

I-Globe: Distributed Planning and Coordination of Mixed-initiative Activities

Antonín Komenda
Agent Technology Center
Czech Technical University in
Prague

Jiří Vokřínek
Agent Technology Center
Czech Technical University in
Prague

Michal Pěchouček
Agent Technology Center
Czech Technical University in
Prague

Gerhard Wickler
AIAI, University of Edinburgh
Edinburgh, Scotland

Jeff Dalton
AIAI, University of Edinburgh
Edinburgh, Scotland

Austin Tate
AIAI, University of Edinburgh
Edinburgh, Scotland

ABSTRACT

We present an approach to distributed planning and coordination architecture for dynamic non-deterministic multi-actor mixed-initiative environment. The system provides flexible planning, replanning, and task allocation. The key idea of the presented approach is in integration of (i) I-X hierarchical planner with (ii) agent-based architecture and (iii) commitment-based plan representation. The implementation of the system was verified and evaluated on simulated environment. The experimental validation confirms the performance, stability, and robustness of the system in complex scenarios.

1. INTRODUCTION

The problem of controlling entities in heterogeneous distributed environment is crucial for many domains [3]. Classical centralized methods depend on one central planning system. Such a system gathers all required input data before the planning process take place. Then the plan (set of plans respectively) is generated using these data. This approach faces various problems. One problem is a need for private local knowledge of the actors. The other problem is the need for real-time replanning based on dynamically changing environment and conditions in the time. On the other hand, in distributed methods of planning each entity plans its own plan. Cooperation and heading to common goals is done by negotiating methods.

The problem of distributed planning has been often discussed in the AI planning and multi-agent research communities recently (e.g. [3], [2], [4], [1]). 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, sub-task delegation, conflict detection, individual planning and plan merging. In this paper we describe those phases as: HTN planning (Durfee's task decomposition and individual planning), distributed resource allocation (Durfee's sub-task delegation and partially conflict detection), and plan coordination (Durfee's plan merging and partially conflict detection).

1.1 Scenario

The approach presented in this paper is being verified on a realistic simulation scenario emphasizing mixed-initiative planning and decision making. Figure 1 shows a scenario island inspired by the Pacifica Suite of Scenarios¹ (Pacifica domain for DARPA planning initiative [7], [8] and other experiments). The Pacifica scenarios adopt the concept *Go places, do things*, a tasking base that allows for a range of missions to be designed to 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. The scenario actors are several unit types (ground, armored, aerial or sea units), civilians and hostile units.

Pacifica covers the need for a scenario with flexibility of the scale of the roads and places involved to allow for the introduction of more details as the scenario experiments demand it.

The scenario is based on an island with hostile environment and limited information visibility and sharing. Due to this, the environment provides non-deterministic behavior from the single unit point of view. There are heterogeneous independent self-interested units in the scenario that commit to the shared/joint goals. To fulfill the desired strategic goals in such 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 for each type of the unit. The strategic plans generation is provided by a set of *commanders* that are responsible for each type of *field units* - the 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 receives the strategic goals from it. The hierarchical structure of the tactical planners then create 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.

¹<http://www.aiai.ed.ac.uk/oplan/pacifica>



Figure 1: Scenario island screenshot

There is a number of heterogeneous unit types operating in the scenario e.g.:

- **Commanders** – abstract units (not geographically simulated) that represent *Ground*, *Armored*, *Aerial* and *Sea* headquarters.
- **Stationary units** – units representing assembling points for civilians – *Cities*, material and resources providers – *Supply depots* and *Petrol pumps*, landing and refueling zones for aerial units – *Airfield* and docking and refueling zones for sea units – *Seaports*.
- **Mobile units** – units with the ability to move on the island. There are ground units, which are *Transporters* (can provide faster transportation of other unit(s), material or civilians), *Construction* (can repair damages or assemble/disassemble stationary units) and *Medical* (provides medical care for other units or some rescue operations). The *Armored* units for protection of other units or secure an area or convoy. The *Aerial* – the UAVs with an extended visibility range and *Sea* units for transportation over the water.

1.2 Issues

In previous section, the environment, domain, and scenario was described. The problem of automated planning in such a world can be described by a several more-or-less separable problems. The problems have to be solved to fulfill the requirement for planning system able to plan in dynamic non-deterministic environment. The overview of the problems follows:

- **Distributed planning** – Planning in such an environment is practically realizable only as a distributed process. This affirmation is supported by several facts: the objects of planning are naturally distributed in the world. The communication among the entities can be

restricted, constrained or limited in other ways. The robustness of the process should be conserved even in such a circumstances. And finally, each entity have to hold its own private knowledge of its capabilities in form of a planning domain.

- **Distributed resource allocation** – 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. The allocation process must be appropriately integrated with the planning system and similarly, the planning process has to be robust with respect to the mentioned constraints in the environment.
- **Distributed plan execution and synchronization** – Constituted distributed plan consisting of several personal plans has to be executed by the entities then. The personal plans need to be distributively coordinated, as the entities do not know each other plans. As well as, the plan has to be robust enough to be able to minimize its volatility and do not need to be completely replanned in case of any non-determined effect in the planning phase.

