarizes the results of experiments based on the experimental setting described above. First, we will discuss the influence of individual decommitment rules used separately. Next, we will show the influence of the mixed strategies. We measure the number of executed decommitment rules and the number of successfully achieved commitments. Due to the over-stressed system, the utilization of agents is 100%, thus this parameter is not evaluated.

4.2.1 Single Rule

The first experiment provides the results of influence to the commitment execution for single rules usage. The delegation (D), relaxation (R) and full decommitment (Fd) rules have been used separately. For comparison we also measured the empty decommitment set noted as basic. For $t_r = 0$ there are zero rule executions and 100 successful commitments. The individual agent stress experiment results are the following (see Figures 2 and 3):

Basic – the number of successful commitments varies from 0 to 2 in the whole range. No decommitment rules are executed.

Full decommitment – the number of rule executions grows with the increasing t_r. The curve converges to the maximum number of commitments for a critically overloaded system. The number of successful commitments decreases with increasing t_r from 18 and converges to 0 for the critically overloaded system.

Delegation – the number of rule executions corresponds to possibility of delegation to the vacant agents. When vacant agents are saturated the delegation uses agents freed by the delegation of longer commitments. The number of successful commitments goes from 19 to 10. The variance between agents starts to be significant when $t_r > 10000$ so the robustness of this rule is decreasing.

Relaxation – the relaxation rule provides the best stability. It is executed for every commitment and provides no violations when the system is overloaded below the relaxation interval limitations. When the relaxation interval fails, the number of executed rules goes to 0 very fast and so does the number of successful commitments.

The delegation rule execution is also affected by the number of vacant agents. The relaxation and full decommitment rules are obviously not affected by the number of vacant agents. The second experiment examines this dependency. Figure 4 shows a typical curve shape for 5 vacant agents and

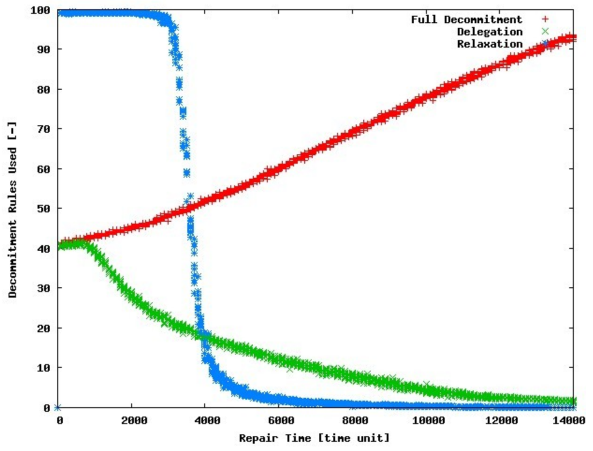


Figure 2: Number of decommitment rules execution for different t_r in single rule setting.

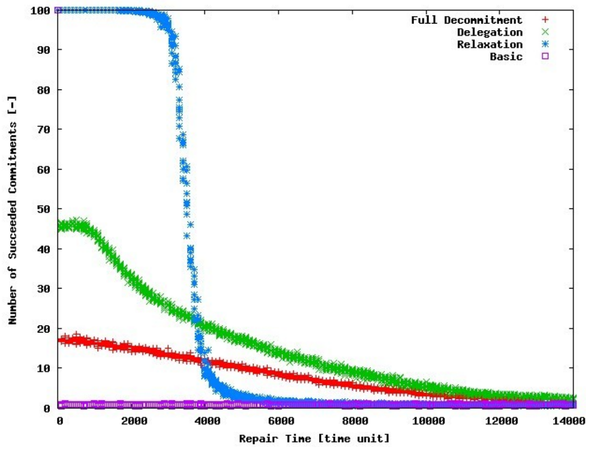


Figure 3: Number of succeeded commitments for different t_r in single rule setting.

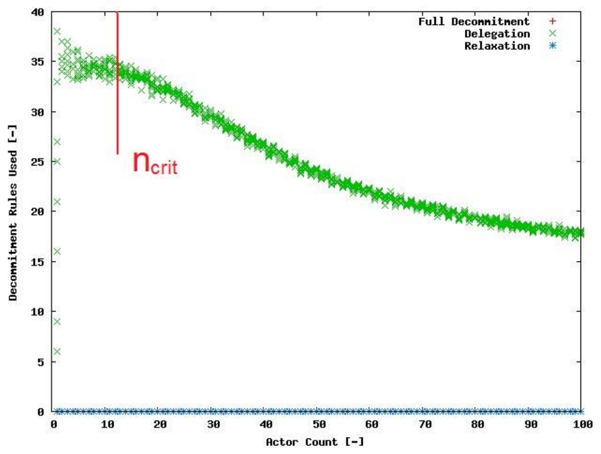


Figure 4: Number of delegation rule execution for different number of resource agents. The number of vacant agents is 5 and $t_r = 2000$

