Coordinated Teams of Reactive Mobile Platforms[1]

J. Sweeney, TJ Brunette, Y. Yang[2], R. Grupen
Laboratory for Perceptual Robotics
Department of Computer Science
University of Massachusetts, Amherst

Abstract

This paper presents techniques for exploiting redundancy in teams of mobile robots. In particular, we address tasks involving the kinematic coordination of several communicating robots. Teams are modeled as highly redundant spatial mechanisms for which multi-objective, concurrent controllers are constructed using a generalization of null-space control. The goal is to develop a methodology in which the robustness and error suppression in a control theoretic substrate can be used to preserve critical properties in teams of reactive robots. The resulting "safe" control options can then be explored while guaranteeing global compliance with system specifications. The proposed architecture depends on a set of concurrent, low-dimensional control processes that interact in a well-defined manner. Cascaded null space projections and coordination templates are used to manage control interactions across platforms that actively maintain constraints for pairs of robots. Pairwise policies can then be combined to represent coordinated, multi-robot tasks. To illustrate the approach, we demonstrate a distributed control that maintains critical connectivity in line-of-sight communication networks.

Keywords: *constraint-based programming, hybrid systems, reactive control*

1 Introduction

We propose a hybrid robot control strategy for multiple interacting robots. Swarms, especially biological swarms, have been noted to produce interesting behavior using massively distributed control algorithms (termites, ants, bees, etc). Distributing swarm behavior over many individuals can be very cost effective and the swarm can be robust to failure in individuals. However, the flexibility and reconfigurability of such an approach is challenging as there do not exist adequate methodologies for programming a swarm to do many different tasks. An n-robot team of mobile platforms can be modeled as a $2n$ dimensional path planning problem, but with predictable scalability issues. Naive implementations can require exponential increases in compute time when an additional robot is added. Moreover, run-time environments can change quickly, so that plans must cover a large variety of possible run-time contingencies to be useful. Finally, centralized solutions can produce globally optimal solutions in principle, but they do not in practice - due in part to the preceding issues. However, distributed controllers might be used to produce practical and scalable team controllers if formalisms can be developed that provide feasible and correct solutions initially that adapt toward optimal coordination policies incrementally. Our approach aims to provide cost effectiveness while providing rich programmability and a flexible run-time framework. We overlay inter-robot communications and control interactions that produce favorable (and correct) group dynamics when stimulated by the environment.

Currently there is a great deal of interest in adaptive control architectures for non-stationary, nonlinear processes [13, 12, 4]. Recent approaches postulate a family of local models that can be used to approximate the optimal, global control surface. *This implies that robots - even single robots - are expressed as collections of interacting agents.* This class of approaches generally rely on local linear models and are applied to simple regulation or tracking tasks. By switching controllers, or by reformulating local models, a linear control substrate can be applied more generally to nonlinear and/or non-stationary tasks. As a result, the robot control program is generally more robust. But so far, implementations based on this framework are not capable of guaranteeing distributed behavior in a multi-robot system. This paper contributes techniques that can be used to provide "best-effort" guarantees that important global properties will be preserved in distributed behavior. "Best-effort" means that subordinate control actions are projected into the null space of global specifications so that incorrect interactions are eliminated. By so doing, a distributed controller can address multiple objectives simultaneously without compromising critical performance guarantees. Under these circumstances, it is possible to assert that critical objectives can be actively maintained by primary controllers against environmental perturbations and interactions with other concurrent controllers.

[1]This work was supported in part by NSF CDA-9703217, DARPA/ITO DABT63-99-1-0022 and DABT63-99-1-0004.
[2]Mr. Yang is now at Cisco Systems, San Jose, California.

2 Related Work

Subsumption programming has been used for reactive robots in a behavior-based framework. Some such approaches advocate learning prerequisite skills that solve predefined subproblems and then combining them in a subsumption or voting framework [9]. Some use previously designed behaviors as primitives within the learning framework [10]. The approach presented here falls into this general category. Subsumption, however, is based on a largely procedural model of behavioral interaction designed by the system programmer that does not support global assertions regarding system behavior.

The Autonomous Robot Architecture (AuRA), developed by Arkin *et al.* at Georgia Tech [2], represents behavior in the form of perceptual and motor schemas. Individual behaviors are run as asynchronous, concurrent processes representing high-level behavioral intentions. Behavior is crafted as weighted, linear combinations of non-linear motor schemas. This paper extends this class of approaches by addressing the range of possible interactions between asynchronous schema. Our approach is couched in a control theoretic framework and organized using a discrete event structure. Such an approach can provide performance guarantees and leads to a reusable basis for behavior designed to be applicable in a wide range of applications and with a variety of multi-objective tasks.