Next section shows the planning system in one whole description and answers the question, how the problems are interconnected and affecting each other. It also describes particular approaches solving the three mentioned issues.

2. APPROACH

The presented approach is based on the concept of multi-layer planning architecture (Section 2.1). The whole planning process of an overall plan is distributed among arbitrary amount of autonomous agents. The planning hierarchy of the entities is not predefined, and it emerges during the planning. Each agent knows only its own planning domain, which describes the agent's capabilities in terms of the environment domain. This private personal domains are described in the form of Hierarchical Task Networks (HTN). The planning process (Section 2.2) is initiated by externally tasked agent(s). The tasks are typically added by a human operator using a human-machine-interface (HMI) system. In our case, it's the I-X Control Panel that serves as the HMI. The I-X Control Panel is a part of the I-X architecture. The agent tries to fulfill the task goals and may need to incorporate sub-plans of other agents, in the case it is not able to fulfill the task on its own. Such agents recursively run the same planning process until the whole plan is formed and ready for execution. In the phase of incorporating the sub-plans, the agents need to mark parts of the plans where the other agents continue with the plan execution. For that purpose, the designed concept of plan interconnection by synchronization-points (Section 2.2.3) can be used. The parameters of the spatio-temporal synchronization points are negotiated during the process of forming of the planning hierarchy. The synchronization points are later used during the plan execution.

From the perspective of one agent, the planning process can be divided into three layers, which form the multi-layer planning architecture. In the strategic layer (the top-most layer), the HTN I-Plan planner [12] is used, creating abstract plan for a long-time horizon. The planner is a part of the I-X

architecture and originates in the O-Plan planner. The plan instantiating process uses a distributed resource allocation (Section 2.2.4) based on the well-known multi-agent Contract Net Protocol [11]. With the help of this protocol, the appropriate subordinate agents are found and the responsibilities of the plan actions are fixed. The tactical layer (the middle layer) optimizes the plan using 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 non-deterministic environment. The effect is directly proportional to the amount of non-determinism in the world. The personal layer (the bottom-most layer) plans potential refinements of the tactical actions. One of these actions is movement, where the path is planned using the A* algorithm. The other responsibility of the personal layer is execution of all low-level actions in the scenario simulator.

All plans are described in the form of social commitments (Section 2.2.1) (substituting plan actions). The commitment is a knowledge-base structure describing agent's obligation to change the world-state and a set of rules what the agent should do if the obligation is not satisfiable. The proposed structure is an extension of a widely used formalization of the commitment [16]. The commitment recursiveness [5] enables more expressive description of the decommitment rules and thus the replanning process. The introduction of the causal commitment inter-referencing enables real-time replanning constraining. The mutual bindings and commitments form a commitment graph. The graph notation can be used for the process of successive solving of the exceptional states (replanning). 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. In the case when the decommitment rule inter-references other commitment, the process switches to the referenced commitment and starts one of the decommitment rules on the side of the referenced commitment. Provided that the decommitment rule terminates the commitment without the need to switch to other commitments, the process ends here, the violation is fixed and the plan is successfully replanned. In other words, the replanning process by means of social commitments can be described as successive re-committing [5]. For the decommitting purposes, three basic decommitment rules were used: full decommitment, delegation, and relaxation [14]. The most suitable decommitment rule set and ordering for non-deterministic domains was used, according to [15]: delegation, relaxation, and full decommitment, in this particular order.

The simulator is based on the ACROSS2² testbed. The testbed is heavily supported by the A-Globe multi-agent platform [10]. The testbed and platform are implemented in the Java programming language, as well as the rest of the system.

2.1 Multi-layer Planning Architecture

The architecture is based on a hierarchy of planning layers using an extended form of social commitments as the integration part. Because of the modularity of the layers, the architecture is open for integration of various planning methods. The variety of possible layers enables maximal

²ACROSS2 is a new major version of the ACROSS system – <http://agents.felk.cvut.cz/projects/#across>

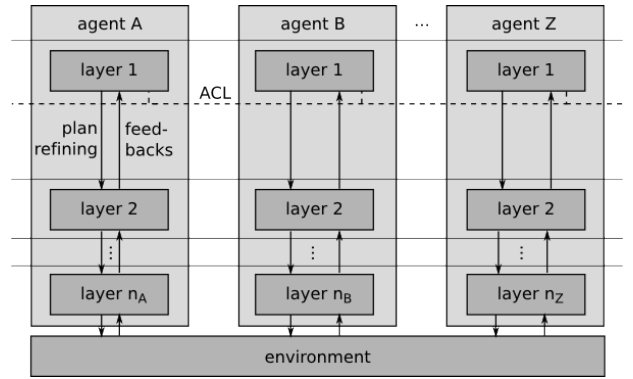


Figure 2: Multi-layer Planning Architecture

utilisation for the distributed planning problem.

