ADAPT: Abstraction Hierarchies to Succinctly Model Teamwork

(Extended Abstract)

Meirav Hadad[1] and Avi Rosenfeld[2]
[1]Department of Computer Science, Bar Ilan University, Ramat Gan 52900, Israel
[2]Department of Industrial Engineering, Jerusalem College of Technology, Jerusalem 91160, Israel
Meirav.Hadad@elbitsystems.com, rosenfa@jct.ac.il

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Experimentation

Keywords

Simulation techniques, tools and environments, agent cooperation

1. ABSTRACT

In this paper we present a lightweight teamwork implementation through use of abstraction hierarchies. The basis of this implementation is ADAPT, which supports **A**utonomous **D**ynamic **A**gent **P**lanning for **T**eamwork. ADAPT's novelty stems from how it succinctly decomposes teamwork problems into two separate planners: a **task** network for the set of activities to be performed by a specific agent and a separate **group** network for addressing team organization factors. Because abstract search techniques are the basis for creating these two components, ADAPT agents are able to effectively address teamwork in dynamic environments without explicitly enumerating the entire set of possible team states. During run-time, ADAPT agents then expand the teamwork states that are necessary for task completion through an association algorithm to dynamically link its task and group planners. As a result, ADAPT uses far fewer team states than existing teamwork models. We describe how ADAPT was implemented within a commercial training and simulation application, and present evidence detailing its success in concisely and effectively modeling teamwork.

2. TECHNIQUE DESCRIPTION

ADAPT's model is based on decomposing teamwork problems' task and group elements in a top-down manner from a high level to progressively lower levels. Specifically, a given teamwork problem is converted into two hierarchical networks: a **task** network to model the set of activities a given agent can perform and a separate **group** network for addressing organization factors. Within both hierarchical networks, behaviors are decomposed such that the general task and group problems are progressively redivided into partial plans involving smaller sets of subtasks and subgroups. ADAPT contains two novel elements designed to further reduce the size of these hierarchies. First, as hierarchical abstraction is used, agents incrementally elaborate only relevant task and group information during task execution. Second, ADAPT uses an association algorithm to effectively perform task allocation. Agents only check those constraints which it may possibly perform, further adding to ADAPT's concise nature. The net result is that ADAPT can effectively implement teamwork problems, even in dynamic environments, yet uses far fewer states than existing approaches.

The planning strategies of the elaboration processes of each network in ADAPT are based on abstract search techniques [3]. Accordingly, the planning procedures of each elaboration process involves three major steps: (1) A *branching* step identifies possible candidates for expanding a partial plan; (2) A *refinement* step for adding constraint information to the partial plan; (3) a *pruning* step for removing unpromising candidates based on these constraints in order to avoid failures. While abstract-search is a well known technique for automated task planning [3], ADAPT's contribution stems from applying these techniques to teamwork modeling.

Cite as: ADAPT: Abstraction Hierarchies to Succinctly Model Teamwork (Extended Abstract), Meirav Hadad and Avi Rosenfeld, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

3. MOTIVATING EXAMPLE

Assume that a group must work as a team on a joint mission, say to capture a flag. A group of blue agents must plan how they will infiltrate the territory of the opposing team of red agents that are defending the flag. In dynamic environments it is almost impossible to predict all possible event permutations that may occur while the blue agents complete their task.

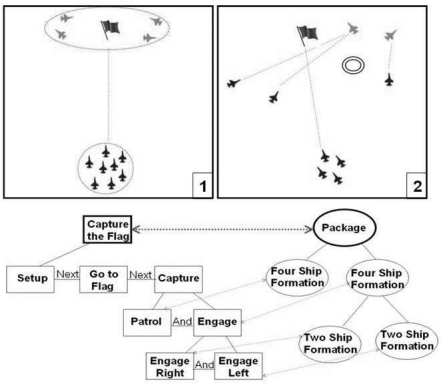

Figure 1: Stages in a Team Mission

Figure 1 depicts group states during the execution of the Capture the Flag mission. At the start, a group of 4 red agents are divided into 2 subgroups of pairs located on either side of the flag to defend it (see the top left corner). At the same time, a group of 8 blue agents approach the flag area. In the second stage, the blue group splits into two subgroups of 4 agents according to their capabilities. One subgroup splits again into two subgroups of 2 agents and each

subgroup approaches and engages the 2 red subgroups. However, during this stage an unplanned event occurs, and one of the blue agents is incapacitated by a red team member. Consequently, the blue team must replan their mission with only 7 of the 8 agents. In the final stage (top right corner), we see the group of 7 remaining blue agents still completing the task and capturing the flag.

We depict the networks of the teamwork model formation for the blue team in the bottom of Figure 1. ADAPT decomposes teamwork into both task and group networks. In the first stage each of these components are only described generally as one abstract node (the bold vertices at Figure 1). To graphically differentiate between the two task and group abstractions, we present the task hierarchy in rectangles, and the group hierarchy in ovals. At the beginning of execution, one rectangular task node describes the high level "Capture the Flag" task, and the group hierarchy "Package" describes the blue agents' attributes and capabilities that can be used to perform this task. In order for the blue agents can perform the team task, "Capture the Flag", their group and task planners must decide exactly how they will properly connect these two hierarchies.

