

Adaptive Multi-Robot Behavior via Learning Momentum

J. Brian Lee (blee@cc.gatech.edu)
Ronald C. Arkin (arkin@cc.gatech.edu)

Mobile Robot Laboratory
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Email: blee@cc.gatech.edu, arkin@cc.gatech.edu

Abstract

In this paper, the effects of adaptive robotic behavior via Learning Momentum in the context of a robotic team are studied. Learning momentum is a variation on parametric adjustment methods that has previously been successfully applied to enhance individual robot performance. In particular, we now assess, via simulation, the potential advantages of a team of robots using this capability to alter behavioral parameters when compared to a similar team of robots with static parameters.

1. Introduction

Learning Momentum (LM) has been previously applied in the context of single robot navigation through obstacle fields [3,7] and was shown to improve performance in certain useful situations by altering, at run time, parameters of a behavior-based controller. We see no reason, however, to confine LM to a single robot interacting with the environment. While previous LM applications were based only on a single robot's goals, such individualism is not always appropriate for team behavior. In team situations, it may be beneficial for an individual robot to adapt to what its other teammates are doing in order to increase the performance of the overall group instead of a single robot acting greedily.

This research is being conducted as part of a larger robot learning effort funded under DARPA's Mobile Autonomous Robotic Software (MARS) program. In our overall effort, five different variations of machine learning, including learning momentum, case-based reasoning [5,9], learning in the presence of uncertainty [4], and reinforcement learning [11], have been integrated into a well-established software architecture, *MissionLab* [10].

In this paper, we study the effects of LM on a team of robots working towards a common goal. A scenario was invented in which "soldier" robots are deployed to protect a target object from incoming enemy robots. This is not unlike the classic prey-predator problem so often studied in multiagent systems [6]. To determine whether or not adaptation through LM can be beneficial, performance comparisons were made between different sized teams of

adaptive and non-adaptive soldier robots running in a simulated environment.

2. Learning Momentum Overview

Learning Momentum is a technique initially developed by Arkin, Clark, and Ram [3] and further explored later by Lee and Arkin [7] to facilitate the adaptation of robots using behavior-based controllers. It functions by allowing for situation-based incremental run-time changes to the underlying behavioral controller parameters. In essence, the central concept of LM is that if you're performing well, you should keep doing what you're currently doing, and try it a bit more. If you are not doing well, you should try something a little different. Specific parametric adjustment rules and situational identification characteristics guide the adjustment policies during learning. Thus it is a continuous, on-line, real-time adaptive learning mechanism whereby a robotic agent responds to changes in its environment.

To implement LM, the controller remembers a small set of sensor readings from its immediate history (time window). These readings are then used to identify which one of several possible pre-defined situations a robot is currently in. In the past [7], situations such as "making progress" or "impeded by obstacles" have been used to identify certain navigational cases, resulting in improved performance for a single robot moving through an obstacle-strewn world. The particular set of situations required, however, tends to be application dependent.

The robot maintains a two-dimensional table, where one dimension's size is equal to the number of possible situations the robot might encounter, while the other is equal to the number of adjustable behavioral parameters of the controller, which depends on the behaviors selected for the underlying task. The parameter type and situation are used to index into the table to get a value, or delta, that is added to that particular parameter during execution, incrementally adjusting the value as needed. In this way, the robot may alter its behavioral response to more appropriately deal with the current situation. All parameter values are bounded to keep them within acceptable levels.

In the context of navigation, behaviors such as *move-to-goal*, *avoid-obstacles*, and *wander* were combined using weighted vector summation to produce the robot’s final output vector [1]. Each behavior’s gain (how strongly it is weighted prior to summation) was included in the alterable parameters, and in this way one or more behaviors could achieve dominance over the others if the situation warranted. For example, if a robot was making progress, it would steadily increase the weight of the *move-to-goal* gain and reduce the weight of the *wander* gain. Details of the underlying methods appear in [3,7].

In this new research, we now extend these adjustment techniques to span multiple robots operating coherently towards an overall group goal. The same underlying principle of LM is exploited: parametric adjustment based on situational assessment. New methods, however, are used to represent team performance, situations, and robot interaction.