The layers are mutually interconnected, meaning that each layer is directly connected only to its neighbouring layers. As the layer forms a batch, each layer is connected to $\langle 0, 2 \rangle$ other layers.

The top-down direction represents the successive refinement process of the plan, plan instantiation, refinement, and finally plan execution. On the other hand, the bottom-up direction is used for controlling of the planning process, which means informing the higher levels of the planning results, success, and failure feedbacks.

The layer loose coupling and simple process flows enable minimization of the plans' volatility, since there is no need to replan the plans on higher layers (typically more abstract and long-term plans) in the case of only minor violations (which can be solved by the bottom-level replanning). As higher levels of planning typically invoke much more resources-consuming processes (such as inter-agent negotiation, HTN planning, etc.) this separation helps to optimize the planning process as a whole.

Each agent uses all layers or their subset (e.g. the commander unit does not use the bottom-most layer, because it is not typically executing the plan at all and only uses its subordinates to implement the plan in the environment). The basic architecture design consists of three layers:

- **Strategic layer:** The layer provides an overall strategic plan for mid and long-term time horizon using I-Plan for high-level planning and distributed allocation based on multi-CNP protocol and social commitments.
- **Tactical layer:** On this layer, the strategic plan is optimized using a commitment condensation algorithm. The algorithm searches for blank time slots in the plan and replaces them with later commitments (taking into account the time constraints of the commitments).
- **Individual layer:** On this layer, the units perform reactive behavior and path-finding based on the tactical plan (i.e. an ordered set of commitments).

2.2 Distributed Planning

Distributed planning is a decentralized process of the plan constitution. According to the proposed Multi-layer Planning Architecture, the planning process takes place on several levels (of abstraction and granularity) in several different agents. Those planning processes collaborate to form one

global plan considering desired goals and particular agent constraints.

2.2.1 Distributed Plan Representation

In planning, appropriate distributed plan representation plays the main role. The representation should be suitable for flexible planning and plan revision purposes and should provide execution robustness and effective apparatus for handling various types of plan execution exceptional effects. All these requirements can be handled using the concept of social commitments for planning [5].

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. For the planning purposes, the recursive form of the commitment can be used:

$$\begin{aligned} (\text{Commit } A \psi \varphi \lambda^*), \lambda^* = \\ \{(\text{Commit } x_1 \rho_1 \gamma_1 \lambda_1^*), \\ (\text{Commit } x_2 \rho_2 \gamma_2 \lambda_2^*), \dots, \\ (\text{Commit } x_k \rho_k \gamma_k \lambda_k^*)\}. \end{aligned} \quad (1)$$

Formula 1 extends the definition in [16] 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 [5].

2.2.2 Distributed Plan Forming

By the means of the commitments, the planning process can be described as committing to appropriate actions. A sequence of these actions can be found by any planning approach. We use HTN I-Plan as the main planner of the system. The planner is supported by a distributed resource allocation mechanism based on extension of the CNPs (see in Section 2.2.4).

The instantiated plan is converted into commitments. The conversion process creates a commitment according to the particular action in the plan and according to forward causal 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 the negotiation respect the particular needs of the agents. The tactical commitments also define decommitments to the strategic layer and they can additionally refine some strategic commitment too. They are much more refined than the strategic commitment in the sense of spatio-temporal constraints, and particular world-states. The individual layer plans commitments for later execution. These commitments copy the tactical commitments, but some of these can be omitted. Each individual commitment contains a decommitment request only to its parent commitment (from the tactical layer).

During the execution of the plan commitments are processed. The commitments can be decommitted according to the plan or due to unexpected environment interactions.

The 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.

2.2.3 Distributed Plan Coordination

Since the plan is executed by autonomous actors in parallel, we had to define an efficient method for distributed plan execution coordination. For that purpose, we designed a concept of plan synchronization-points (or synchro-points in short). A synchro-point is a pair of commitments (or actions in the planning terminology). One member of the pair is a wait commitment and the other is a notify commitment. Each side of the synchro-point is on a different agent. The wait commitment is not succeeded until the paired notify commitment is executed (or intended in the BDI terminology), which means that the execution of one agent's plan can be temporarily delayed until another agent finishes some actions in its plan. The parameters of the spatio-temporal synchronization-points are negotiated during the process of forming of the plan.

In some cases there is a need for synchronization of plans of agents, which are not neighbours in the planning hierarchy. Since the planning hierarchy can emerge into any number of sub-ordinate agent levels the synchronization-points have to be able to handle synchronization of distant levels too. We chose an approach in which the trans-level synchro-points are composed of primitive synchronization-points connecting only neighbour levels of the planning hierarchy and forming a chain across several plans and agents. More formally, we can define one line of the hierarchy as:

$$A \rightarrow B_1 \rightarrow B_2 \rightarrow \dots \rightarrow B_n \rightarrow C, \quad (2)$$

where A , B_i , and C are agents. Now the trans-level synchro-point can be described using the primitive synchro-point as follows:

$$A \implies C \equiv \{A \rightarrow B_1, B_1 \rightarrow B_2, \dots, B_{n-1} \rightarrow B_n, B_n \rightarrow C\}. \quad (3)$$

The main advantage of this approach is no need for search of all causally affected agents in case of replanning.

