Empirical Study on the Effects of Synthetic Social Structures on Teams of Autonomous Vehicles

Adam Campbell, Annie S. Wu, Keith Garfield, Randall Shumaker, Sean Luke, and Kenneth A. De Jong

Abstract— **The goal of this research is to explore the effects of social interactions between individual autonomous vehicles (AVs) in various problem scenarios. We will take a look at one way to construct the social relationships and generate data from computer simulations to compare the behaviors of each. A difference can be noticed when Synthetic Social Structures (SSS) are used to control the interactions between neighboring AVs. Our experiments show that SSSs can be used to improve team performance on a problem in which a team of AVs must maneuver through a narrow corridor to reach a goal.**

I. Introduction

Various algorithms have been constructed to manage the behavior of groups of autonomous agents. Some methods use global data as the main source of information when planning a robot's course (Bayazit, Lien, and Amato 2004), while others try to develop complex systems with emergent behavior using simple local interactions between neighboring robots (McLurkin and Smith 2004; Payton, Estkowski, and Howard 2004; Spears, Spears, Heil, Kerr, and Hettiarachchi 2004). Biologically inspired work, such as artificial insect swarms and ant colony optimization, has shown that global information is not required in order to obtain complex behavior in robot swarms (Payton, Estkowski, and Howard 2004). A few of the goals of these previous experiments has been to coordinate groups of agents to solve problems such as surveillance (Seyfried, Szymanski, Bender, Estaña, Thiel, and Wörn 2004; Spears, Spears, Heil, Kerr, and Hettiarachchi 2004), navigating difficult terrain (Bayazit, Lien, and Amato 2004), and even herding sheep (Blumenthal and Parker 2004).

The goal of our work is to study the effects of Synthetic Social Structures (SSS) on the ability of a team of distributed agents cooperating to accomplish a mutual goal. Our test problem can best be described by the following real world examples. Imagine leaving an opera house after the show's completion, and as you exit the building, you notice another person attempting to leave through the same exit. This situation occurs frequently in our lives, and we normally solve it by being polite and letting the other person go first. Sometimes, we even change the speed of our walk to reach the door at a different time than the other person. But, what if you were a simple, autonomous robot with no concept of politeness, and because you and the neighboring robot have both been "trained" to avoid collisions with oncoming robots, you each decide to stop before exiting the building? You and the other robot would remain there indefinitely, causing a blockage at the exit. This problem would never be resolved, resulting in the robots behind you to never leave the building either. Another example of this type of problem would be an encounter at a four-way stop sign, when two cars approach the intersection simultaneously. Depending on your local customs or laws, the drivers of the two cars should know that the person on the left (or the right) has the right of way. This would be simple enough to program into the robots' control systems, but what happens when four cars approach the intersection simultaneously? Obviously, the person on the right (or left) method would not be applicable to this situation. Humans are able to resolve such deadlocks by following accepted social rules of engagement, such as a polite driver motioning for the others to go first. We believe that such rules of interactions can also benefit teams of interacting autonomous agents. In this paper, we investigate the impact of introducing a social dominance hierarchy into a team of cooperating agents.

Adam Campbell and Annie S. Wu are with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816-2362, USA (email: {acampbel,aswu}@cs.ucf.edu).

Keith Garfield and Randall Shumaker are with the Institute for Simulation and Training, University of Central Florida, Orlando, FL 32816, USA (email: {kgarfiel,shumaker}@ist.ucf.edu).

Sean Luke and Kenneth A. De Jong are with the Department of Computer Science, George Mason University, Fairfax, VA 22030, USA (email: {sean,kdejong}@cs.gmu.edu).

II. Implementation Details

The simulated agents that we will be studying are autonomous vehicles (AVs) deployed in a flat-world environment. For ease of collision detection, the AVs are modeled as discs. Individual agents have limited sensor range and can only obtain the information about their immediate environment. This piece of information is what the agent's control system uses to steer its movement. Decisions are made by all of the AVs simultaneously and the individual decisions made will control the AVs movement until the next decision is made. The time between consecutive decisions remains constant throughout a particular run.

The basic components of the agents in our simulation are their control system, sensors, and decision making system. In the remainder of this section, each of these items will be discussed in detail.

A. Control System

Creating a control system for a group of autonomous agents that will perform well in various problem scenarios is no simple task, and because our goal is to observe the effects of social interactions on group behavior, we decided to create a control system that is simple to implement, yet robust enough to manage the AVs in simple problem

scenarios. Avoiding obstacles, other agents and reaching a specified goal-point comprise the set of goals that an AV attempts to satisfy when it comes time for it to make a decision. Using its sensors, an AV picks up data from its surrounding environment and sends the data to its decision making "brain", which in turn instructs the AV where to go. These three basic steps are repeated until the simulation has ended.