A robot is *redundant* if it possesses more controllable degrees of freedom than are required to achieve a reference configuration. Redundant systems may have an infinite number of solutions for a given task. The system Jacobian for such a system is redundant so that rows and columns are no longer linearly independent (and the Jacobian is no longer square). Consequently, a *null space* can be identified in the manipulator Jacobian in which motions produce no progress toward the goal. For a forward kinematic transformation, the null space of the Jacobian at a given location is referred to as the *self-motion manifold*. The Moore-Penrose generalized inverse (or pseudoinverse) of a redundant Jacobian selects the minimum length solution among all candidate solutions. A null space trajectory can be chosen to produce internal motions that avoid kinematic singularities that address force and velocity constraints, or that optimize the kinematic condition of the transformation manipulator with respect to generic cost functions [6, 11]. In general, any configuration space trajectory in service to a subordinate control objective can be projected onto the null space of superordinate objectives.

3 Redundant, Multi-Robot, Navigation Controllers

The techniques above can be generalized to any control formulation that can be linearized locally to produce a control action (the negative gradient of the artificial potential) and an orthogonal null space defined by the "level-curve" on the artificial potential function. We employ these techniques to preserve constraints between multiple robots running concurrent controllers whose actions may conflict.

Control Primitive $\{\phi_i^{g_j}\}$ $i, j \in R$
A single controller is an association of an artificial potential, ϕ, and effector resources, i and sensor feedback, g_j, drawn from resource pool R. Sensor feedback includes goals and/or obstacles derived from robot j that may have been observed directly by robot i or communicated from robot j.

Coordination Primitives $\phi_i \triangleleft \phi_j$ $i, j \in R$
A coordinated pair of control processes can reside on the same mobile platform ($i = j$), or may be distributed across platforms ($i \neq j$). These processes interact through the $\triangleleft$-"subject-to" operator which enforces a local null space trajectory. In our application, we employ a path planner based on harmonic potentials [7]. The level curves in the harmonic potential of the dominant controller define its null space globally and introduces a nonholonomic constraint into the configuration space of subordinate controllers. If, for example, ϕ_j represents a property that the system must actively maintain, and that property depends on the proximity to robot i, then robot i may only move inside of the null space of ϕ_j - actions in robot i must not cause robot j's potential to increase. If ϕ_i is a path toward and goal g, and ϕ_j is actively maintaining the line-of-sight between robots i and j, then ϕ_i must project its gradient onto the null space of ϕ_j in order to guarantee that its actions will not disturb robot j's objective.

3.1 Coordinating Multiple Robots

In this section, we will introduce a class of distributed solutions which have a common format. Each is composed of a combination of two path controllers - one that preserves a kinematic line-of-sight (LOS) relationship between the two robots and another that executes a path to the reference configuration, g. Line-of-sight is an important kind of constraint to consider because it is an important subgoal for communicating robots in a distributed control environment. If robot i is the "leader" (headed toward the goal g), this set of controllers can be written:

$$\Phi|^g_{\{i,j\}} = \{\phi|^g_i \triangleleft \phi|^{LOS_{ij}}_j, \phi|^g_i \triangleleft \phi|^{LOS_{ji}}_i\} \qquad (1)$$

These pairwise control options are pictured schematically in Figure 1. If we permit the leader/follower roles to be reversed, there would be four possible elements of the set $\Phi|^g_{\{i,j\}}$.

The first option in Equation 1 states that controller $\phi|^g_i$ will move robot i to the external goal g by descending a harmonic potential, ϕ. It does so in a manner that

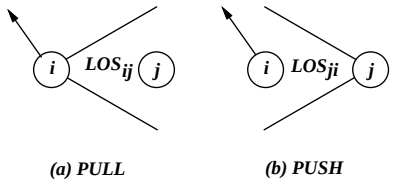

Figure 1: Two asymmetric configurations of the pairwise LOS controller.

does not disturb the constraint expression $\phi|_j^{LOS_{ij}}$ by the $\triangleleft$ -"subject-to" constraint. This is the leftmost control configuration in Figure 1, deemed the *pull* primitive. The complementary configuration shown in Figure 1(b) uses the LOS_{ji} region to represent the LOS constraint. This configuration is referred to as the *push* configuration.