4. MODELING ADAPT'S NETWORKS

ADAPT contains many similarities to previous Hierarchical Task Network (HTN) planning approaches [3]. Formally, we define an *atomic task* in ADAPT as an action $act(\vec{v})$ that can be directly executed by the agents (e.g., $FlyTo(origin, dest)$). A (higher-level) *complex task* $c(\vec{v})$ is one that cannot be executed directly and is decomposed into subtasks. To execute a high-level complex task $c(\vec{v})$, agents must identify a *method* which encodes all constraints for how this task including key information about who and how it can be performed. We define a method, m, as a 5-tuple containing: $\langle \text{name}(m), \text{task}(m), \text{constr}(m), \text{subtasks}(m), \text{relation}(m) \rangle$, where $\text{name}(m)$ is the name of the method, and $\text{task}(m)$ is the name of the complex task. We define $subtasks(m)$ as the sequence of tasks and $constr(m)$ as the set of constraints $\{\rho_1 \dots \rho_p\}$ that may apply when using the method m. Each constraint ρ_k involves a subset of variables and specifies all combinations of values for these variables. We define these variables as the set of $\{X_1 \dots X_n\}$ where each value X_i is taken from a given domain D_i with a set of possible values. Constraints may include specific required capabilities that a certain number of agents perform a specific $subtasks(m)$. The relationship between subtasks, $relation(m)$, contains constraints on the execution of the $subtasks(m)$ and may be one of the following: (i) AND; (ii) OR; and (iii) NEXT.

In parallel to the task hierarchy, ADAPT also deconstructs teamwork into a group component to model constraints about which agents can perform given tasks. We refer to the hierarchy about the entities' combined capabilities as the **group**. Parallel to our task definitions, we decompose the hierarchy as per the **group decomposition** into higher levels of **complex entities** and **atomic entities** which cannot be divided into further levels.

We define two separate networks d_{task} and d_{group}. A *network* $d_i = [G_i, \rho_i]$ is defined as a collection of items i that have to be accomplished under constraints ρ_i (the item i denotes the type of the network, i.e., group/task). Network d_i is represented by an acyclic digraph $G_i = (V_i, E_i)$ in which V_i is node set, E_i is the edge set, and each node $v \in V_i$ contains an item i. The *Planning domain* $\mathcal{D}_i = (\mathcal{M}_i, \mathcal{A})$ consists of library methods $\mathcal{M}_i$ and library $\mathcal{A}$ of atomic items. A *task planning problem* is defined as a triple $P_{task} = \langle d_{task}, \mathcal{B}, \mathcal{D}_{task} \rangle$, where d_{task} is the task network to be executed, $\mathcal{B}$ is the initial state and $\mathcal{D}_{task}$ is the planning domain. A *task plan* is a sequence $act_1 \dots act_n$ of atomic actions. A *group planning problem* is defined as a triple containing P_{group} defined as $\langle d_{group}, \mathcal{B}, \mathcal{D}_{group} \rangle$ where d_{group} is the group network to be executed, $\mathcal{B}$ is the set of agents with their concrete capabilities and $\mathcal{D}_{group}$ is the planning domain. A *group plan* assigns agents to the appropriate nodes in the group network based on their capabilities in such a way that all the constraints are satisfied. Given either task or group planning problem instance, the planning process of each of them involves the branching, refinement and pruning steps.

The branching step is defined by retrieving the entire set of methods in $\mathcal{M}_i$ which may be applied to the required item. Refinement then has each local agent check its $constr(m)$ and sends what it considers to be its best option to the mediator agent within the DCOP solver. In ADAPT's pruning stage, the mediator uses the OptAPO algorithm (see [2]) to search for this teamwork solution. If a solution for $\mathcal{M}_i$ cannot be constructed the mediator agent asks each agent to iteratively selects its next possible method until a solution is found. This process can either result with a plan being found, or a NULL plan in failure.

5. IMPLEMENTATION AND RESULTS

We have implemented ADAPT within a commercial training and simulation system at Elbit Systems Ltd. Specifically, we applied the general technique in Section 3 regarding the Capture the Flag problem to scenarios involving fighter jets attempting to destroy an enemy target. Each scenario involved a target that needed to be destroyed, as well as groups of attacking and defending planes. We relied on a group of professional fighter pilots to provide details about how they would perform theoretical missions. We then encapsulated this information to form ADAPT's networks.

	BITE		**ADAPT** max		**ADAPT** average	
# of Agents	Task	Group	Task	Group	Task	Group
5	561	18	44	5	37.1	3.67
8	624	146	53	8	39.65	6.29
12	829	400	68	8	56.86	6.17

Table 1: Comparing the number of task and group teamwork states in ADAPT versus BITE teamwork models

To study the savings in the number of states within ADAPT versus other previous BITE static approaches [1], we focused on missions with groups of 5, 8 and 12 blue planes which needed to destroy one target on the red team guarded by a fixed number of 5 jets. We recorded the number of task and group nodes required to encode teamwork within ADAPT throughout the task's execution versus BITE. As Table 1 demonstrates, we found that ADAPT's use of abstraction yielded an enormous savings in the number of teamwork states needing to be stored and represents a radical departure over previous models which need to exhaustively describe all possible interactions prior to task completion [1]. ADAPT builds teamwork models incrementally during task execution, thus allowing agents to apply refinement and pruning steps in order to limit the size of the teamwork model which needs to be stored. This fundamental difference not only yields teamwork models that are smaller by several orders of magnitude, but allows agents to quickly find their optimal behavior within this smaller model.

6. REFERENCES

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

[2] R. Mailler and V. Lesser. Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems. In *AAMAS '04*, pages 446–453, New York, 2004.

[3] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.