Tunably Decentralized Algorithms for Cooperative Target Observation

Sean Luke
sean@cs.gmu.edu

Keith Sullivan
ksulliv@cs.gmu.edu

Gabriel Catalin Balan
gbalan@cs.gmu.edu

Liviu Panait
lpanait@cs.gmu.edu

Abstract

Multi-agent problem domains may require distributed algorithms for a variety of reasons: local sensors, limitations of communication, and availability of distributed computational resources. In the absence of these constraints, centralized algorithms are often more efficient, simply because they are able to take advantage of more information. We introduce a variant of the cooperative target observation domain which is free of such constraints. We propose two algorithms, inspired by K-means clustering and hill-climbing respectively, which are scalable in degree of decentralization. Neither algorithm consistently outperforms the other across over all problem domain settings. Surprisingly, we find that hill-climbing is sensitive to degree of decentralization, while K-means is not. We also experiment with a combination of the two algorithms which draws strength from each.

1 Introduction

Cooperative target observation (CTO) problems are interesting testbeds for studying multi-agent coordination, planning, and robot control. These problems are interesting both because they are good examples of dynamic multi-agent interaction and emergent behavior; and because they have real-world application to robotics, security, and AI.

In this paper we use CTO as a problem domain for comparing the performance of centralized, partly-decentralized, or fully decentralized algorithms under different levels of dynamism and sensor capabilities. The algorithms presented are interesting in several ways. First, we have found them very effective solutions to the problem. Second, they are *tunably decentralized*, meaning that we can adjust them gradually from fully decentralized versions to fully centralized versions with a single parameter. This provides us with a useful tool for analysis of degree of decentralization. Third, surprisingly, in the CTO problem studied, the degree of decentralization has very different effects on otherwise reasonably comparable algorithms. It is *not* necessarily the case that decentralization is likely to perform worse, at least in the problem studied, as we would intuitively imagine due to its lack of complete information about the intentions of other agents.

The kind of CTO problem we are using is one in which mobile agents, (called *observers*) collectively attempt to stay within an "observation range" of as many targets as possible. The targets wander randomly and are slower than the observers. For purposes of this paper, the environment is bounded and clear of obstacles. CTO problems of this type have been popularized by Lynn Parker [15], and are sometimes known as CMOMMT ("Cooperative Multi-robot Observation of Multiple Moving Targets"). In Parker's configuration of the problem, an observer does not have a global view of all available targets to observe. This formulation of the problem strongly suggests a distributed and greedy control solution due to the lack of global information.

We have reformulated CTO in a slightly different fashion. In our problem domain, observers know the posi-

tions of all other observers and targets in the environment. This is not an unrealistic assumption for some real domains, where long-range radar may provide bearings for all targets of interest, and vision or other short-range sensors may dictate the observation range. However our primary reason for reformulating the problem in this way is to lift sensing constraints which strongly bias the problem towards distributed algorithms; this gives us an opportunity to study the degree to which global control is advantageous over varying degrees of dynamism and other environment variables.

In this work, we examine two algorithms for controlling the observers, based on K-means clustering and hill-climbing respectively. We also consider the two in combination: K-means clustering followed by hill-climbing. These three algorithms are tunably decentralized by adjusting a parameter which dictates how many *subsets* the observers are divided into. All observers within a given subset collectively participate in a separate, concurrent decision-making process. Thus one unified set yields a centralized algorithm, whereas many small subsets are decentralized. We note as an aside that the ability to work under different subset sizes conveys another advantage: the observers can work in environments where independent agents (humans perhaps) are also participating in the observation task. The observers can simply treat the humans as a separate "subset" and will attempt to take advantage of the presence of the human observers in the environment.

The paper is organized as follows. Section 2 discusses related work. Section 3 describes the simulation and the K-means and hill-climbing algorithms. Section 4 compares the algorithms and discusses the results. The algorithms are compared for different subset sizes to see how performance degrades as the targets become faster, the sensing radius decreases, and speed of the world increases relative to algorithm execution. Section 5 provides concluding remarks.