Figure 2 shows a sequence of frames derived from our implementation of a pull coordination primitive for use as a simple, two robot leader/follower controller on our UMass UBot platforms. The harmonic potential of both robots are updated continuously, and the controllers are recomputed periodically. In our implementation, this happens at between 2 and 3 Hz.

3.2 Estimating LOS Regions and Determining LOS Goals

Two types of goals are introduced with which to define the null space operator for interacting controllers. In Figure 3, there are two types of otherwise equivalent goals configurations: LOS goals, and occlusion threshold goals. If the dominant controller is safely inside the interior of the LOS goal set, then both controllers run concurrently under the management of the null space operator. If, however, the dominant controller finds itself on the edge of the LOS region (in the set of "occlusion threshold" goals), then the subordinate controller is disabled until the dominant controller enters the LOS goals once again. The relative size and position of LOS goal sets and occlusion threshold goals causes significant measurable variation in the performance of the coordinated pair. The relative size of the threshold region influences the aggressiveness of the follower's motion controller as shown in Figure 4.

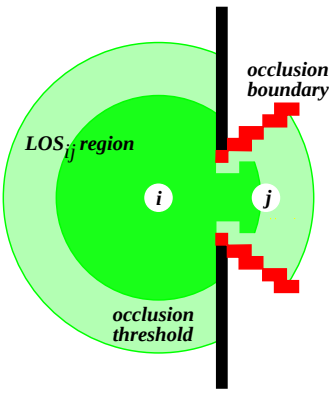

Figure 3: Two different types of goal configurations for defining subordinate controller activation.

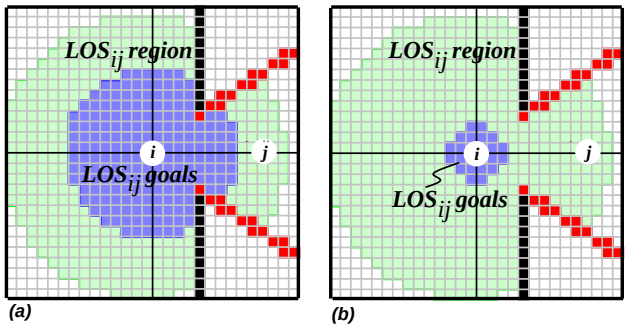

Figure 4: The *pull* configuration can select LOS goals in two qualitatively different ways; (a) a conservative follower that does not move until occlusion threatens, and (b) an aggressive follower that tracks the leader more closely.

Teams of $n > 2$ robots can assemble controllers from combinations of many *push* and *pull* control configurations that serve to coordinate pairs of robots. Figure 5 illustrates a relatively aggressive follower (robot 1) that follows the leader (robot 0) closely as it moves toward the goal, denoted by the black square in the lower right quadrant of the map. The grey lines between the robots represent the LOS property. Robot 2 is a stationary hub in this example that maintains a push relationship with robot 1. Only when the LOS from robot 2 to robot 1 is threatened, does robot 1 increase the following distance. Assuming robot i is the leader and is followed by robots j and k, there are four configurations between the two contiguous pairs of robots $\langle i, j\rangle$ and $\langle j, k\rangle$: *pull-pull*, *pull-push*, *push-pull*, *push-push*. This set represents all combinations of control options or $i - j - k$ sequences. If we permute the three robots, there are $6 \times 4 = 24$ possible coordinated triples that can guarantee that LOS will be preserved but are otherwise unsorted.

This approach scales to n robots by virtue of employing pairwise coordination primitives that bound the scope of inter-robot communications and whose per processor compute load is nearly evenly distributed for singly connected chains. Load can be balanced by noticing that the LOS regions required can be (1) computed directly using sensor data, or (2) constructed using parameters communicated between peers.

In team behavior, it may be each robot's prerogative to select a run-time strategy to satisfy global performance constraints. For instance, a robot whose battery is low may elect to adopt a less aggressive LOS policy. Every pair in a singly connected network topology may, in fact,

Figure 2: A two robot leader/follower coordination primitive in action. The lower robot is the leader and is moving right to left.