3. The Soldier Robot Scenario

For the experiments reported in this paper, a scenario was invented that places a team of soldier robots in an open environment charged with protecting a target object from enemy intruders. In certain respects this scenario is not unlike the defensive aspects of the childhood game “capture the flag”. The job of the soldier is to protect the target object from enemy soldiers who will try to attack it. When a soldier identifies the presence of an enemy robot, it should try to intercept the enemy. Upon making contact, the enemy is considered destroyed and eliminated from the scenario.

An enemy either tries to get directly through to the target object or decoy soldier robots. If the enemy makes contact with the target, its attack is considered successful, and the enemy is removed while the target remains intact for other enemies to similarly approach it. For the purposes of this experiment, the target is invincible; this allows us to see how many enemies successfully make it to the target. The overall goal of the soldier team is to intercept all enemies with none of them safely reaching their objective.

3.1 The Soldier

In these experiments, three different types of soldiers (2 static non-learning ones that serve as benchmarks, and 1 adaptive one employing LM) were created using the AuRA architecture [2] as embodied in the *MissionLab*¹ mission specification system [10]. All three types use weighted vector summation to combine four behaviors

¹ *MissionLab* is freely available over the Internet and can be downloaded from www.cc.gatech.edu/ai/robotlab/research/MissionLab.html.

(*MoveToTarget*, *InterceptEnemies*, *AvoidSoldiers*, and *AvoidObstacles*) to accomplish their task. These behaviors are described in more detail below.

MoveToTarget – This behavior produces a vector pointing to the target object and has a magnitude of $2\sqrt{D}-1$, where D is the distance to the object. The -1 allows for some repulsion when the soldier is extremely close to the object, so it will remain some distance away.

InterceptEnemies – This behavior groups enemies based on their angular distance from each other with respect to the observing soldier and extracts the closest enemy from each group. For each of these enemies, an intercept point is calculated, and a vector to that point that has magnitude $(1 + \alpha(G - 1))$, where G is the size of the group the soldier was in, and α is a constant. For these experiments, α is set to 0.05. This equation basically creates a vector pointing to an enemy group’s intercept point with a base magnitude of 1 that increases slowly but linearly with each subsequent group member. If the intercept point is within a certain radius around the soldier, it is scaled up to allow the soldier to proceed to home in on the enemy. The vectors for each calculated group are then summed

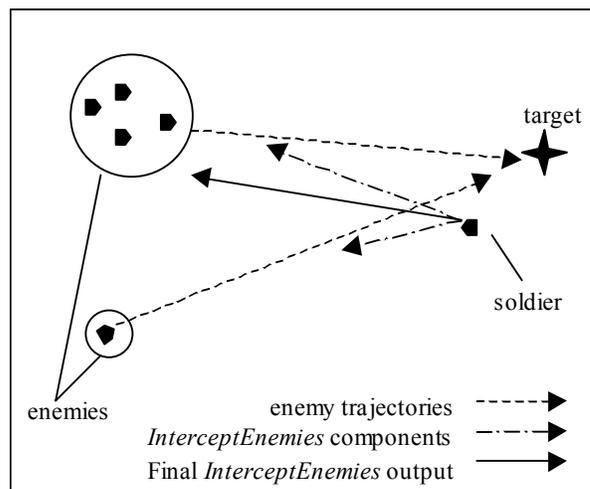


Figure 1. *InterceptEnemies* behavior

for the final output (Fig. 1).

AvoidSoldiers – This behavior takes as input a list of enemy groups and friendly soldier positions. For each friendly soldier, the behavior checks to see if it is occluded by an enemy group, where “occluded” is defined to be “within a certain angular distance of and behind”. If so, that soldier is ignored. Otherwise, the soldier is checked to see if it is engaged with any enemies, where “engaged” is defined to be “within a certain angular distance of an enemy and between the observing soldier and that enemy”. For each soldier not occluded by an enemy group, a unit vector pointing away from that soldier is calculated. For soldiers not already

engaged, this is then scaled down by a constant, which is required to keep this behavior from becoming overly dominating in the presence of a large number of robots. For these experiments, that constant was 0.4. For soldiers that *are* engaged, this scaling down is not needed since the *InterceptEnemies* behavior typically counterbalances that soldier's influence. The vectors from the unoccluded soldiers are then summed for this behavior's output. *AvoidObstacles* – This behavior produces a vector pointing away from nearby obstacles. For each obstacle

