AMIaGO: Autonomous Multiple Intelligent-agents with Goal Orientation

Meirav Hadad and **Natalie Fridman** and **Gilad Armon-Kest** [1]

1 Introduction

Recently, several research groups as well as commercial enterprises strived to develop a general Artificial Intelligence (AI) engine for real-world applications that involves coping with problems which occur in uncertain, complex and dynamic environments. In these environments it is impractical and usually unfeasible to envisage and preprogram in advance all the possible situations and events that may occur. Accordingly, development of an AI engine should be aimed to include capabilities that enable it to reason about unanticipated opportunities. Moreover, typically the AI engine should be independent of domain knowledge to allow reusability across several domains. These essential properties enable more realistic behavior of the application; provide added flexibility to the system and save implementation efforts. The AI Research Team at BVR, Israel, developed an AI engine, which was integrated with training and simulation systems. The development was based on innovative and well-established techniques from the AI field. The objective of the AI engine is to support Autonomous Multiple Intelligent-agents with Goal Orientation (AMIaGO) behaviors.

The major novelty of the *AMIaGO* engine is its ability to plan a joint mission for multiple agents, where the information about the environment is typically incomplete, uncertain and dynamic. The agents may act in a cooperative fashion where various agents work jointly with each other, performing a common task or carrying out activities that need to be achieved to satisfy a shared goal. The agents make their decisions autonomously, however their decisions may also be influenced by human intervention. The agents' decision-making process and their cooperative behavior are based upon a well-established model of collaborative planning [2] that defines essential characteristics of teamwork and provides requisite flexibility. Unlike existing teamwork systems that focus on reactive team operators [3, 5] our engine also includes autonomous planning of the joint mission. Thus, the main advantage of our methodology is the ability to generate plans as new opportunities are recognized in the dynamic environment.

2 The System Architecture

AMIaGO was integrated with existing advanced training and simulation systems that were developed at BVR. Figure 1 depicts a high level description of the system components. The simulation engine is responsible to simulate the arena. In addition, there are four types of static Data Bases: (1) Geographical Knowledge Base (GKB): contains geographical data; (2) Exercise Planner (EP) includes the initial data of the training exercise (e.g., agents' types, agents' forces, their initial location, their initial mission); (3) Technical Knowledge Base

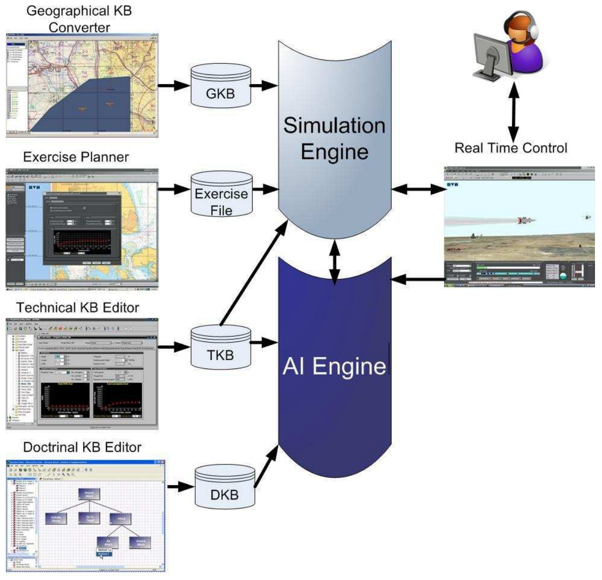

Figure 1. General description of the system

(TKB) contains properties on each agent (e.g. aircraft type, min/max velocity). The TKB also includes various types of teams and sets of agents aggregations possibilities (e.g., platoon, battalion) that describe possible ways of decomposing teams into sub-teams. (4) Doctrine Knowledge Base (DKB): contains a set of actions that the agent can perform. In addition, it includes a set of methods that indicate various possible ways of decomposing tasks into subtasks (equivalent to recipe library in planning systems). AMIaGO is responsible for the intelligent decision process of each agent in the system. The agent's decisions are based on the dynamic and static knowledge it gathers from the simulation engine as well as the information in the TKB and the DKB. The Real-Time (RT) control component enables the human trainer to influence the simulated arena. Furthermore, it provides the human way to intervene in the agent decision making process.

2.1 The AMIaGO Engine

Figure 2 represents the intelligent decision process of each agent, that is comprised from: decision maker process; cooperation level; handle failures process; and two types of planners (associated with the association process). The decision maker is responsible to receive the agent's perceptions and to decide on the agent's next steps accordingly. The cooperation level is based on a model of collaborative problem solving which enables the agent to autonomously reason about coordination and communication. The failure handler is responsible to handle failures (e.g., failures of its task performance, failures of is teammates, communication failures, etc.)

[1] Research Division, BVR Systems Ltd, Rosh Ha'Ayin 48091, Israel, email: { meirav,natalie,gilada }@bvr.co.il