B. Sensors

Using the information about its local environment and the location of the goal point, AVs make decisions that control their movement. Movement of the AVs is contained in two variables; *currentBearing* and a boolean flag *isMoving*. Notice that in these experiments, there is no *speed* variable, because the magnitude of each moving AV's velocity is identical. The number of possible values for *currentBearing* is a finite value equal to the number of sensors it has. The reason for this restriction is explained later.

Sensors can detect other AVs and environment obstacles, the number of sensors that all AVs have is constant during each run of the simulation, and each AV has the same number of sensors. Each sensor is responsible for detecting objects in only a small portion of the total viewing area of the AV. The output of a sensor consists of a simple boolean value indicating whether or not it detected an object in this small viewing area. Note that individual sensors are not able to determine whether or not there are multiple objects in its viewing area, but they are able to differentiate between the detection of terrain obstacles and AVs. The amount of area that each sensor is responsible for is dependent upon the viewing distance of the sensor and the number of sensors available to an AV. Thus, if an AV had eight sensors, each would be responsible for a separate, mutually exclusive, $45°$ "slice" of the area around the AV. As stated before, sensors can only see a finite distance, which will be referred to as *sensorDistance*. It is obvious to see that at any moment in time, an AV can detect an area of size

$$\pi * sensorDist^2 \tag{1}$$

Figure 1 shows the viewing area of the AV, the segments that can be seen by the individual sensors, and the numbering of the sensors, which will be discussed in the next section. The small portion of the environment observed by the sensors will be a large part of the total information used by AVs during the decision making process.

C. Decision Making

Decisions are made by the AVs during each simulation time step. The AV can choose to become (or remain) stationary, turn in the direction of one of its sensors, or remain heading in its current direction. As stated before, the number of possible directions that an AV can move is equal to the number of sensors that an AV has.

The decision making process begins by creating an array, *possibleMoves*, of size *numSensors*. Each item in this array can contain one of three values: *badMove*, *neutralMove*, or *goodMove*. There exists a one-to-one correspondence between the array locations, the directions that an AV can head, and the number of sensors an AV has (see Figure 1). The value at location n in *possibleMoves* indicates whether or not the AV should move towards the sensor at location n; $0 \leq n < numSensors$. For simplicity, we have denoted sensor zero as being responsible for the rightmost viewing area of an AV, with the other sensors placed in counter-clockwise order around the AV. Thus, in the eight sensor example, sensor one would be located $45°$ counter-clockwise from sensor zero, sensor two would be $90°$ counter-clockwise from sensor zero, and so on. Note that even after the AV has rotated, we still consider sensor zero to be on the right most side of the vehicle, as this makes for easiest implementation. Next, the procedure used to update the *possibleMoves* array will be discussed.

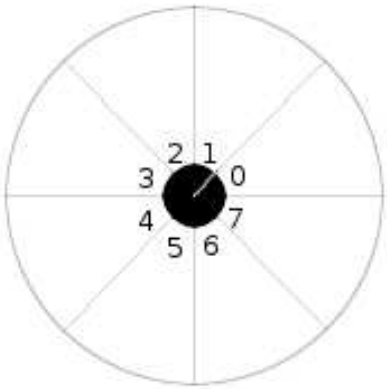

Fig. 1. In this image, the agent is depicted as a black circle, with a viewing area equal to that enclosed by the larger, grey circle. The viewing area, and numbering, of the eight sensors is also shown.

The values in *possibleMoves* are updated based on the sensor data that is gathered, and if the AV has not yet reached the goal point, the direction to the goal point is also used. Reaching the goal point is the main objective of the AVs, so the primary function of the control system is to guide the AV towards this point. The control system achieves this by first determining which direction the AV must head to reach the goal. Using this information, the control system then figures out which locations in *possibleMoves* should be updated to *goodMove*. To illustrate this, imagine the AV being at location (0, 0) on an x-y grid and the goal point being located at (300, 0). The AV must head along a straight line in the positive x-direction. (In the simulation, one would observe the AV heading towards the right of the screen.) Earlier, it was defined that the value at *possibleMoves[0]* corresponded to the right-most direction, therefore the value at *possibleMoves[0]* would now equal *goodMove*. To make