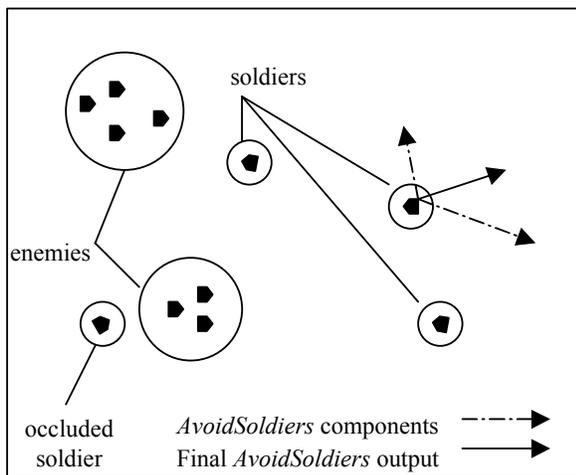


Figure 2. AvoidSoldiers Behavior

within a certain distance from the robot, a vector pointing away from the obstacle is computed, and the vectors are summed to produce the output.

In the experiments, two of the specified soldier types are non-learning and use fixed static weights for the vector summation. These are referred to as the *Static1* and *Static2* soldier types. The third soldier type uses learning momentum to dynamically change its behavioral weights and is referred to as *LM*. Table 1 gives the weighting scheme for each of the static robots. The only difference between the two robot types is the *MoveToTarget* weight. The lower weight of the *Static1* types allows them more freedom to intercept as they are

	Static1	Static2
<i>MoveToTarget</i>	0.02	0.1
<i>InterceptEnemies</i>	1.0	1.0
<i>AvoidSoldiers</i>	0.5	0.5
<i>AvoidObstacles</i>	0.1	1.0

Table 1. Behavior weights of non-learning soldiers.

less drawn to stay near the target object, but they are unstable in the absence of any enemies, in the sense that their *AvoidSoldiers* behavior may keep them moving away from each other far beyond the area they should be protecting in search of enemies. The *Static2* types are stable without enemies present, i.e., they tend to stay clustered closer to the target object, but as a consequence they don't have as long of a reach when intercepting enemies.

The *LM*-type soldiers continuously look to see which one of five possible situations it finds itself in:

- *All Clear* – There are no visible enemies present.
- *Clear to Enemy* – The soldier is between the target object and an enemy group that is not occluded by another soldier.
- *Flanking Enemy* – The soldier sees an enemy group that is not occluded by any soldier, and the soldier is not between that group and the target object.
- *Soldier Needs Help* – All enemy groups are being intercepted by soldiers, but at least one soldier group is overwhelmed by an enemy group. "Overwhelmed" is defined to mean that $S/E < T$, where S is the size of the intercepting soldier group, E is the size of the enemy group being intercepted, and T is a threshold (set to 0.5 for these experiments).
- *No Soldiers Need Help* – Enemy groups exist, but they are all being intercepted by soldier groups of appropriate size.

The behaviors' weight changes for each situation are given in Table 2. In addition, all weights are bound to