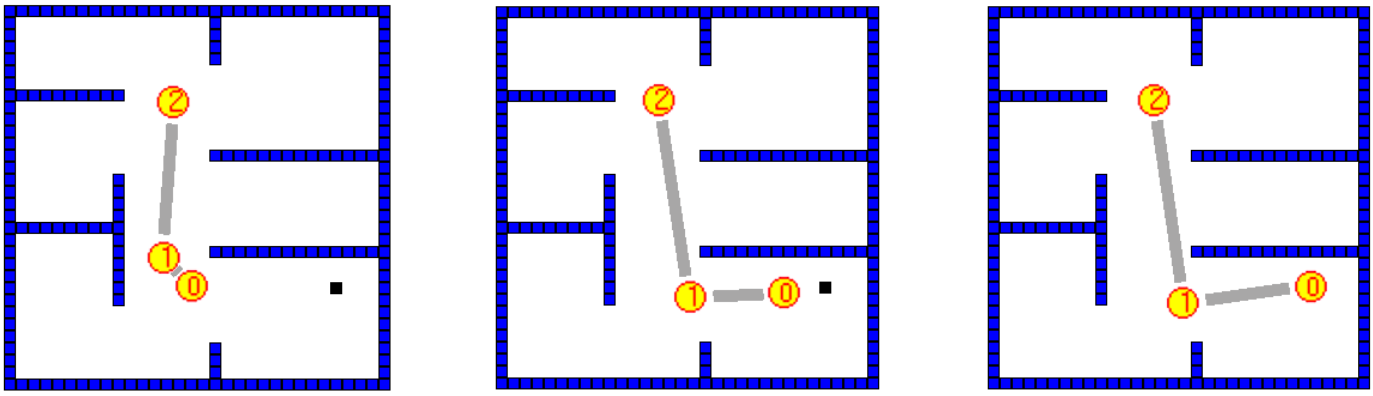

Figure 5: A typical aggressive *pull* behavior - the follower tracks the leader closely. The grey lines between the robots represent the LOS property.

choose to preserve the LOS specification in a manner appropriate for the local run-time conditions. In Figure 6, robot 0 is leader, and robot 1 selects an aggressive *pull* strategy for maintaining 0-1 line-of-sight. Robots 2 and 3 adopt a more conservative *pull* strategy. Robot 4, in this example, was designated a stationary host.

3.3 Sorting Equivalent LOS-Preserving Controllers

A simulator was used to test the performance of different versions of the coordinated *pull* strategy. The test environment was a simple office-style floor-plan such as those pictured in Figures 5 and 6. The position of the leader, the follower, a stationary host, and goal were randomly generated such that they formed an initially valid line-of-sight configuration. Goal locations were classified into two sets, based on the number of robots that would be needed to be active in a coordinated LOS behavior for the leader to reach the goal. Goals that can be reached using only one active robot maintaining LOS with the stationary host are denoted "one-robot" problems. Goals that required a LOS chain using two robots and one stationary host are called "two-robot" problems.

In each trial, the leader searched for the goal while line-of-sight was maintained throughout the team using the *pull* coordination primitive. By varying the occlusion threshold of the *pull* controller, three different levels of aggressiveness of the LOS behavior were chosen qualitatively, which we deemed AGGRESSIVE, NEUTRAL, and CONSERVATIVE. The time taken to reach the goal and the total energy consumed were recorded for each trial. Two sets of trials were performed. The first set used goals that were both one-robot and two-robot problems. The second set only selected goals that were two-robot problems. Two-robot goals could either be located far enough away from the leader to require the LOS chain, or they could be located behind an occluder. Figure 7 summarizes the results of running 100 trials for each set of goals, using the three variations of the *pull* primitive.

From these results, we can see that the AGGRESSIVE strategy took the least time in general, as we might expect, while the NEUTRAL configuration required less time than the CONSERVATIVE configuration. In the two-robot trials, where encounters with occluders happened more often, the time difference between the three styles of behavior was larger than in the first set of tasks.

In both sets of trials, AGGRESSIVE strategies took more energy in general, also as predicted. This trend is accentuated in the set of trials using both one- and two-robot problems, where some of the randomly placed goals are within LOS of the stationary host. In such a situation, conservative strategies can require only one robot to be