the control system more robust, the two values to the left and the two values to the right of *possibleMoves[0]* would also contain the *goodMove* value. Because sensor zero is adjacent to sensor *numSensors-1*, *possibleMoves[0]* is considered adjacent to *possibleMoves[numSensors-1]*. The AV now has the information of where it wants to go, but to avoid collisions, the AV must also know where not to go. When the control system finds that sensor *n* has detected an obstacle or oncoming AV, it places *badMove* values in the array locations *possibleMoves[n - numSensors/4]* through *possibleMoves[n + numSensors/4]*, looping from *numSensors - 1* to *0* if necessary and overwriting the *goodMove* values if they exist. This removes half of the possible moves the AV can take, to ensure that the AV will not be any where closer to the obstacle that it has detected. After *possibleMoves* has been updated using the data from the sensors, the control system tries to find all of the *goodMove* values in the array. If any are found, the AV will change its direction towards the *goodMove* direction that minimizes its angle of turn. If no *goodMove* values are found in *possibleMoves*, the control system will try to turn towards a *neutralMove* direction that minimizes its angle of turn. Finally, if no *neutralMove*s are found, the AV simply halts and will have to wait for the neighboring AVs to leave its viewing area before it can continue moving again. If two AVs happen to collide during the simulation, the *currentBearing* values of each AV are simply reversed. In our simulations, we set *numSensors* to 72 as this seemed to provide a sufficient number of possible values for *currentBearing*. The mechanisms described here define the movement algorithm of the AVs.

III. Synthetic Social Structures

To learn about how Simulated Social Structures affect the overall behavior of a group of individuals, we will be studying the behavioral differences of the AVs when a social dominance hierarchy is introduced into the system. For this social structure to work, communication is required between the agents. The communication is simple, as the AVs are only required to broadcast their dominance level to neighboring AVs. Only after an AV has discovered the dominance level of its neighbors can it decide whether or not to ignore them.

An AV's level of dominance is based on the group it belongs to. Groups are numbered sequentially from 0 to $numGroups - 1$, each AV is in exactly one group, and the number of AVs in any particular group is greater than or equal to one. An AV in group *i* ignores all other AVs in groups less than *i*, while avoiding all AVs in groups greater than or equal to *i*. The maximum number of groups that are possible with this scheme is *numAVs*, and the number of possible groupings is $2^{numAVs-1}$. Because the number of groupings increases exponentially with respect to *numAVs*, all possible groupings cannot be enumerated. To get an understanding of this SSS, we chose to study the dynamics of three completely different groupings. We hope that by studying these three groupings, one will obtain a deeper understanding of the emergent behavior created by the local social interactions.

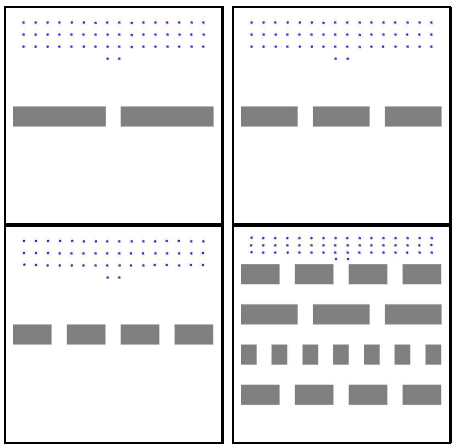

Fig. 2. The four different terrain topologies. Each hallway is just wide enough for an AV with a *sensorRadius* of 30 to fit through.

IV. Experimental Method

Our experiments are set in a flat-world, square environment with side lengths of 1,000 simulation units. For each run, 50 AVs are packed in to the top of the environment in evenly spaced rows and columns. Initially, the distance between each AV is equal to *2 * sensorRadius*, where *sensorRadius* is equal to 30 units. Also, the initial direction an AV is facing is random. Four different terrain topologies are used and shown in Figure 2, and the vertical passageways in each of the four terrains are just wide enough for the AVs to pass through. The goal point that the AVs are trying to reach is set in the center of the hallway's end. If a terrain has multiple hallways, we set goal points at the end of each of the hallways, and for the complex terrain, with multiple layers, the goal points are set at the end of the bottommost set of hallways. For each of the following SSS parameter configurations, 20 runs of the simulation were conducted, with each run lasting 10,000 time steps.

The following four experiments were run on each one of the terrains pictured in Figure 2, giving a total of sixteen experiments. To determine the impact of social rules on the group behavior of the system, we set up baseline experiments with no social interactions between AVs. These experiments were compared to experiments in which three different types of dominance structures exist in the team of AVs. Throughout the rest of the paper, we will refer to the individual experiments by their numbers below, along with a reference to the particular terrain used.