2 Related Work

There is considerable previous work in areas related to CTO. Many of these areas deal with multi-agent problems in dynamic environments with mobile agents. One example, robot foraging, asks a team of robots to collectively forage for pucks or cans, and to move them to specially designated areas. The efficiency of an approach may be defined by how quickly it completes the foraging task [6, 13], or by the number of items collected in a fixed amount of time. Related tasks include collective sorting and clustering [3, 7].

Foraging and clustering tasks do not generally require interaction and coordination: in fact, many can be performed by a single agent. In contrast, stick-pulling requires robots to cooperatively lift sticks out of holes, a task so arranged as to be impossible for a single robot to perform [9, 10, 12]. Another task requiring coordination is robot formation [1], where a team of agents must move across a field in minimal time without colliding with obstacles or other robots. Problems such as collaborative mapping [4, 8] can be performed by a single robot, but are useful for studying *how* the multiple robots collaboratively agree on which of several possible map interpretations is correct.

Parker ([15, 17]) has studied the form of CTO most similar to our work, termed CMOMMT ("Cooperative Multi-robot Observation of Multiple Moving Targets"). As discussed earlier, in CMOMMT a team of observer agents attempt to move within a given distance of as many targets at once. An agents' sensor range is limited, but it can additionally see targets within sensor ranges of certain nearby teammates. One approach to solving the problem is to use weighted force vectors applied by nearby targets, observer agents, and obstacles to guide agent movement. CMOMMT has been proven to be NP-hard [16], and has been demonstrated in simulation and on real robots [14].

Other techniques exists for controlling robots in a CTO domain. One approach is to include bargaining in the control algorithm [19]. Each robot is ranked according to its "eligibility" to perform each of N tasks, and robots then perform tasks according to preferred rankings. Lazy reinforcement learning has also been applied to the CTO problem, with favorable results [18].

Finally, CTO is related to the problem of multi-target tracking [2, 5], which is concerned with the generation of target tracks from data collected by non-geospatially located sensors. A typical application is air-traffic control, where the air-traffic control operators need a complete picture of aircraft tracks.

3 Model Description

Our version of CTO is a variation of CMOMMT which allows all observer agents to see all targets and other observers in the environment. The environment is a non-toroidal rectangular continuous 2D field free of obstacles. The model contains N observers and M targets, with $N < M$. Let O denote the set of all observers, and T denote the set of all targets. The observers can move in any direction. Each observer has an identical *observation range R*, and can observe any target which falls within a circle centered at the observer and of radius R. Figure 1 shows a snapshot of the model in action.

The targets move randomly throughout the space, and do not try to avoid the observers. Movement of both
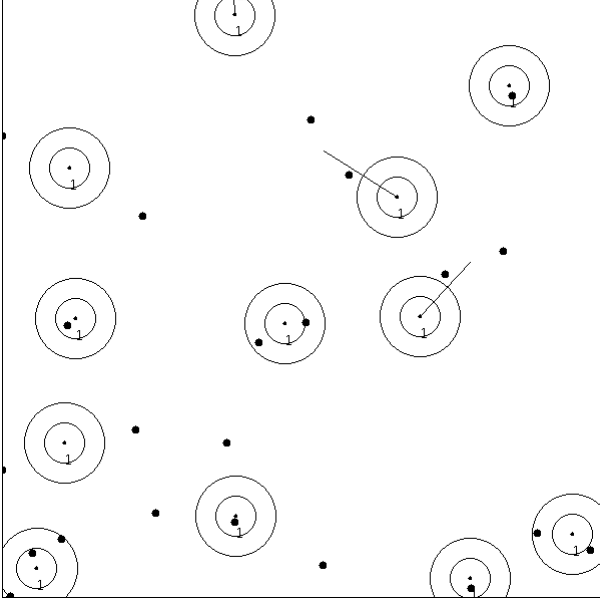
Figure 1: Screenshot of the model. Small doubly-circled dots are observers. Outer circles are their observation ranges. Large dots are targets. Straight lines connect observers with newly-chosen desired destinations.

the targets and observers is done by setting a destination point, then having the agent travel towards this point. The targets travel towards their destination point for at most 100 time steps. If they reach the destination before 100 time steps, then they compute a new destination point immediately. Targets' destination points are chosen at random from within a local region (one quarter of the environment height and width) centered on the target: the locality helps prevent targets from clustering near the center of the field on the way to their intended destinations.