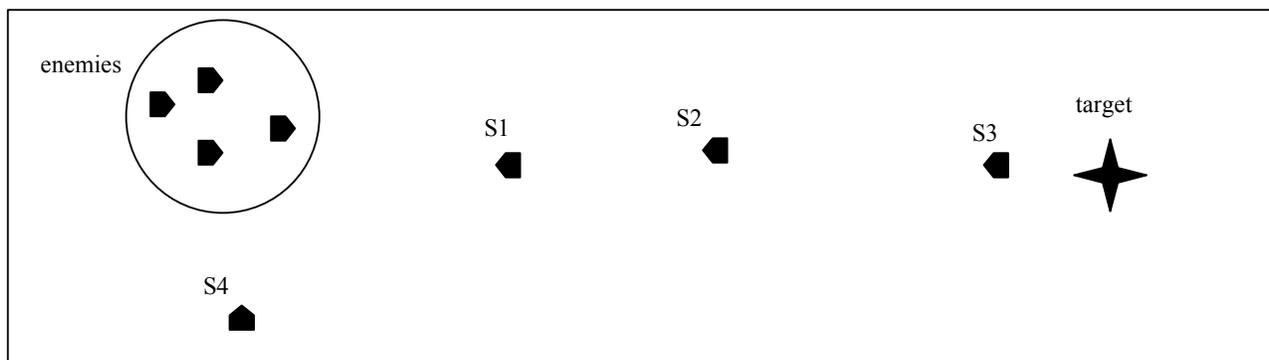


Figure 3. Different soldier situations. S1, S2, S3, and S4 are soldiers in the situations *Clear to Enemy*, *Soldier Needs Help*, *No Soldiers Need Help*, and *Flanking Enemy*, respectively.

remain between 0.01 and 1.0, except for the *AvoidSoldiers* weight, which was bound by 0.1 and 1.0.

The desired interplay of the behaviors in the *LM* teams is to allow a soldier to go where it is needed the most in light of changing circumstances. For example, if a soldier sees a group of enemies, but other soldiers are handling that group, the *No Soldiers Need Help* situation should be invoked, and the behavior weights are adjusted so that the soldier moves back to the target object instead of running out after the enemy group. This is desirable so that the target is not left without protection if any enemies from the group get through or if another enemy group

	Move To Target	Intercept	Avoid Soldiers
All Clear	0.05	-0.05	0.05
Flanking Enemy	-0.1	0.1	-0.1
Clear To Enemy	-0.1	0.1	0.1
Help Needed	-0.1	0.1	-0.1
No Help Needed	0.1	-0.1	-0.1

Table 2. Behaviors' weight changes for different situations.

appears from another direction.

3.2 The Enemy

Enemy soldiers come in two types: runners and decoys. A runner, upon creation, will immediately move in a straight line to the target object. The decoy, on the other hand, will begin a random walk when it is created. The purpose of the decoy is to draw soldiers toward them so runners have a clear path to the target object.

4. Simulation Experiments

The *MissionLab* software suite was used to conduct the experiments. The target object was placed in the middle of a 200m x 200m mission area with 3% obstacle coverage, where randomly placed circular obstacles ranged from 1 to 5 meters in radius.

A total of eighteen different soldier teams were constructed; for each of the three types of soldiers, there were teams ranging in size from one to six members. Each team was tasked with protecting the target object against five different enemy attacking strategies.

For the following descriptions, compass directions will be used. North and east refer to the positive Y and X-axes, respectively, in global coordinates.

1. The first strategy had enemies approaching from the north, south, east, and west. Enemies approaching from the south did so with a third of the frequency of the other directions.
2. The second strategy had enemies approaching from the northwest, west, and southwest.

3. The third had decoy enemies to the west while groups of five enemy runners approached from the northeast.
4. The fourth was like the third, except decoys also appeared in the south.
5. The last had enemies coming from all directions, but always in pairs from opposite sides of the target object (i.e. if an enemy appeared to the east, one would simultaneously appear to the west). Enemies appearing from a particular "spawn point" did so with a regular frequency.

For strategies 1 and 2, enemies appeared every 100 simulation time steps. For strategies 3 and 4, decoys appeared in groups of two every 200 time steps, while runners appeared every 500 time steps. For strategy 5, there were six spawn points with a frequency of 300 time steps, but they were staggered such that two enemies were created every 100 time steps.

Each of the eighteen teams was then allowed to run against each of the five enemy strategies 100 times. Each run had a duration of 20,000 simulation time steps.

4.1 Results

As the tests ran, statistics on enemy births and deaths were collected to see how many times the target was reached and how far from the target enemies were intercepted by soldiers. Since decoys were not drawn to the target, they posed no danger, and whether or not they were intercepted was of no consequence to the protection of the target. Therefore, the following data comes only from enemy runners.

Figures 5 – 9 show mean distances of interceptions of enemy runners from the target object that were using different attack strategies. Figures 10 – 14 show the overall percentage of interceptions of enemy runners that were using different strategies.