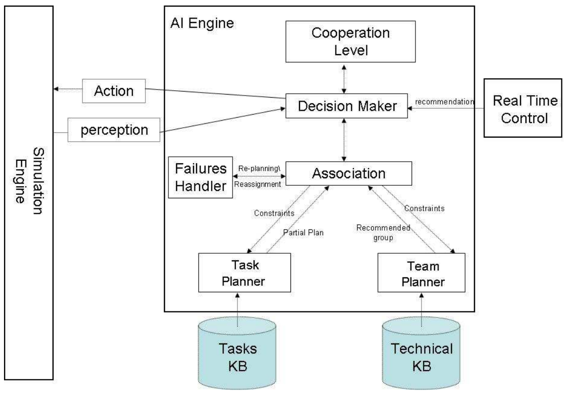

Figure 2. General description of the AI engine

The AMIaGO engine is designed to support goal-oriented behaviors of multiple agents that cooperate in a dynamic environment. Moreover, the objective is that agents will be able to generate their plans when new opportunities are recognized. To accomplish this challenge, a novel technique of dynamic association between several abstract solutions, represented by abstract graphs, is proposed. This technique enables to divide a complicated problem into simpler problems that may be resolved differently and employ different abstract graph representations. As each simple problem may be applied in the uncertain and dynamic world, it becomes impractical to achieve complete knowledge about the world, thus, the solution of each simple problem is resolved by the abstraction based technique. Also, each simple problem is represented by its own abstract graph and is resolved in its own way (based on the partial knowledge of the specific simple problem). Thus, in contrast with existing solutions in which the combination of simpler problems is represented by a single abstract graph (e.g., [4]), the proposed solution enables combination between different types of abstract graphs with different levels of abstractions. The association between the abstract graphs is done dynamically and is based on predetermined rules (e.g., assignment constraints between nodes of different graphs). Hence, the technique enables greater flexibility and modularity during the development of the solution, as well as during exercise planning and execution. Furthermore, the complicated problem may be resolved in a distributed approach as each simple problem can be resolved independently.

In training systems' scenarios we refer to a mission that is performed jointly by multiple agents as a complicated problem that is comprised of two simpler problem solvers - task planner and team planner. Each planner generates its own abstract graph. The task planner is based on a hierarchical planning strategy that exploits well-designed hierarchies of a task and is also known as a Hierarchical Task Network (HTN) planner [1]. Similarly, the team planner is responsible to organize teams into sub-teams according to their competence, resources or available information. At any stage of the planning development, each planner may generate only a partial plan (based on its current partial knowledge), which is repeatedly refined and becomes more complete as new knowledge is gathered. The integration between the partial plans is done dynamically by the association process as long as the partial plans are refined. The integrated solutions are applied on the environment, thus, planning and execution are interleaved.

Figure 3 depicts an example of an abstract plan of the mission *Attack ground station* that should be performed by *Aircraft package*. As shown in this figure, the abstract plan of the task is maintained by the task planner in a form of a task graph in which the tasks are represented as nodes and the dependency among the tasks are represented by edges (the left graph of Figure 3). Similarly, the abstract plan of the team is maintained by the team planner in a form of a team tree in which the teams are represented as nodes and the hierarchies among the teams are represented by the edges (the right graph of Figure 3). The graphs of both planners are extended dynamically by our technique and are associated by the association process as necessary. The task graph becomes complete when all the tasks are deconstructed into smaller and smaller subtasks until reaching the lowest level of atomic executable task. The team tree becomes complete in the identical way when reaching the lowest level of an individual agent.

Consider the above example of the *Attack ground station* mission by the *Aircraft package*. The mission may be deconstructed by several methods. Each method is appropriate for a specific situation in the environment. For example, to attack the ground station the force may use the low attack method or high attack method. The task planner is responsible to select the best method according to the knowledge about the environment's situation. If the ground station is well secured, the force may prefer the high attack technique which is much safer in these conditions.

After the planner selects the appropriate method it deconstructs the task into subtasks. For example *Attack ground station* according to low attack method may be deconstructed into the following subtasks: *Organize to attack*, *Fly to target*, *Air attack* and *Ground attack*. Yet, in an uncertain and dynamic environment, the knowledge about the environment is incomplete and further deconstruction of these subtasks must consider several properties on the environment which may be unknown in advance. Using the abstract based technique enables to gather new knowledge as the partial solution is applied within the environment. In a similar way, the team planner may deconstruct the *Aircraft package* to sub-teams in several different ways (e.g., Fourship formation, Two-ship formation). The association between the plan of the task and the plan of the team is done according to the constraints of the sub-plans.

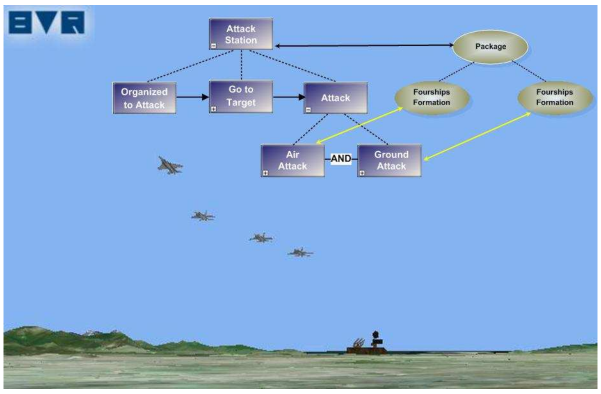

Figure 3. Example of graph association

REFERENCES

[1] K. Erol, D. Nau, and J. Hendler, 'HTN planning: Complexity and expressivity', in *AAAI-94*, pp. 1123–1128, (1994).

[2] B. J. Grosz and S. Kraus, 'Collaborative plans for complex group action', *AIJ*, **86**(2), 269–357, (1996).

[3] G. A. Kaminka and I. Frenkel, 'Integration of coordination mechanisms in the BITE multi-robot architecture', *ICRA-07*, 2859–2866, (2007).

[4] M. Paolucci, O. Shehory, and K. Sycara, 'Interleaving planning and execution in a multiagent team planning environment', *Electron. Trans. Artif. Intell.*, **4**, (2000).

[5] M. Tambe, 'Toward flexible teamwork', *JAIR*, **7**, 83–124, (1997).