Observers compute a new destination point every α time steps. If one reaches its destination point in less than α time steps, then it waits until a new destination point is computed. The destination point is determined using one of three cooperative target observation algorithms:

1. Hill-climbing with subsets

2. K-means clustering with subsets

3. K-means clustering with subsets followed by hill-climbing with subsets

Tunable decentralization of the algorithms works as follows. The N observers are divided into C disjoint subsets, and a separate instance of the chosen algorithm is run for each subset. Each algorithm instance adjusts the locations only of the observers in its subset, and presumes that all observers outside the subset, and all targets, are fixed in their current positions. A cluster size C/N may range from 1 to N: if the cluster size is N, then all observers participate in the same decision-making process, whereas smaller clusters yield higher degrees of algorithm decentralization.

3.1 Hill-climbing with Subsets

Hill-climbing iteratively improves candidate observer destinations by first copying the current locations into the initial candidate destinations, then performing 1000 iterations of the following operation:

1. Copy the current candidate observer positions and mutate the copy by randomizing the position of one observer member of the subset. The observer position is picked at random from the intersection of the field and a box centered at the observer with an initial width and height equal to one-half the environment width and height. The box decreases in width and height by 1% each iteration, but not below one tenth of the environment width and height.

2. If the mutated child is "better" than its parent, replace the parent with the child, else discard the child.

3. Goto 1

A candidates' quality is assessed by testing how it affects the *entire observer team*, not just the subset. We use a lexicographic ordering of various useful quality measures. Specifically, we accept the child and replace the parent with it using the following test:

1. Accept the child if it observes more targets.

2. Else if the child and parent observe the same number of targets, let compute

$$H = \sum_{o \in O} \min_{t \in T} f(o,t)$$

where

$$f(o,t) = \begin{cases} \min(R, \text{dist}(o,t)) & \text{if } R/2 \leq \text{dist}(o,t) \\ 0 & \text{if } R/2 > \text{dist}(o,t) \end{cases}$$

where $\text{dist}(o,t)$ is the Euclidian distance between the observer $o \in O$ and target $t \in T$. Accept the child solution if $H_{child} < H_{parent}$. This encourages the observers to center themselves among the targets they are observing and to not share targets.

3. Else if $H_{child} = H_{parent}$, and the child and parent observe the same number of targets, then compute

$$G = \sum_{o \in O'} \min_{t \in T'} \text{dist}(o,t)$$

where O' is the set of observers that do not observe any targets, and T' is the set of targets that are not observed by any observers. Accept the child solution if $G_{child} < G_{parent}$. This encourages the observers to move towards unobserved targets.

4. Else reject the child.

After the hill-climber has iterated 1000 times, then we have to determine which observer in the subset will go to which of the new candidate destinations. For this we used a simple greedy coloring algorithm.

1. Pick the candidate destination and the observer in the subset that are closest to one another

2. Assign the observer to that destination

3. Remove the observer and the destination from consideration

4. Goto 1

3.2 K-means Clustering with Subsets

K-means is a widely used clustering algorithm which assumes that the number of clusters is known *a priori*. Suppose that N points need to be grouped into K clusters, with the requirement that the mean distance from the points to the centers of the clusters is minimized. The algorithm works as follows

1. Select K initial centers for the clusters.

2. Assign each of the N points according to which of the cluster centers is closest to the point.

3. Reposition each cluster center slightly closer to the mean location of all the points assigned to that center.

4. Go to 2.

This process is repeated until no more significant progress is made, or until time is exhausted.