With respect to distance from the target object, *LM* and *Static1* teams vary in their ability to outperform each other, depending upon the attack strategy. *LM* is the clear winner for strategy 1, while *Static1* has the larger distances from target for strategy 3. For strategies 2, 4, and 5, *LM* performs better for teams of five or less, but *Static1* matches or outperforms *LM* with six robots. For strategies 2 – 5, the maximum average distance of interception for the *LM* teams occurred with five robots. The *Static1* teams, however, never hit a peak with their mean interception distance (except for strategy 3); whenever their team size increased, so did their mean interception distance. Both *LM* and *Static1* robots outperformed *Static2* robots.

The percentage of runners intercepted is arguably the more important of the two metrics presented here. Although there may be exceptions, if given a tradeoff between intercepting more enemies or intercepting

enemies while they are farther from their target, the former is likely more desirable. With regards to the number of enemy runners intercepted, the *LM* and *Static2* teams generally seemed to set the upper and lower performance bounds for this metric, respectively, between which the *Static1* teams' performances resided. Other than that, few cross-strategy generalizations can be drawn. At times, *LM* performed substantially better than *Static2* over almost the entire range of group sizes (strategy 3), and at times the two were more closely matched (strategy 5). Most results, however, fell in between these two extremes. For strategy 2, the teams were closely matched through team sizes of three, but on sizes above that, *LM* took a clear lead until *Static1* teams gained an advantage with six-robot teams. It is worth noting that, by the time six-robot teams were evaluated, nearly all types of robots are intercepting a high percentage of enemy runners. The only exception for this is the *Static2* team when defending against strategy 4.

These observations imply that adaptation can be beneficial for a limited number of robots, but given enough soldier team members it may not be necessary. This seems consistent with the intuition that if you have enough robots on hand to deal with the enemies they likely don't have to be adaptive, i.e., their strength lies in their numbers. On the other hand, when available soldier resources are stretched to their limits then adaptation seems to be of more value.

We must also keep in mind, however, that the presence of enemies during the tests was constant. As was previously mentioned, the reduced *MoveToTarget* weight of the *Static1* robots makes teams of this type unstable in the absence of enemies. Without an enemy presence, the *AvoidSoldiers* behavior dominates, and the team continues to disperse. Therefore, the lower *MoveToTarget* weight that allows *Static1* robots more freedom to move can also be detrimental to the group's overall performance. The *Static2* teams are stable without enemies, but are more constrained spatially when enemies are present. It is likely that adaptation have greater value when enemy strategies change rather than remain constant during an overall attack. We hope to investigate this in the near future.

5. Conclusions

Several statements can be made from the data gathered. The first is that, of the soldier types tested, there was no clear-cut winner in all situations. Enemy strategy 1, where runners simultaneously came from different directions, played well to *LM*'s strength in that it could split up soldiers to go in different directions, and so the *LM* team prevailed on both distance and percentage of interception metrics.

Since the *Static2* teams never performed significantly better than any other teams, we can conclude that *Static2* should not be used exclusively. However, we cannot use *Static1* robots exclusively, either, since they are not stable in the absence of enemies, i.e., they will move arbitrarily far away from the target object they are trying to protect while in search of enemies. One strategy may be to have soldiers switch between controllers depending on whether or not enemies are present. This would have been beneficial in instances when enemy strategy 3 is used, where *Static1* outperformed *LM* with respect to the distance metric and performed comparably to *LM* with respect to the interception percentage metric. This is very similar to our previous work in integrating case-based reasoning with LM [8].

Mixing the two static robot types on a single team would also probably lead to better performance when a large number of soldiers are available. When we get to six-robot teams, *Static1* robots either outperform or indicate a future out-performance (if we extrapolate the graphs) of *LM* robots.

The *LM* robots appear to be of greatest benefit with a smaller number of soldiers. This conclusion should not be surprising. If a large number of soldiers are available, then they can simply disperse into a cloud of soldiers around the target such that any approaching enemy has to "run the gauntlet" to get to the target. If a limited number of soldiers are available, however, the density of soldiers around the target is reduced, so all soldiers must be put to the best use possible, be it moving forward to attack an enemy or staying with the target object.