2.2.4 Distributed Resource Allocation

We have encountered three dimensions of the resource allocation problem in our domain. The first two dimensions are two points of view of a similar problem. The third one is an inverse problem.

The first resource allocation problem is to allocate a set of tasks, defined by a total ordered plan, which means that the tasks are dependent and ordered. The tasks of the plan correspond to the linear horizontal commitment graph. Each agent creates a plan to achieve its commitment. According to the scenario and the domain of the agent, some parts of the plan cannot be executed by this agent and thus have to be delegated to other agents. The allocation problem is then to find the best candidates that are able to take oversuch parts of the plan (tasks). Every successive task allocation constraints depends on previous task allocation. The criteria function for allocation optimization is a function of all tasks allocations. This means (i) no task can be allocated optimally without knowledge of other tasks allocations and (ii) no task can be allocated without fixation of allocation of all preceding tasks.

The second resource allocation problem is hierarchical task allocation. This problem represents vertical commitment graph creation. When an agent commits (or even prepares the commitment) to some task, it has to decompose this task and make a plan for it. So each task that is being allocated implies the need of consequent allocation of other tasks on other actors.

The third problem occurs when multiple independent tasks are requested simultaneously. This potentially produces (i) overbooking of the best resource provider and (ii) unbalanced recourse utilization.

In the complex scenario, there should be a complicated hierarchical structure of agent domains. All three allocation problems occur simultaneously. The agents' hierarchy lead to both horizontal and vertical commitments dependencies. The task concurrency is caused by multiple goals inserted in the system and by the substitutional agents' capabilities. To get over the distributed allocation problem, we have implemented a progressive task allocation mechanism with as-soon-as-possible execution heuristic and backtracking. We have applied a recursive task allocation mechanism using iterative CNP with backtracking when allocation fails because of overbooking caused by concurrency. Efficiency and speed of convergence of the allocation mechanism is illustrated in Section 3.

2.3 I-X Architecture Integration

There are a number of tools available that help people organize their work. One of these is provided with virtually every organizer, be it electronic or paper-based: the "to-do" list [6]. This is because people are not good at remembering long lists of potentially unrelated tasks. Writing these tasks down and ticking them off when they have been done is a simple means of ensuring that everything that needs to be done does get done, or at least, that a quick overview of unaccomplished tasks is available. In responding to an emergency this is vital, and the larger the emergency, the more tasks need to be managed.

I-X Process Panels constitute the primary user interface to an I-X application. A panel more or less directly reflects the ontology underlying the whole I-X system, the <I-N-C-A> ontology [13], which is a generic description of a synthesis task (such as design, planning or configuration), dividing it into four major components: Issues, Nodes, Constraints, and Annotations. When used to describe processes, nodes are the activities that need to be performed in a course of action, thus functioning as the items in an intelligent to-do list. The other elements contain issues as questions remaining for a given course of action, information about the constraints involved and the current state of the world, and notes such as reports or the rationale behind items in the plan.

In <I-N-C-A>, both processes and process products are abstractly considered to be made up of a set of Issues which are associated with the processes or process products to represent potential requirements, questions raised as a result of analysis or critiquing, etc. They also contain Nodes (activities in a process, or parts of a physical product) which may themselves have sub-nodes making up a hierarchical description of the process or product. The nodes are related by a set of detailed Constraints of various kinds. Finally there can be Annotations related to the processes or products, which provide rationale, information and other useful descriptions.

<I-N-C-A> models are intended to support a number of different uses:

- for automatic and mixed-initiative generation and manipulation of plans and other synthesized artifacts and to act as an ontology to underpin such use;
- as a common basis for human and system communication about plans and other synthesized artifacts;

- as a target for principled and reliable acquisition of knowledge about synthesized artifacts such as plans, process models and process product information;
- to support formal reasoning about plans and other synthesized artifacts.

These cover both formal and practical requirements and encompass the requirements for use by both human and computer-based planning and design systems.

The I-X architecture is used in I-Globe on several levels. The main HTN planner of the I-Globe system is the I-Plan planner. It is instantiated in each entity and used by the multi-layer planning architecture for long-term planning. The scenarios are extended such that commander agent has I-P² (I-X Control Panel) to allow for mixed initiative planning and the state subscription interfaces and protocols are used to allow state monitoring for the commander through a selective subscription mechanism. The other I-P² is used as a user interface for the Sense Maker actor, which can task through the I-X the sub-ordinate commanders using their control panels.

2.4 AGENTFLY Integration

The AGENTFLY system [9] was integrated as one of the individual layers in the planning architecture. AGENTFLY is a multi-agent air-traffic-control system, focusing on distributed control and collision avoidance of unmanned aerial vehicles. It adopts a free flight concept and in I-Globe it is primarily used as a planner and a collision solver of aerial units.