In order to apply the K-means algorithm to our cooperative target observation problem domain, we consider the targets to be the points to be clustered, and observer destinations as the candidate cluster centers. We set the initial positions of the cluster centers to the current positions of the observers, and assign each to the observer at that position. We modify the clustering algorithm so that the only candidate centers which are allowed to move (in step 3) are those assigned to observers which are members of the current subset.
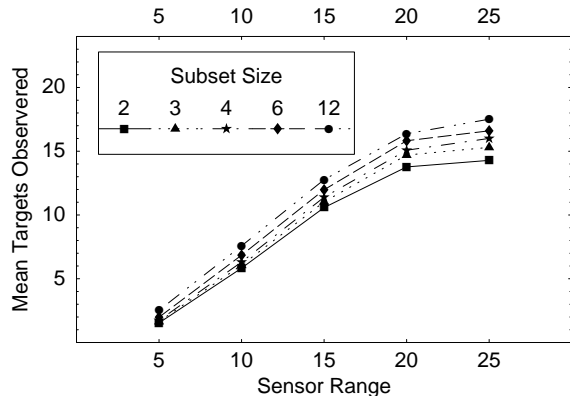

Figure 2: Performance of the hill-climbing algorithm using different subset sizes when varying the sensor range.

4 Experiments

We expected that both the K-means clustering and the hill-climbing algorithms would degrade in performance as the targets got faster, as the rate of algorithmic updates slowed, and as the size of the sensing radii decreased. This was consistently borne out in our results. However our primary interest was in seeing how different levels of decentralization would effect the degradation. As both algorithms are tunably decentralized in a similar fashion, we expected that both would degrade in the same way: but this was not at all the case. Below we discuss this result, followed by a comparison of the two algorithms against one another, and against their combination.

All experiments were done on the MASON simulation environment [11]. MASON is a fast discrete-event multi-agent simulation library core in Java, designed to be the foundation for large custom-purpose Java simulations, and contains a model library and an optional suite of visualization tools in 2D and 3D.

In performing the experiments, for each simulation run we gathered the mean over all timesteps of the number of targets under observation. If a target was observed by multiple observers, then we counted the target only once. In all experiments we held the following parameters fixed:

- Width and Height of Field: 150 x 150 units

- Timesteps per simulation run: 1500

- Number of simulation runs per data point: 30

- Number of targets: 24

- Number of observers: 12

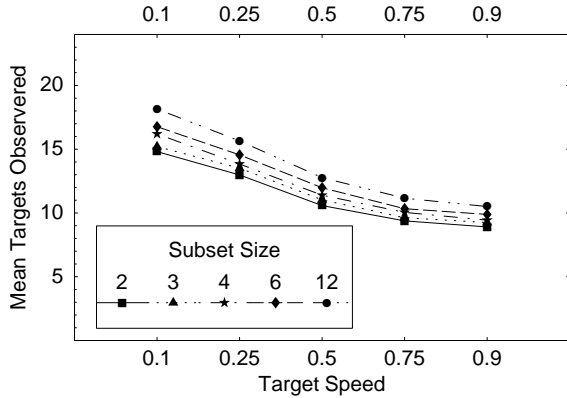- Speed of observers: 1 unit per timestep

Figure 3: Performance of the hill-climbing algorithm using different subset sizes when varying the target speed.

Figure 4: Performance of the hill-climbing algorithm using different subset sizes when varying the update rate.

In order to establish statistical significance when comparing the results, we used a series of Welch's two sample tests[1]. Additionally, we used Bonferroni's inequality to compensate for the large number of tests performed. As a consequence, each of our two sample tests was performed at confidence level 99.995%.

4.1 Results

Initial experiments compared K-means and hill-climbing against random and stationary behaviors. We found that K-means and the hill-climber are statistically better than the random and stationary algorithms over all combinations of target speed, subset size, range, and rate of updates.