1) No SSS
2) SSS experiment with one dominant AV
3) SSS experiment where each AV is in a different group
4) SSS experiment consisting of five dominance groups, with each group containing ten AVs

V. Experimental Results

Initial experiments found that the double, triple and complex hallway tests were too easy for the AVs, even without social structures. The ease of these three terrains can be attributed to the fact that blockages at a hallway can be

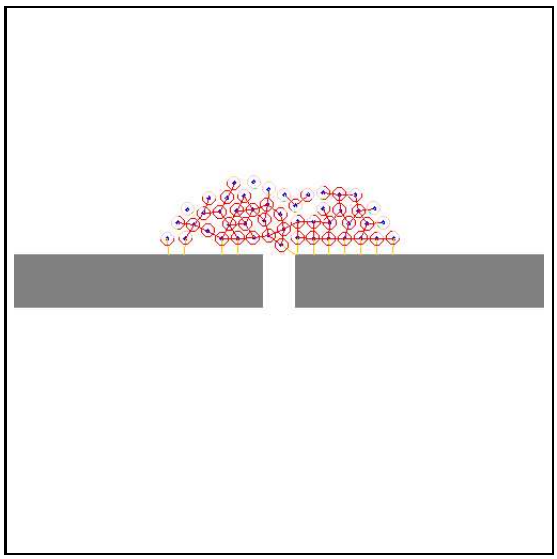

Fig. 3. This example illustrates the deadlock problem, in the single hallway experiments with no SSS. The AVs are trying to go through the hallway opening, but because they sense the corners of the hallways and neighboring AVs, they cannot move, which forces the AVs behind them to also pile up. Lines between AVs and terrain show the sensor detections, and the circle around the AVs shows where the AV can and cannot go.

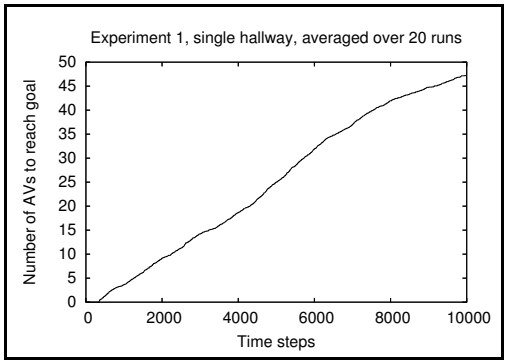

Fig. 4. This image shows the number of AVs to make it through the single hallway. Notice that some AVs never make it to the goal after the 10,000 time steps.

reduced by the AVs going through one of the other, less crowded hallways. The single hallway terrain was more difficult for the AVs to traverse because AVs in the center of a deadlock have nowhere to go (see Figure 3). As shown in Figure 4, many of the baseline experiments are unable to reach the goal of moving the entire team through the hallway in the 10,000 time steps. Because the other three terrains were simple for the AVs without social interactions, the rest of the paper will focus only on the single hall experiments.

Three different SSS experiments were conducted. In the first experiment, one AV was randomly selected from the population of agents and set to be completely dominant. The remaining 49 AVs were left unchanged. Because the dominant AV will ignore all other AVs, it will make its way through the hallway opening, regardless of whether or not AVs are in the area. We hope that this dominant AV will be able to break up deadlocks around the hall opening by

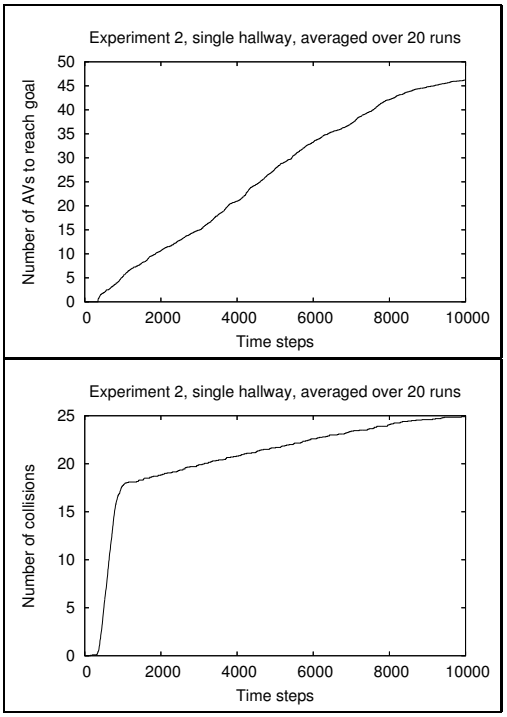

Fig. 5. Here we show the graphs for our first set of SSS experiments. We had hoped that having one dominant AV would break up deadlocks around the hallway entrance, but this did not occur. Notice that there is not a big difference between the top graph here and the graph in Figure 4.

forcing the other AVs to get out of its way.