In order to seamlessly integrate the two existing systems, an AGENTFLY wrapper was implemented. On one hand, the wrapper fulfills the needs of the multi-layer planning architecture and on the other it acts as a control module of the AGENTFLY pilot agent. Using this wrapper, the I-Globe system can transparently control missions of the simulated UAVs and does not need to implement the plane's behavior and simulation on its own. The planner wrapper is based on an agent-to-agent protocol (Figure 3).

The protocol can handle two main operations: mission definition with planning and plan execution with replanning. The initial definition of the mission is a repeatable process of plan duration approximations and planning attempts. The plan is executed after agreement of both sides was reached. During the execution AGENTFLY notifies I-Globe of successfully achieved mission points and/or of mission spatio-temporal deviations. I-Globe is able to change a part of the currently processed AGENTFLY plan. This process takes into account the time delays between the two planning systems. It is based on the concept of an unchangeable mission point. Such a point is designated in run-time and lies on the processed plan in the future. Only the part of the plan beyond that point can be changed by replanning, which implies that the communication between the two systems must be done before the unchangeable point is reached. If the time interval is exceeded the replanning starts again. The replanning process includes the whole planning process: mission definition, planning, execution, but the new mission begins at the unchangeable point. The replanning process is always initiated by I-Globe and it can occur at any time I-Globe system needs it. AGENTFLY can only notify I-Globe of deviated mission points, but the decision if the replanning is necessary is up to I-Globe. The replanning necessity is re-

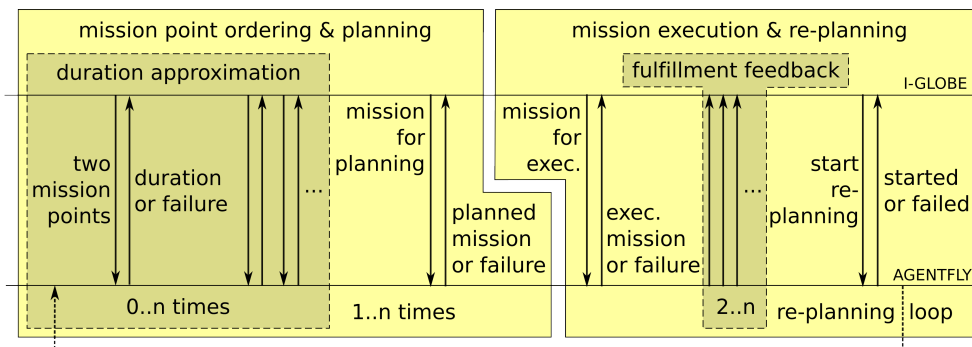


Figure 3: I-Globe – AGENTFLY control protocol

solved by the multi-layer planning architecture, specifically by the commitments modification process.

3. EVALUATION

This section shows the behavior of the system in the complex interaction scenarios. We focus on the planning performance and resources utilization in various settings. In the first experiment the resource utilization and plan quality is empirically investigated. The second experiment targets the scalability of the planning and allocation algorithms.

We have evaluated the system in various scenario settings. The three levels of resource allocation problems has been gauged. We have varied (i) number of actors in the scenario, (ii) levels of planning hierarchy, and (iii) type of task concurrency. In the experiments the planning and allocation algorithms are empirically investigated. The plan execution duration (lengths of the plans and their deviation across all actors in the system) is examined. In the second experiment we focus on the communication complexity of the distributed task allocation process. The amount of contract-net-protocols needed for planning and the speed of convergence is examined.

The experimental scenario uses the three levels of planning/allocation hierarchy as described in Figure 4. The *commander* represents the abstract unit, which is able to generate tasks – in this case *build houses* task. This task is passed to the construction units of *builder* type and it is decomposed to the sequence of requests for transportation and houses construction. The mobile *truck* units are able to handle transportation request.

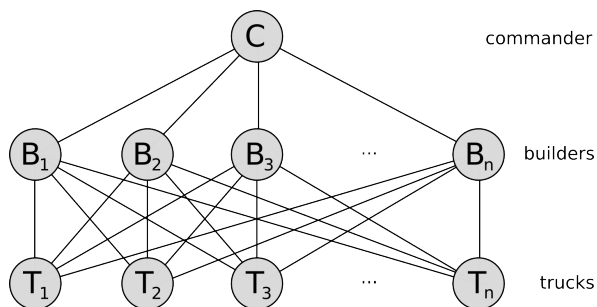


Figure 4: Experimental scenarios setting. There are three levels of hierarchy – *commander*, set of *builders* and set of *trucks*

The tests has been performed on the following settings:

- **Scalability test** – this test has been performed on simplified *build-houses* scenario with no commander. There are 20 builders in the scenario, each with one predefined *build houses* task. The number of trucks varies from 1 to 20. The construction materials are present on the construction sites, so only the builder transportation is needed. Each builder tries to find the best possible transportation to the construction site. The trucks are requested by all builders concurrently.
- **Over-booking test** – this test has been performed on *resources* scenario. There are 10 builders in the scenario, each of them is requested to perform one 1 *build houses* task. The decomposition of *build houses* generates two requests: *request transport* and *request resources*. Those two requests are allocated to the trucks sequentially. When the builder finds the best transportation for itself, it request the transportation of construction materials based on constraints of the first transport. Due to concurrent allocation, the first transport constraints should not be ensured and thus backtracking (reallocation of first transport) may occur. The number of trucks varies from 1 to 20.
- **Cross-booking test** – this test has been performed on *multi-task* scenario. There are 10 builders in the scenario, each with predefined sequence of 5 *build houses* tasks. The number of trucks varies from 1 to 10. For each builder the quality of allocation of individual requests are strongly interdependent. Every successive request allocation depends on previous request allocation. The criteria function for allocation optimization is function of all requests allocations. The optimization of requests allocation in parallel generates huge backtracking due to massive cross-booking.
- **Multi-level test** – this test has been performed on *build-houses* scenario with commander. The setting is similar to the scalability test. There is 1 commander and 4 builders. The number of trucks varies from 1 to 15. The commander requests sequence of 2 to 20 *build-houses* tasks from builders. The builders requests transport to the construction site concurrently.

3.1 Execution Length

This section shows the behavior of the system in the complex interaction scenarios. We focus to the planning per-

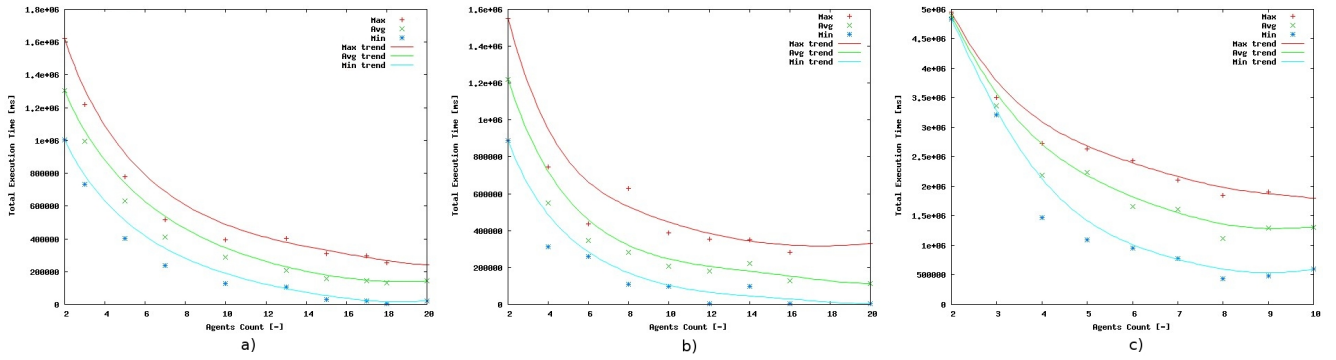


Figure 5: Length of the plans for different agents counts and scenarios: (a) scalability test (build-houses scenario), 20 builders, 1 task, n trucks; (b) over-booking test (resources), 10 builders, 1 task refined to 2 requests, n trucks; (c) cross-booking test (multi-task scenario), 10 builders, each with sequence of 5 tasks, n trucks.

formance and resources utilization in various settings. The measured parameter is length of plans (measured in seconds of simulation) and its deviation across all actors in the system. This experiment has been evaluated for the scalability, over-booking and cross-booking tests

Figure 5 shows the total execution time (a.k.a. length of plan) in the described settings. All settings provide similar results. The execution time goes down fast with increasing number of trucks. Due to overheads needed for truck sharing (waiting times, empty kilometers, etc.), the improvements slows down for approx. 7 trucks in the scalability test, 5 trucks in the over-booking test and 4 trucks in the cross-booking test. The general tendency of the execution time provides logarithmic dependency on the number of trucks in all three scenarios. The saturation of trucks is the lowest (the highest scalability) in the scalability test because of low dependency of the tasks. In the overbooking scenario the system is saturated for approx. 10 trucks and then doesn't provide execution time reduction (average total execution time converges to 200s). The cross-booking scenario provides the worst scalability. The saturation can be observed for approx. 6 trucks and the average total execution time converge to 1500s.

3.2 Negotiation Complexity

This section shows the behavior of the system in the complex interaction scenarios. We focus to the negotiation complexity of the distributed task allocation (as a part of the commitment planning – see Section 2.2.4). We investigate the number of open contract-net-protocols, successfully closed protocols and failed protocols (corresponds to backtracking in planning process) and it's tendency over the time. The number of failed CNPs is aggregated number of failures on the protocol initiator and responders side. The CNP is failed when one party drops the agreed commitment. The initiator is no longer interested in the commitment (because of another conflicts) or the responder is not able to fulfill the commitment (because of overbooking). The values are measured in time intervals $u = 500ms$ of processor time and provides the overview of the complexity of task allocation during planning phase.