Hill-climbing Figures 2, 3, and 4 show the performance of the hill-climbing algorithm as the range, target speed, and update rates vary, respectively. If not being varied, the sensor range is set to 15, the target speed to 0.5, and the update rate to 10. In each graph, we also change the degree of decentralization.[2] The results verify that performance significantly decreases when either range decreases, target speed increases, or update rate decreases. More importantly, the graphs show that increasing the degree of decentralization (reducing the subset size) leads to a decrease in mean number of targets observed. Notably though, the specific difference due to decentralization is almost invariant over any change in environment parameter.

Our findings are supported by statistical tests: we compared subset sizes of 2 and 12 and found significant differences across all range, target speed, and update rate settings. Additionally, we compared extreme settings of the range, target speed, and update rate for the same subset size: all tests showed significant differences.

K-means We ran the same experiments with the K-means algorithm, but it yielded a very different result. Like hill-climbing, K-means performed best at large ranges, small target speeds, and small update rates. However, K-means showed *almost identical performance* regardless of subset size. Decentralization appears to have had no effect on the results at all.

Comparison Given that hill-climbing degraded with subset size but K-means did not, we wondered how the two algorithms compared against one another. We directly compared K-means against hill-climbing across all combinations of domain parameters. We chose to use a subset of size 12 as it yielded the best results for hill-climbing. The results are shown in Table 1. In summary, hill-climbing was better with slower-moving targets, while K-means was better with faster targets.

This seems an intuitive result. When the targets are slow, or when the sensor range is small, hill-climbing can often discover superior solutions to K-means because K-means is centering observers in the middle of clusters of targets without considering how far away the targets in a cluster are from one another (see Figure 5). But as targets increase in speed, K-means clustering begins to outperform hill-climbing. Qualitatively, it is our observation that with fast targets hill-climbing cannot reach its "optimal" destination in time, and essentially chases targets around the field. But in many cases clusters form be-

---

[1]Welch's two sample test is a variant of Student's t-test for non-equal variances of samples.

[2]Due to time constraints, our smallest cluster size is 2, not 1 (fully decentralized). While this is unlikely to effect the results, we hope to add a cluster size of 1 in the final version of the paper.
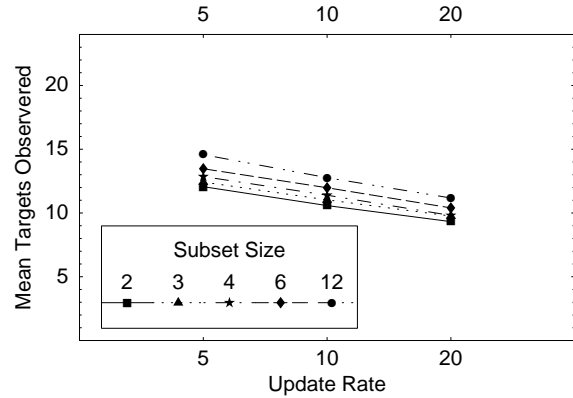
Update Rate	Sensor Range	Target Speed				
		0.1	0.25	0.5	0.75	0.9
5	5	≪	≪	≫	≫	≫
	10	≪	≪	≫	≫	≫
	15	≪	≪	≫	≫	≫
	20	≪	≪	≃	≫	≫
	25	≪	≪	≪	≪	≪
10	5	≪	≫	≫	≫	≫
	10	≪	≃	≫	≫	≫
	15	≪	≃	≫	≫	≫
	20	≪	≃	≫	≫	≫
	25	≪	≪	≪	≪	≪
20	5	≃	≫	≫	≫	≫
	10	≃	≫	≫	≫	≫
	15	≃	≫	≫	≫	≫
	20	≃	≫	≫	≫	≫
	25	≪	≪	≃	≃	≃

Table 1: K-means versus hill-climbing, subset size of 12. ≫ means that K-means is statistically better, ≪ means that hill-climbing is statistically better and ≃ means they are statistically the same.