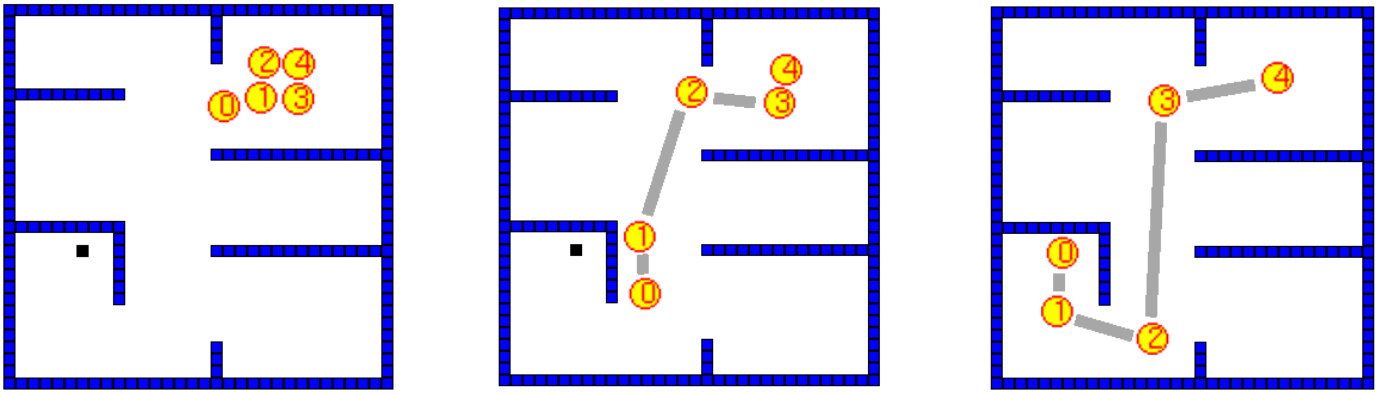

Figure 6: A sequence of aggressive and conservative *pull* controllers applied in a multi-robot situation. The goal is the solid black square in the lower left. The grey lines between the robots represent the line-of-sight property.

active, while aggressive strategies cause the extra robot to tag along with the leader unnecessarily, thus increasing the total amount of energy consumed.

3.4 Generalizing Network Connectivity

Since robots must interact, they must actively preserve network connectivity between peers. In this section, we continue to use the LOS kinematic constraint to represent connectivity and show how larger scale networks might be preserved. Equation 2 describes network connectivity in a network of k robots. The *push/pull* control configurations for a given pair of robots are arranged symmetrically in $\mathbf{G}$'s off-diagonal elements. An element $\mathbf{G}_k^n[i,j]$ is a Boolean variable asserting whether robots i and j are connected in n hops. *Push/pull* controllers that achieve LOS goals can be employed both to evaluate the current state of connectivity in the system, or to determine whether a new assignment of *push/pull* controllers among the constituent robots will be able to attain or preserve LOS connectivity.

$$\mathbf{G}_k^n = \begin{bmatrix} \phi_0^{los_0} & \phi_0^{los_1} & \dots & \phi_0^{los_k} \\ \phi_1^{los_0} & \phi_1^{los_1} & \dots & \phi_1^{los_k} \\ \vdots & \vdots & & \vdots \\ \phi_k^{los_0} & \phi_k^{los_1} & \dots & \phi_k^{los_k} \end{bmatrix}^n \qquad (2)$$

For example, Equation 2 can recruit k robots into a network by determining who among its peers are LOS connected and then protect those peer relationships by engaging an adequate complement of pairwise *push/pull* relationships. $\mathbf{G}$ defines the equivalence class of "network preserving" control options.

4 Conclusion and Discussion

In this paper we have proposed a formalism for representing control interactions in teams of mobile robots with excess degrees of freedom. We have demonstrated its use in tasks that require kinematic properties in the team. The Line-of-Sight (LOS) communication model is developed in some detail to illustrate the ideas proposed. We discussed how LOS constraints can be configured to be functionally equivalent, but optimized for different criteria depending on the state of the local robot. We present results in simulation showing that different LOS parameters produce different qualitative behaviors in otherwise equivalent control options.

In future research, we plan to expand our implementation to larger teams of UMass UBots and to use learning algorithms to develop policies for robots to choose among equivalent options based on observable state information. We are considering applications such as formation control, parallel search controllers, bounded overwatch localization (using subsets of the team to track movements and correct for odometry errors in another, possibly disjoint, subset), and network QoS guarantees.

References

[1] S. Akishita, S. Kawamura, and K. Hayashi. Laplace potential for moving obstacle avoidance and approach of a mobile robot. In *1990 Japan-USA Symposium on Flexible Automation, A Pacific Rim Conference*, pages 139–142, 1990.

[2] R.C. Arkin and T. Balch. Aura: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

[3] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, December 1991.

[4] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[5] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[6] S.L. Chiu. Control of redundant manipulators for task compatibility. In *Proceedings of the 1987 Conference on Robotics and Automation*, volume 3, pages 1718–1724, Raleigh, NC, April 1987. IEEE.

[7] C. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotics Systems*, 10(7):931–946, 1993.