The first experiment have been performed on the same scenarios as in Section 3.1. The number of trucks has been fixed to 17 for scalability test, 14 for over-booking test, and

10 cross-booking test. The results are presented in Figure 6.

In the scalability test, there are 20 open CNPs on the beginning (corresponds to 20 builders requesting single transport). The number of successful CNPs grows linearly. In the time $u = 40$ the conflicts are detected and number of canceled CNPs starts to grow. The agents replans and thus number of open CNPs also starts to grow. The number of open and canceled CNPs grows almost linearly until the number of successful CNPs converges to the number of open CNPs in the time $u = 100$. At this moment, the planning phase is finished and all transport tasks are allocated to the trucks. The allocation of 20 transport tasks has been accomplished using 49 contract-net-protocols.

In the over-booking test, there are 10 open CNPs on the beginning (corresponds to 10 builders requesting transport to the construction site). Sequentially, the builders starts to request transport of construction materials (resources). The tendency of the CNPs is similar to the scalability test, but the number of canceled CNPs is higher because of trucks overbooking. The conflicts are started detected in the time $u = 12$. The planning procedure converges in time $u = 65$. The allocation of 20 transport tasks (transporting builders and materials) has been accomplished using 74 contract-net-protocols.

The cross-booking test provides the highest number of CNPs. There are 10 open CNPs on the beginning (corresponds to first transport of each builder). Sequentially, the builders starts to request transport to second construction site and following transports. The tendency of the CNPs is also similar to the scalability test. The curve of the open CNPs clearly shows the polynomial complexity of the negotiation, because of high cross-booking, the number of failed CNPs is higher then number of open CNPs (the commitment is dropped by both party simultaneously). The conflicts are started detected in the time $u = 60$ (the number of open CNPs is 50 – all the transportation are allocated and builders start to close the CNPs. At this moment, number of successful CNPs grows, but also the number of failed CNPs. The horizontal steps of the curves corresponds to the planning activity and opening new CNPs for previously failed allocations. The planning procedure converges in time $u = 165$. The allocation of 50 transport tasks has been accomplished using 275 contract-net-protocols. For compari-

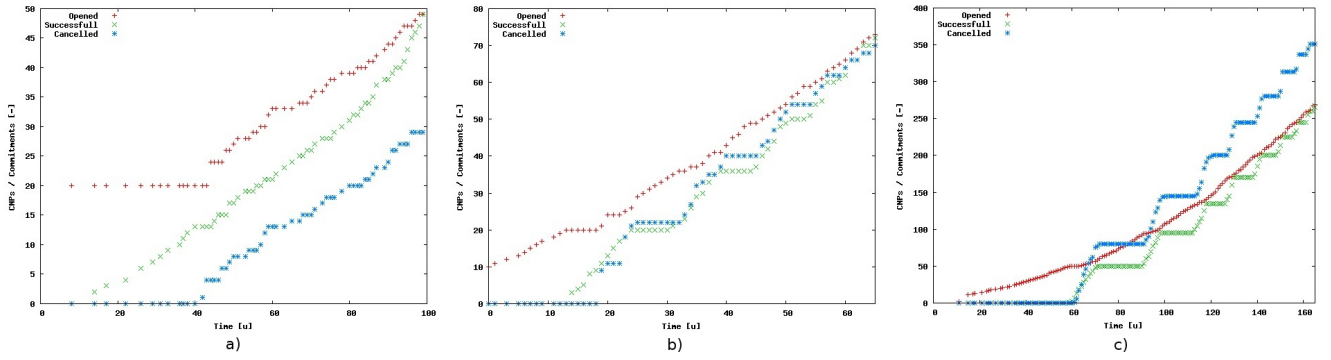


Figure 6: Course of one execution run showing the dynamic values of the CNPs for different scenarios: (a) scalability test (build-houses scenario), 20 builders, 1 task, n trucks; (b) over-booking test (resources), 10 builders, 1 task refined to 2 requests, n trucks; (c) cross-booking test (multi-task scenario), 10 builders, each with sequence of 5 tasks, n trucks.

