Towards Flexible Teamwork in Behavior-Based Robots: Extended Abstract

Gal A. Kaminka[*]
The MAVERICK Group
Computer Science Department
Bar Ilan University, Israel
galk@cs.biu.ac.il

Inna Frenkel
The MAVERICK Group
Computer Science Department
Bar Ilan University, Israel
frenkei1@cs.biu.ac.il

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Coherence and coordination

General Terms

Algorithms, Design

Keywords

Multi-Robot Systems, Multi-Agent Systems

1. INTRODUCTION

Research into teamwork in autonomous robots is quickly gaining significant interest, motivated by future applications of multi-robot teams. A key challenge is to automate the control of teamwork, such that the designer can focus on the taskwork to be done. *Teamwork architectures* have been proposed to automate the interactions between team-members, in order to facilitate robust and speedy deployment. Existing architectures provide important teamwork services to deployed teams, such as synchronized task execution [4], and task allocation [3, 1, 5]. These allow the designer to spend most efforts on building controllers for the specific tasks to be carried out, rather than the collaboration of the robots.

However, existing architectures leave important challenges open when applied to multi-robot teams. First, most existing robot teamwork architectures do not address both synchronized task execution and dynamic task allocation in a single architecture. Thus the team's developer must make a choice as to whether synchronization or allocation is more important. Second, with rare exceptions, existing architectures are monolithic, in the sense that they commit to using a fixed interaction protocol throughout task execution, e.g., confirm-request for synchronization [4]. Finally, while previous work in robotics have begun to explore the use of task-allocation in robot teams [3, 1], lessons from the application of multi-agent techniques to robotics have not been generated.

This extended abstract briefly presents BITE (*Bar Ilan Teamwork Engine*), a novel behavior-based teamwork architecture targeting physical robot applications, and addressing the open challenges discussed above. Similarly to previous architectures ([4, 5]),

[*]Also affiliated with Carnegie Mellon.

Figure 1: Robots using BITE to move in formation.

BITE maintains an organization hierarchy and a task/sub-task behavior graph to manage teamwork. However, in addition, BITE maintains a novel third structure, a set of hierarchically linked social interaction behaviors implementing interaction protocols for synchronization and task allocation. The key idea in this separation of the interaction behavior from the architecture is to make team control flexible.

Indeed, BITE implements a teamwork micro-kernel approach, in which different synchronization and allocation protocols can be used interchangeably and mixed as needed. This allows BITE to integrate and synthesize many features of existing teamwork architectures, and offer novel features. In addition, BITE incorporates features that stem from lessons learned in applying multi-agent systems technique in physical robots. In particular, BITE emphasizes pre-processing of sensory information before usage (including fusion of information from multiple robots), and facilities for human operator control.

BITE has been fully implemented and used with teams of Sony AIBO platforms (Figure 1). Extensive experiments with BITE has shown that BITE's novel separation of social interaction from task-oriented control is a significant contribution, in the sense that it accounts for non-trivial effects on team performance. We additionally discuss lessons learned in applying multi-agent systems techniques to robots. Full details are available in [2]

2. BITE: STRUCTURES AND CONTROL

Given the popularity of behavior-based control in existing robotics work, BITE uses a hierarchical *task behavior-graph* as the basis for a representation underlying the controllers for the team members. To this, it adds two additional structures: a set of social interaction behaviors, and an associated organization-hierarchy.

The task behavior-graph specifies the sequential and hierarchical relationships between task-oriented behaviors. It is an augmented connected graph, in which task-oriented behaviors are vertices, and edges are of two types: *Sequential* edges that specify temporal order of execution of behaviors; and *task-decomposition* edges, which allow a single higher-level behavior to be broken down into execution chains containing multiple lower-level behaviors. As with previous teamwork architectures, each robot executes its own copy of the behavior graph. Behaviors whose execution is to be coordinated in some fashion (henceforth, *team behaviors*) are tagged in advance by the designer.

BITE automatically takes actions to select and de-select team-behaviors in different robots, when appropriate. When a choice is available as to what team-behavior is to be executed next, BITE's synchronization services are triggered. Similarly, when a choice as to which decomposition behavior should be selected (i.e., which sub-task to execute), BITE triggers is task-allocation services, to make sure that the selection of a sub-task is coordinated with the tasks allocated to other robots. Finally, when a team-behavior's termination conditions are satisfied for a robot, BITE is triggered to coordinate the termination of this behavior with the other robots.

BITE therefore needs to maintain knowledge about the robots that are responsible for coordinated execution of team behaviors. To do this, BITE maintains a second structure, the organization hierarchy (called the team hierarchy in [4, 5]). This is a DAG (Directed Acyclic Graph) whose vertices are associated with sub-teams of agents, and whose edges signify sub-team-membership relationships. Vertices correspond to multi-robot sub-teams of robots and are connected such that if there exists an edge $< R_1, R_2 >$, then $R_2 \subset R_1$. The team hierarchy thus forms a partial lattice, from the root team R which includes all team-members, to sub-teams corresponding to each of the members by itself (i.e., to the individuals in the organization).