[8] A. Gelb, editor. *Applied Optimal Estimation*. Technical Staff — The Analytical Sciences Corporation, The MIT Press, Cambridge, MA, 1986.

[9] J. Hoff and G. Bekey. An architecture for behavior coordination learning. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 2375–2380, Perth, Australia, November 1996. IEEE.

[10] P. Maes and R. Brooks. Learning to coordinate behaviors. In *Proceedings of the 1990 AAAI Conference on Artificial Intelligence*. AAAI, 1990.

[11] Y. Nakamura and H. Hanafusa. Optimal redundancy control of robot manipulators. *Journal of Robotics Research*, 6(1), Spring 1987.

[12] M.H. Raibert. *Legged Robots that Balance*. MIT Press, Cambridge, MA, 1986.

[13] A.A. Rizzi, L.L. Whitcomb, and D.E. Koditschek. Distributed real-time control of a spatial robot juggler. *IEEE Computer Magazine*, 25(5), May 1992.

[14] J. Rosenblatt. Damn: A distributed architecture for mobile navigation. In *Proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, Stanford, CA, March 1995. AAAI Press.

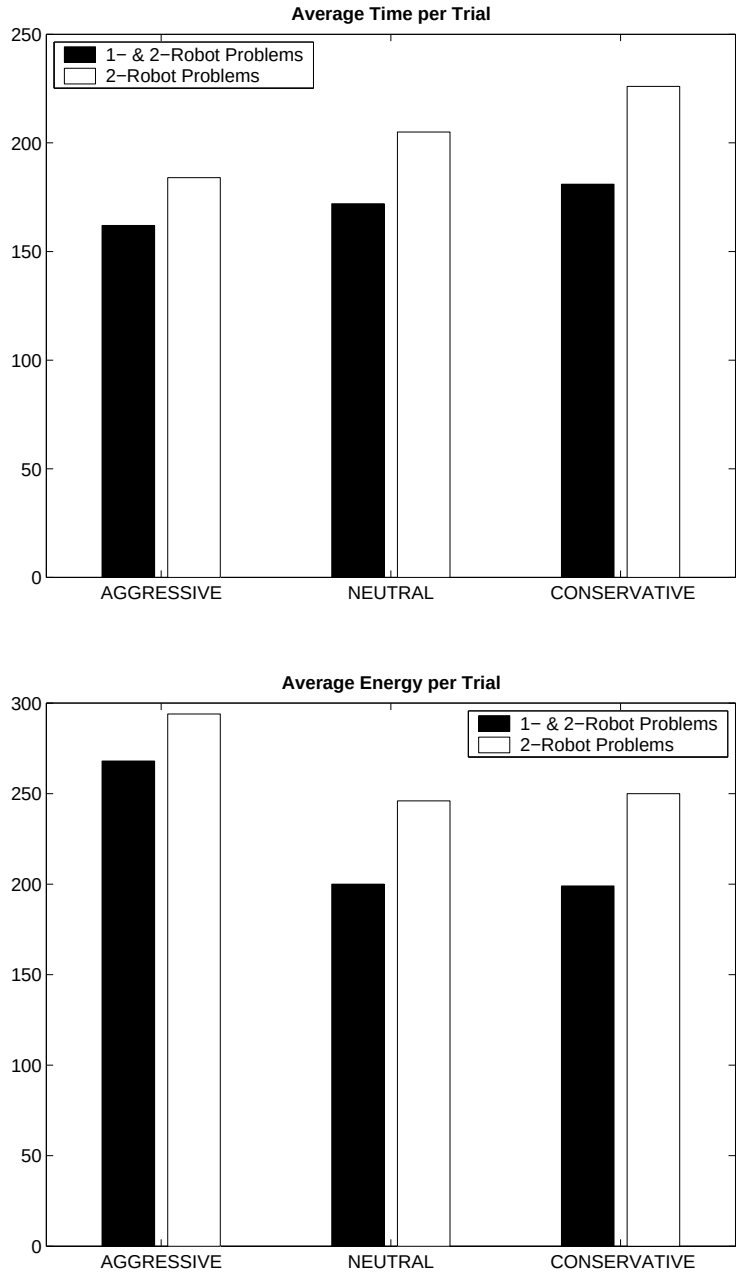

Figure 7: Average Time and Energy for 100 trials using aggressive, neutral, and conservative levels of aggressiveness in following with the *pull* coordination primitive. Each set of trials was run in both one- and two-robot problems and exclusively two-robot problems.