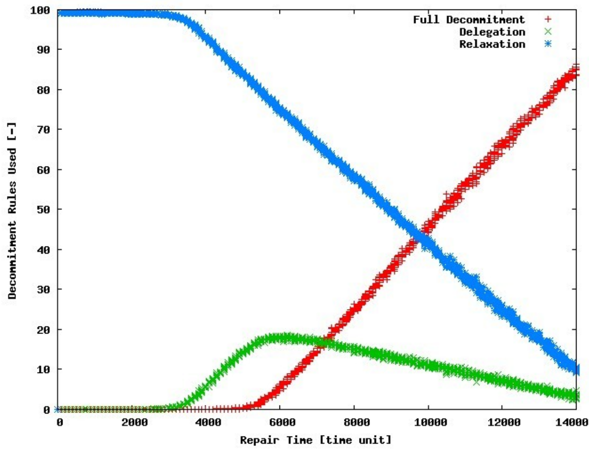


Figure 5: Number of rules execution for varying t_r in combined rule R-D-Fd setting.

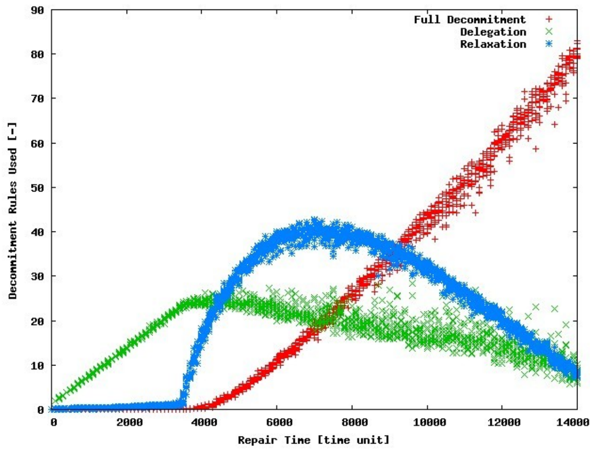


Figure 6: Number of rules execution for varying t_r in combined rule D-R-Fd setting.

$t_r = 2000$. This shape remains the same even for a different setting of the parameters. The number of decommitment rule executions is constant until the number of agents reaches the critical value n_{crit}. After this point the number of executions decreases because of saturation of the vacant agents. At this stage, the commitments are delegated mainly to other resource agents (the influence of vacant agents is decreasing). The position of the n_{crit} point depends on the experiment setting. With an increasing number of vacant agents the n_{crit} shifts to the right and with an increasing t_r it shifts slightly to the left. It's position corresponds to the relative number of vacant agents in the system and the individual agent's stress (the vacant agents are saturated sooner with increasing t_r).

4.2.2 *Combined Rules*

This set of experiments inspects the influence of decommitment rules combinations and their ordering. The main focus is on the two ordering scenarios – R-D-Fd for

$$relaxation \succ delegation \succ fulldecommitment$$

and D-R-Fd for

$$delegation \succ relaxation \succ fulldecommitment$$

that provide the most significant results. The full decommitment rule is ordered as the last one, because of its nature – no decommitment rule can be applied after its application.

The combination of rules provides complex results. The number of individual rules execution can be seen in Figures 5 and 6. The number of the full decommitment rule executions is similar in both cases but with different impact on the number of successful commitments. In the first part of the chart, when the system is *slightly overloaded* ($t_r < 3000$), only the first rule applies. In the range of an *overloaded* system ($t_r \in \langle 3000, 12000 \rangle$) the second and the third rule starts to apply. When the system is close to the *critical overload* ($t_r > 12000$) the number of rules executed starts to provide increasing deviation across the experiments runs and the robustness of the execution is reduced.

The number of successful commitments is compared for R-D-Fd and D-R-Fd with D-Fd, R-Fd, R-D and D-R sets and presented in Figure 7.

4.3 Discussion

The experiments show that for an overloaded system there is an increasing number of dropped commitments using full decommitment rule. The full decommitment rule applies when all preceding rules fail. This rule effectively release resources originally booked for dropped commitments. This causes a bigger chance for delegation of commitments and space for relaxation. Delegation rule provides the ability of real-time re-allocation of commitments according to current agents performance. The experimental results shows the ability of the system to adapt to the overload and thus to increase the number of succeeded commitments with increasing size of decommitment rule set and keep high utilization of available resources (execution time of the commitments compared to free time of resources excluding prolonging events).

As shown in Figure 7, the number of successful commitments is reaching 50% for R-D-Fd (D-R-Fd) for $t_r = 7500$ (7000) that corresponds to system load of 200% (187%). At this point, the R-D-Fd (D-R-Fd) method is able to utilize 100% (94%) of the overall system resources available. For $t_r = 7500$, the single rule settings (including basic rule set with no decommitment rules) reach maximum of 15% of succeeded commitments for delegation rule (Figure 3), which is 30% of utilization of available resources. Combined rule sets composed from decommitment rule pairs reach maximum of 30% of succeeded commitments, which is 60% of utilization of available resources.

This experimental observations prove Hypothesis 3.1 for any combination of rules. The best performance provides the biggest sets of decommitment rules. The R-D-Fd set has the biggest success rate of commitments execution until $t_r = 7000$. For bigger t_r the higher success rate can be observed for D-R-Fd, but with lower stability (higher variation of experiment runs). The best success rate for near critical load ($t_r > 10000$) can be reached with D-R set, but with minimal stability (most of the experiment runs provide worse results then both R-D-Fd and D-R-Fd).