Future work in this domain includes verification of these results on physical robots. Other possibilities include exploring more enemy strategies and looking into other adaptive strategies that could improve results when the number of robots in the team is increased. Interesting results may also come from using a case-based reasoner for the parameters either in conjunction with or in place of learning momentum. Switching enemy strategies in the middle of a run could also be insightful.



Figure 4. A Pioneer 2-DXe to be used in physical robot experiments.

Acknowledgments

This research is supported under DARPA's Mobile Autonomous Robotic Software Program under contract #DASG60-99-C-0081. The authors would also like to thank Dr. Douglas MacKenzie, Yoichiro Endo, Alex Stoytchev, William Halliburton, and Dr. Tom Collins for their role in the development of the *MissionLab* software system. In addition, the authors would also like to thank Amin Atrash, Jonathan Diaz, Yoichiro Endo, Michael Kaess, Eric Martinson, and Alex Stoytchev.

References

- [1] Arkin, R.C., "Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation", *Robotics and Autonomous Systems*, 6 (1990), pp. 105-122.
- [2] Arkin, R.C., Balch, T.R. "AuRA: Principles and Practice In Review," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9(2-3), 1997, pp. 92-112.
- [3] Arkin, R.C., Clark, R.J., and Ram, A., "Learning Momentum: On-line Performance Enhancement for Reactive Systems", *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, May 1992, pp. 111-116.
- [4] Atrash, A. and Koenig, S. "Probabilistic Planning for Behavior-Based Robots", *Proceedings of the International FLAIRS conference (FLAIRS)*, pp. 531-535, 2001.
- [5] Endo, Y., MacKenzie, D.C., and Arkin, R.C. "Usability Evaluation of High-Level User Assistance for Robot Mission Specification", Georgia Tech Technical Report GIT-GOGSCI-2002/06, College of Computing, Georgia Institute of Technology, 2002.
- [6] Haynes, T. and Sen, S. "Evolving Behavioral Strategies in Predators and Prey", *Workshop on Adaptation and Learning in Multiagent Systems*, Montreal, Canada, 1995, pp. 32-37.
- [7] Lee, J. B., Arkin, R. C., "Learning Momentum: Integration and Experimentation", *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, May 2001, pp. 1975-1980.
- [8] Lee, J. B., Likhachev, M., and Arkin, R.C., "Selection of Behavioral Parameters: Integration of Discontinuous Switching via Case-Based Reasoning with Continuous Adaptation via Learning Momentum," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, May 2002, pp. 1275 – 1281.
- [9] Likhachev, M., Arkin, R.C., "Spatio-Temporal Case-Based Reasoning for Behavioral Selection," *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, May 2001, pp. 1627-1634.
- [10] MacKenzie, D., Arkin, R.C., and Cameron, R., "Multiagent Mission Specification and Execution", *Autonomous Robots*, Vol. 4, No. 1, Jan 1997, pp. 29-52.
- [11] Martinson, E., Stoychev, A., and Arkin, R., "Robot Behavioral Selection Using Q-Learning", *Proceedings of the 2002 IEEE International Conference on Intelligent Robots and Systems*, 2002.