Update Rate	Sensor Range	Target Speed				
		0.1	0.25	0.5	0.75	0.9
5	5	≫	≃	≪	≪	≪
	10	≃	≃	≪	≪	≪
	15	≃	≪	≪	≪	≪
	20	≃	≃	≪	≪	≪
	25	≫	≫	≃	≃	≃
10	5	≫	≪	≪	≪	≪
	10	≃	≪	≪	≪	≪
	15	≃	≪	≪	≪	≪
	20	≃	≪	≪	≪	≪
	25	≃	≃	≃	≃	≃
20	5	≪	≪	≪	≪	≪
	10	≪	≪	≪	≪	≪
	15	≪	≪	≪	≪	≪
	20	≪	≪	≪	≪	≪
	25	≃	≃	≃	≃	≃

Table 2: Hill-climbing versus hill-climbing and K-means in combination, subset size of 12. ≫ means that hill-climbing is statistically better, ≪ means that the combination is statistically better, and ≃ means they are statistically the same.
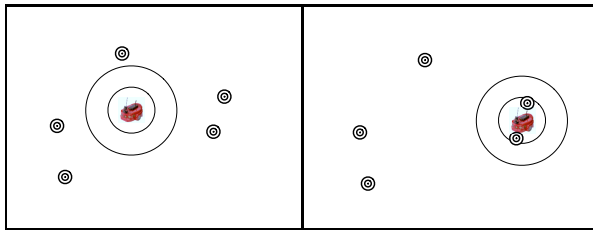

Figure 5: K-means clustering disadvantages. K-means clustering (left) centers in the region of its cluster, whereas hill-climbing (right) attempts to maximize coverage inside the sensor range.

cause targets are moving towards one another (and hence towards the mean of the cluster). By centering itself in the cluster mean, K-means clustering more often than not positions an observer to be in the path of fast incoming targets ahead of time, giving it a pronounced advantage in faster environments.

Hill-climbing outperformed K-means for very large radii: we have yet to form an explanation for this, and indeed we had expected the opposite result.

In Combination If each method did better under a certain range of parameters, how about the two in combination? We compared K-means followed by hill-climbing against K-means alone, and also against hill-climbing alone. The combination of the two does quite well as a middle-ground. In Table 2, the combination is com-

pared against hill-climbing: here it performs better than or equal to hill-climbing almost everywhere, except for very small radii or very low target speeds. Table 3 shows that it also performs well against K-means, though K-means is still superior at very high target speeds.

4.2 Discussion

Why is K-means clustering invariant over subset size? We believe the answer may lie in the fact that changes in the cluster means are made through small increments; thus it is unlikely that the targets assigned to a given cluster will be claimed by a far remote cluster. Nonetheless target disputes are bound to happen along cluster boundaries, so we would have expected *some* difference in performance.

At any rate, it is striking that two good algorithms, neither consistently better than the other, would produce such different results in terms of degradation due to decentralization. K-means clustering is proof against the argument that a centralization is by nature superior in this problem domain: but hill-climbing likewise suggests that centralization *can* offer advantages over decentralization for the same problem. To us, it is a surprising result.

Update Rate	Sensor Range	Target Speed				
		0.1	0.25	0.5	0.75	0.9
5	5	≪	≪	≃	≫	≫
	10	≪	≪	≪	≃	≫
	15	≪	≪	≪	≃	≫
	20	≪	≪	≪	≪	≃
	25	≪	≪	≪	≪	≪
10	5	≪	≪	≃	≫	≫
	10	≪	≪	≃	≃	≫
	15	≪	≪	≃	≃	≫
	20	≪	≪	≪	≃	≃
	25	≪	≪	≪	≪	≪
20	5	≪	≪	≃	≫	≫
	10	≪	≃	≃	≫	≫
	15	≪	≃	≃	≫	≫
	20	≪	≪	≃	≫	≫
	25	≪	≪	≃	≃	≃

Table 3: K-means versus hill-climbing and K-means in combination, subset size of 12. ≫ means that K-means is statistically better, ≪ means that the combination is statistically better, and ≃ means they are statistically the same.