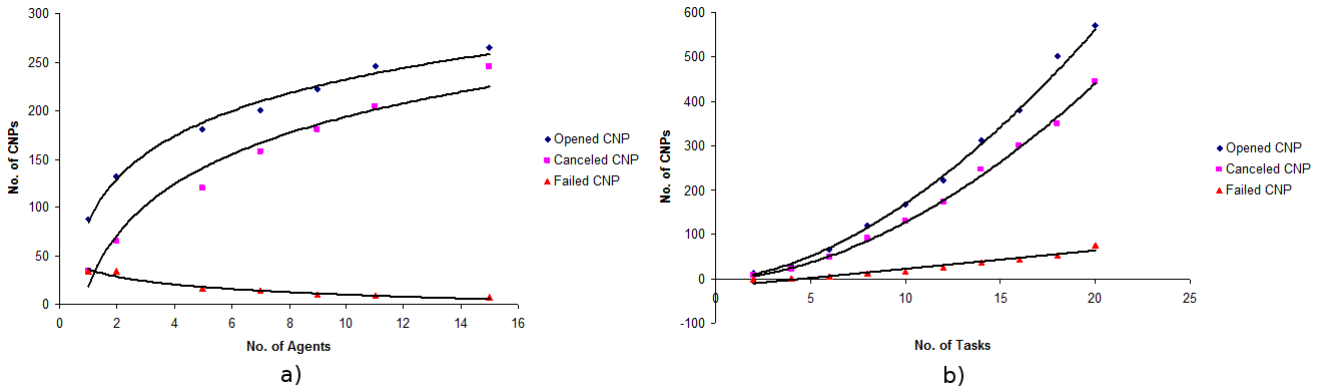


Figure 7: The number of CNPs in multi-level test for (a) 1 to 15 truck agents (1 commander and 4 builders) and (b) 2 to 20 tasks.

son, the setting of this scenario with 4 builders and 4 trucks (20 transport tasks total) the total number of contract-net-protocols is 50.

The second experiment shows the behavior of multi-level task allocation. Figure 7 shows the dependency of number of CNPs on number of agents (trucks varying from 1 to 15) and on number of tasks (varying from 2 to 20). For fixed number of tasks the number of needed CNPs grows logarithmically with increasing number of truck agents. With increasing number of simultaneous tasks in the system there is exponential grow of the needed CNPs.

4. CONCLUSION

The paper describes in a coherent way a complex planning system, which is based on multi-agent techniques and I-X systems integration architecture. Into the planning process, the human operators are involved in the level of commanding agents. The rest of the planning hierarchy is controlled by the autonomous planning agents. As the multi-layer planning architecture is modular enough, the different types of units can be controlled by different systems on the bottom-most planning level. The ground units use simple path planning algorithms. In contrary of them, the aerial units are operated by complex agent-based air-traffic-control system.

The robustness and the stability of the system has been verified on extended scenario suite. The performance has been experimentally evaluated on various scenarios – scalability tests, over-booking tests, cross-booking tests, and multi-level tests. We have identified exponential complexity of resource allocation problem with increasing number of concurrent tasks. However the implemented approach provides polynomial tasks allocation heuristics with complexity $\mathcal{O}(n^{2m}/2m)$ for m -level of planning hierarchy and n -agents in each level. The observed resource utilization is sufficiently spread over all actors. The length of plans decrease logarithmically with number of available actors.

5. ACKNOWLEDGMENTS

The presented research has been supported by the European Research Office of the US Army under grant number W911NF-08-1-0041. 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.

6. REFERENCES

- [1] M. M. de Weerd, 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*, 37(1-2):93-130, Jan. 2003.
- [2] M. E. DesJardins and M. J. Wolverton. Coordinating a distributed planning system. *AI Magazine*, 20(4):45-53, 1999.
- [3] 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.
- [4] E. Ephrati and J. S. Rosenschein. A heuristic technique for multiagent planning. *Annals of Mathematics and Artificial Intelligence*, 20(1-4):13-67, 1997.
- [5] 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.
- [6] T. Kreifelts, E. Hinrichs, and G. Woetzel. Sharing to-do lists with a distributed task manager. In G. de Michelis and C. Simone, editors, *Proceedings of the 3rd European Conference on Computer Supported Cooperative Work*, pages 31-46, Milano, 13-17 September 1993. Kluwer, Dordrecht.
- [7] 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.
- [8] 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.
- [9] D. Šišlák, M. Pěchouček, P. Volf, D. Pavlíček, J. Samek, V. Mařík, and P. Losiewicz. *AGENTFLY: Towards Multi-Agent Technology in Free Flight Air Traffic Control*, chapter 7, pages 73-97. Birkhauser Verlag, 2008.
- [10] D. Šišlák, M. Rollo, and M. Pěchouček. A-globe: Agent platform with inaccessibility and mobility support. In M. Klusch, S. Ossowski, V. Kashyap, and R. Unland, editors, *Cooperative Information Agents VIII*, number 3191 in LNAI. Springer-Verlag, Heidelberg, September 2004.
- [11] 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.
- [12] 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.
- [13] A. Tate. <I-N-C-A>: An ontology for mixed-initiative synthesis tasks. In *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems (MIIS) at the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 125-130, Acapulco, Mexico, August 2003.
- [14] J. Vokřínek, A. Komenda, and M. Pěchouček. Relaxation of social commitments in multi-agent dynamic environment. In *Proceedings of International Conference on Agents and Artificial Intelligence (ICAART09)*. Springer (to appear), 19-21 January 2009.
- [15] J. Vokřínek, A. Komenda, and M. Pěchouček. Deccommitting 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*, 2009 (to appear).
- [16] M. Wooldridge. *Reasoning about Rational Agents*. Intelligent robotics and autonomous agents. The MIT Press, 2000.