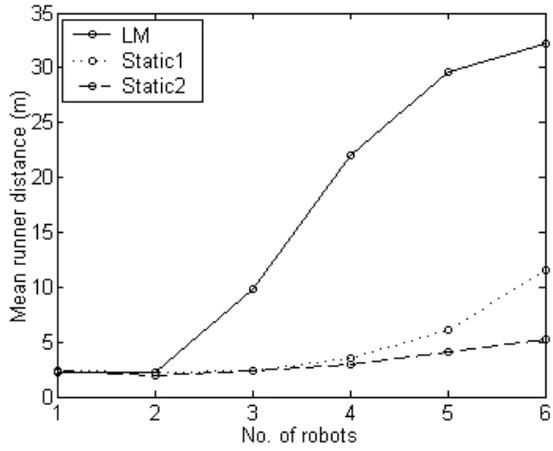


Figure 5 – Strategy 1

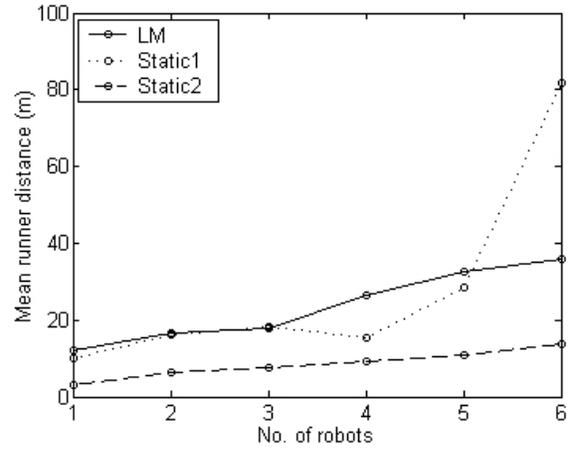


Figure 6 – Strategy 2

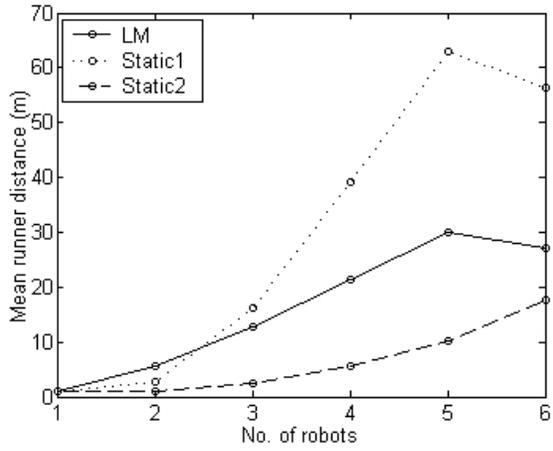


Figure 7 – Strategy 3

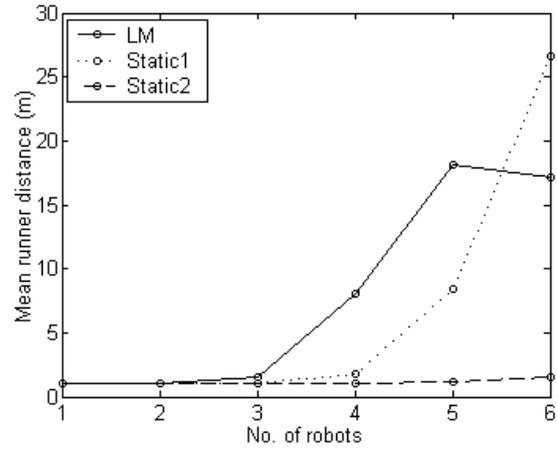


Figure 8 – Strategy 4

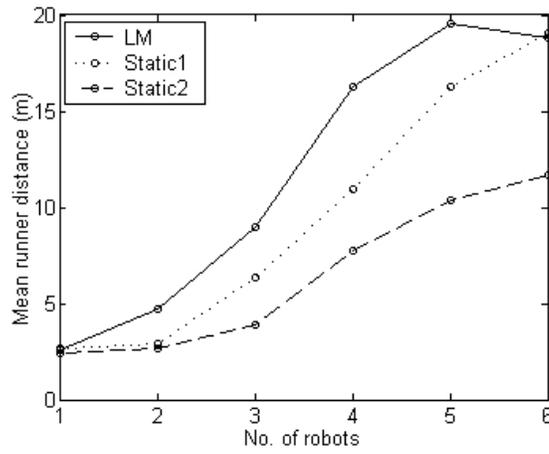


Figure 9 – Strategy 5

Figures 4-8 show mean distances from the target object of interceptions of enemy runners that were using different strategies.