5 Conclusion

This paper examined two algorithms for cooperative target observation, one inspired by K-means clustering and the other based on hill-climbing. Both allowed for a customization of the degree of decentralization for team control. At one extreme, each observer decided where to move next; the opposite extreme allowed for a unique central "team brain" to analyze the current situation and compute destinations for each observer. We also tested the sensitivity of the algorithms to three problem domain parameters: target speed, observation range, and algorithm update rate relative to the speed of the world.

Surprisingly, hill-climbing was sensitive to the degree of decentralization, but K-means was not. This was the case even though neither algorithm was uniformly superior to the other across all problem settings. We further examined a combination of the two algorithms which appeared to inherit advantages of both.

These algorithm-specific results suggest avenues for future work. We plan to extend this problem domain to permit soft constraints, such as communication rate, which increasingly make decentralized methods more appealing. We will examine other approaches to partial decentralization, such as control hierarchies, which might confer the advantages of both centralized and decentralized approaches. Additionally, we are interested in the inclusion of obstacles and other environmental features which increase the complexity of the problem.

References

[1] T. Balch. *Behavioral Diversity in Learning Robot Teams*. PhD thesis, College of Computing, Georgia Institute of Technology, 1998.

[2] Y. Bar-Shalom, editor. *Multitarget-multisensor Tracking*. Artech House, 1990.

[3] R. Beckers, O. E. Holland, and J.-L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Artificial Life IV: Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems , third edition*. MIT Press, 1994.

[4] A. Billard, A. Ijspeert, and A. Martinoli. A multi-robot system for adaptive exploration of a fast changing environment: Probabilistic modelling and experimental study. *Connection Science*, 11:359–379, 1999.

[5] S. S. Blackman. *Multiple-Target Tracking with Radar Applications*. Artech House, 1986.

[6] T. S. Dahl, M. J. Mataric, and G. S. Sukhatme. Adaptive spatio-temporal organization in groups of robots. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-02)*, 2002.

[7] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting: Robot-like ants and ant-like robots. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 356–363. MIT Press, 1991.

[8] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344, 2000.

[9] A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2), 2001.

[10] L. Li, A. Martinoli, and Y. S. Abu-Mostafa. Diversity and specialization in collaborative swarm systems. In T. Balch and C. Anderson, editors, *Proceedings of the 2nd International Workshop on the Mathematics and Algorithms of Social Insects*, pages 91–98, Atlanta, Georgia, USA, Dec. 15–17 2003.

[11] S. Luke, G. C. Balan, L. A. Panait, C. Cioffi-Revilla, and S. Paus. MASON: a java multi-agent simulation library. In *Proceedings of Agent 2003 Conference on Challenges in Social Simulation*, 2003.

[12] A. Martinoli and F. Mondada. Collective and cooperative group behaviours: Biologically inspired experiments in robotics. In *Proceedings of the Fourth Symposium on Experimental Robotics, ISER-95*, 1995.

[13] M. J. Mataric. *Interaction and Intelligent Behavior*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1994. Also Technical Report AITR-1495.

[14] L. Parker. Alliance: An architecture for fault tolerance multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 1998.

[15] L. Parker. Distributed algorithms for multi-robot observation of mulitple moving targets. *Autnomous Robots*, 12(3):231–255, 2002.

[16] L. Parker and B. Emmons. Cooperative multi-robot observation of mulitple moving targets. In *Proceedings of 1997 International Conference on Robotics and Automation*, pages 2082–2089, 1997.

[17] L. Parker and C. Touzet. Multi-robot learning in a cooperative observation task. In J. B. Lynne Pakers, George Bekey, editor, *Robotic Systems 4*, pages 391–401. Springer, 2000.

[18] C. Touzet. Robot awareness in cooperative mobile robot learning. *Autonmous Robots*, 2:1–13, 2000.

[19] B. B. Werger and M. Mataric. Braodcast of local eligibility for multi-target observation. In *Proceedings, 5th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 347–356, 2000.