The sets containing Fd provides generally better results, but may not be suitable in all application domains because of commitment drop-out by this rule.

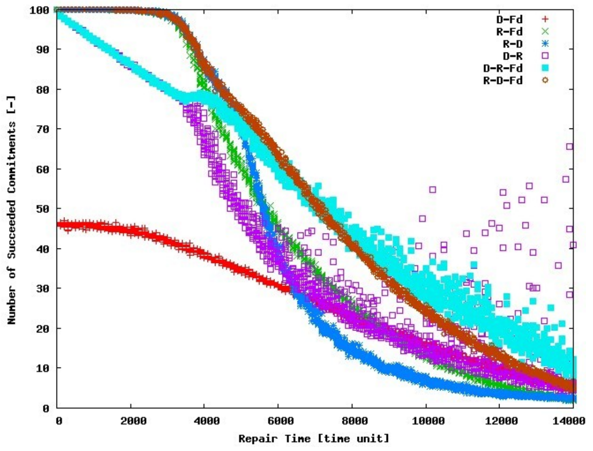


Figure 7: Number of succeeded commitments for varying t_r for different rules settings.

5. CONCLUSION

Based on the well-known social commitment representation we have defined basic decommitment rules for open-minded commitments representation. We have formalized three decommitment strategies (relaxation, delegation, and full decommitment) and showed how they affect application of the commitments in non-deterministic stressed environment. The evaluation of the presented approach has been made on an experimental realistic scenario and deployed in a multi-agent system for commitment-based distributed planning.

The relaxation decommitment rule provides the best performance in the limits of an estimated relaxation model. The delegation rule produces a relatively high amount of violations (even for small t_r) because of the usage of the non-optimal algorithm. Further improvement of this algorithm (e.g. by involving future commitments' decommitment rule sets) may lead to reduction of the number of violations and all the results affected by delegation rules may scale down. The full decommitment rule significantly reduces the number of violated commitments under high load, but may not be suitable for all applications (because of commitment drop-out).

The combinations of particular rules provides a complex decommitment behavior and significantly improves commitment execution performance and stability. The success rate of commitment execution and available resources utilization significantly increases with the size of the decommitment rule set. Different rule combinations have to be chosen for different application scenarios. We have identified, evaluated and discussed the strong and weak points of the presented combinations of decommitment rules.

6. ACKNOWLEDGMENTS

Presented effort was sponsored by Czech Ministry of Education grant 6840770038, supported by the European Research Office of the US Army under grant number W911NF-08-1-0041, and by Internal Grant of Czech Technical University under number CTU0802913.

The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

7. REFERENCES

[1] A. Barrett and D. S. Weld. Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112, May 1994.

[2] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[3] E. Emerson and J. Srinivasan. Branching time temporal logic. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop*, volume 354 of *LNAI*, pages 123–172. Springer-Verlag, 1988.

[4] M. Fasli. On commitments, roles, and obligations. In B. Dunin-Keplicz and E. Nawarecki, editors, *CEEMAS 2001*, volume 2296 of *LNAI*, pages 93–102. Springer-Verlag Berlin Heidelberg, 2002.

[5] H. Levesque, P. Cohen, and J. Nunes. On acting together. In *AAAI-90 Proceedings, Eighth National Conference on Artificial Intelligence*, volume 2, pages 94–99, Cambridge, MA, USA, 1990. MIT Press.

[6] S. M. Majercik and M. L. Littman. MAXPLAN: A new approach to probabilistic planning. In *Artificial Intelligence Planning Systems*, pages 86–93, 1998.

[7] A. Mallya, P. Yolum, and M. Singh. Resolving commitments among autonomous agents. In F. Dignum, editor, *Advances in Agent Communication. International Workshop on Agent Communication Languages, ACL 2003*, volume 2922 of *LNCS*, pages 166–82. Springer-Verlag, Berlin, Germany, 2003.

[8] D. J. Musliner, E. H. Durfee, and K. G. Shin. CIRCA: A cooperative intelligent real time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1561–1574, - 1993.

[9] M. P. Singh, A. S. Rao, and M. P. Georgeff. *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*, chapter Formal Methods in DAI: Logic Based Representation and Reasoning, pages 201–258. MIT Press, Cambridge, MA., 1999.

[10] R. G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. *In IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[11] M. Veloso and P. Stone. Flecs: Planning with a flexible commitment strategy. *Journal of Artificial Intelligence Research*, 3:25–52, 1995.

[12] M. Wooldridge. *Reasoning about Rational Agents.* Intelligent robotics and autonomous agents. The MIT Press, 2000.

[13] M. Wooldridge and N. Jennings. Cooperative problem solving. *Journal of Logics and Computation*, 9(4):563–594, 1999.

[14] P. Xuan and V. R. Lesser. Incorporating uncertainty in agent commitments. In *In: Proc. of ATAL-99*, pages 221–234. Springer-Verlag, 1999.