To allow behaviors to reason about the organizational unit responsible for their execution, we create links between the behavior graph and the team hierarchy, such that there is a link from a behavior B_j to a sub-team R_i if B_j is to be executed by R_i. Similarly, to allow reasoning about allocated tasks, we link sub-teams to the behaviors they are responsible for, such that there's a link from a sub-team R_i to behavior(s) B_j if R_i is responsible for B_j. Using these links between the behavior graph and the team hierarchy, a robot executing a behavior may easily find out whom it should contact in order to coordinate execution of this behavior.

To carry out the interactions themselves, BITE maintains a novel third structure, holding a set of *social interaction behaviors* which control inter-agent interactions. Interaction behaviors execute the synchronization and task allocation protocols (e.g., voting) that govern coordinated activity. Each interaction behavior is encoded in a separate behavior graph. For instance, a simple interaction behavior implementing voting synchronization may be decomposed into four atomic interaction behaviors, executed in sequence: Announce vote, send votes, tally votes, and announce winning selection. The addition of the social interaction behaviors, separate from the task-oriented behavior graph, allows the use of different interaction protocols at different times, depending on the team behaviors in question (and other context information).

The main control loop for BITE's behavior selection and execution uses *a behavior stack*—root behavior to leaf—where top behaviors on the stack are executed simultaneously with their currently selected children. First, the root (initial) behavior is put on an execution stack. Then the algorithm loops over four phases in order: (i) It recursively decomposes the top behavior on the stack into its children, allocating them to sub-teams if necessary (by calling an allocation social behavior). (ii) It then executes the behavior stack in parallel, waiting for the first behavior to announce termination. All descendants of a terminating behavior are popped off the stack, and then (iii) synchronized termination takes place. This can result in a newly-allocated behavior within the current parent context, in which case, it will be put on the stack for decomposition. Otherwise, (iv) this indicates that the robot should select between enabled sequential transitions from the terminated behavior (typically, calling a synchronization social behavior in the process). The process then repeats.

3. LESSONS LEARNED

BITE's design, including its introduction of social interactions as first-class objects, incorporates several lessons learned in using multi-agent techniques in multi-robot systems.

Lesson: Flexibility Makes a Difference. BITE allows for a significant degree of flexibility in teamwork over existing architectures. For instance, it allows creation of systems that marry the capabilities of different architectures. It also allows different interactions to be used depending on context. The robots, running BITE, have been in used in several tasks. In experiments (reported elsewhere [2]) we show that the ability to mix and match interaction behaviors can be a significant factor in task performance, lending support to this novel feature being an important contribution.

Lesson: Automated Teamwork Saves. By implementing the behaviors in BITE, the burden of worrying about coordination is put on the robots. For instance, in a coordinated movement task (movement in formation) when one robot loses track of the lead robot, the appropriate termination condition is satisfied, and the behavior terminates. This in turn triggers the appropriate social interaction behavior in BITE, which causes the other robots to also stop executing their movement behavior. Thus the team automatically starts and stops its movement together.

Lesson: Fuse and Process Sensor Information. A key lesson we have learned touches on the inherent unreliability of sensing in physical robots, compared to software agents. Teamwork architectures for software agents typically do not need to reason about whether their perceptions are truthful, only whether they have sensed all that they need to. This is most definitely not true in physical domains, were sensors may not only fail to detect a feature, but may simply lie about it. To address this, BITE pre-processes sensor data before it is handed off to task and interaction behaviors. It also facilitates fuses information from multiple robots, to mitigate both incorrectness as well as incompleteness.

Future efforts. We will focus on human-team interactions, and on extending BITE's capabilities towards greater fault-tolerance. We also plan to investigate the use of BITE in multiple robotic platforms, and a variety of tasks.

Acknowledgments. This research was supported in part by BSF grant 2002401.

4. REFERENCES

[1] M. B. Dias and A. T. Stentz. A free market architecture for distributed control of a multirobot system. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS-6)*, 2000.

[2] G. A. Kaminka and I. Frenkel. Flexible teamwork in behavior-based robots. In *AAAI-05*, 2005.

[3] L. E. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, April 1998.

[4] D. V. Pynadath and M. Tambe. Automated teamwork among heterogeneous software agents and humans. *JAAMAS*, 7:71–100, 2003.

[5] T. D. Vu, J. Go, G. A. Kaminka, M. M. Veloso, and B. Browning. MONAD: A flexible architecture for multi-agent control. In *AAMAS-03*, 2003.