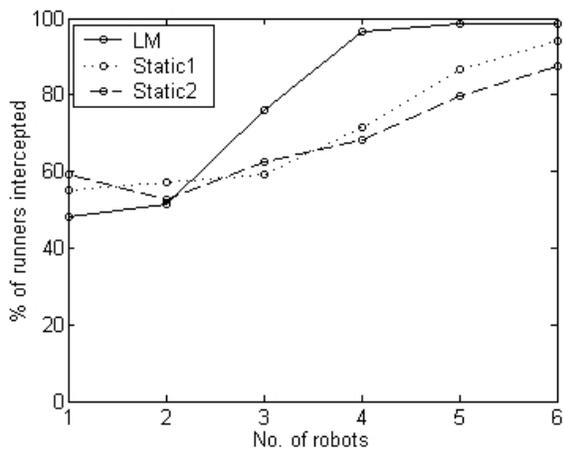


Figure 10 – Strategy 1

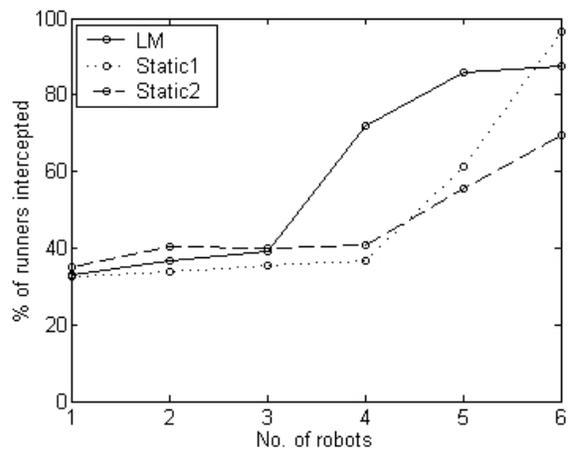


Figure 11 – Strategy 2

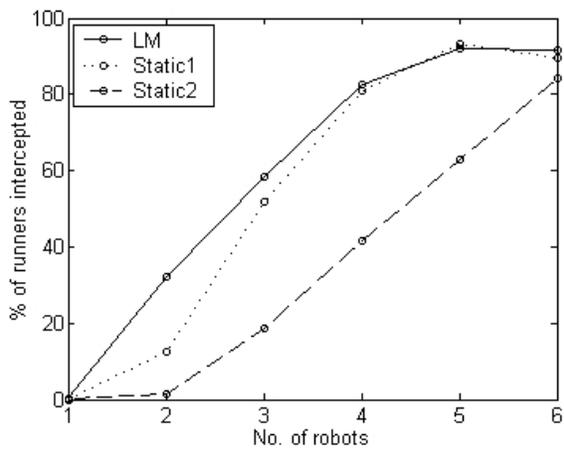


Figure 12 – Strategy 3

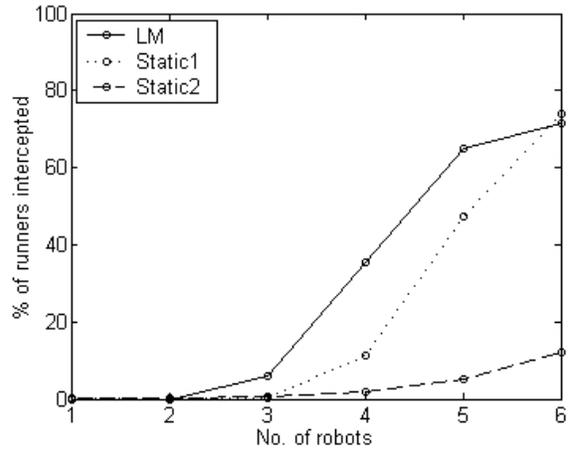


Figure 13 – Strategy 4

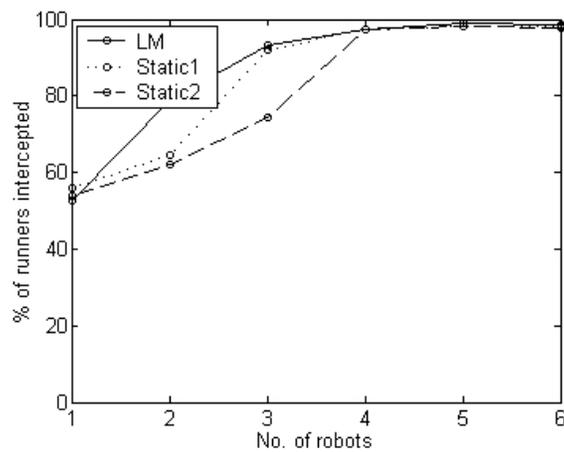


Figure 14 – Strategy 5

Figures 9-13 show the percentage of interceptions of enemy runners that were using different strategies.