In our first set of SSS experiments, we expected to see a slight increase in the number of collisions, along with an increase in the rate at which the AVs reach the goal. The dominant AV is always going to reach the goal quickly, as it ignores all other AVs and may even break up the deadlock formed at the hallway. When AVs hit an obstacle, whether it be the environment boundary, another AV, or the terrain, it simply reverses its direction. Therefore, the dominant AV will pass through the hallway, reach the environment boundary and begin heading back up through the hall again. As the dominant AV passes through the hall on its way up, it has the chance to force the deadlocked AVs to move slightly, thus helping to break up the deadlock. Unfortunately, this is not what happened. The dominant AV does pass back through the hallway, but because the deadlocked AVs have other AVs surrounding them, their *possibleMoves* array is full of *badMove* values and thus, they have nowhere to go. The dominant AV eventually collides with one of the stationary, deadlocked AVs and then heads back down the hallway again, not freeing up any of the less dominant AVs. An increase in collisions is observed, but an increase in the number of AVs to reach the goal is not noticed (see Figure 5). Additional tests were conducted to determine whether or not having two dominant AVs would increase the rate at which the AVs passed through the halls. These tests yielded similar results to the one-dominant tests.

Making one AV completely dominant was not a large enough change to affect the overall behavior of the system

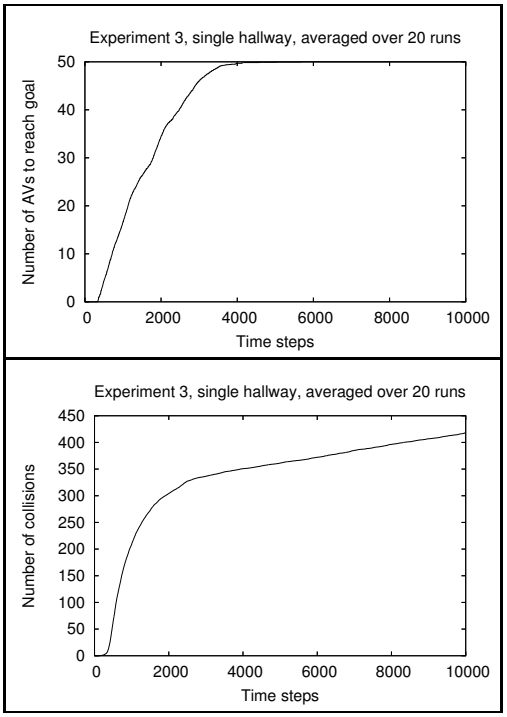

Fig. 6. Here we see the graphs for Experiment 3. The AVs make it through the hallway very fast, but with a huge number of collisions when compared to the bottom graph in Figure 5.

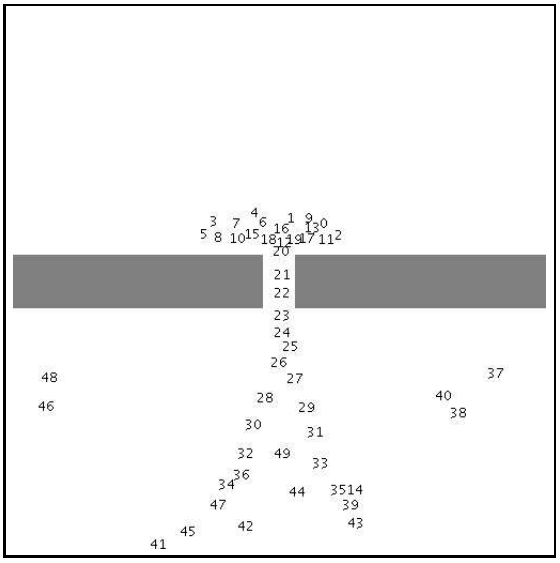

Fig. 7. This picture shows the order in which AVs with all different dominance ranks pass through the hallway. To illustrate this scenario better, we simply represent the AVs by their grouping. Less dominant AVs (lower numbers) have yet to pass through the hallway, whereas the more dominant AVs have already gone through and reached the goal.

significantly, so in our second set of experiments, we studied the behavioral changes when every single AV is in a different dominance group. We predict that every AV should get through the hallway fast but with many collisions. In these experiments, the number of groups is set to be equal to the number of AVs. At the beginning of each of the 20 runs, the AVs are randomly placed into a different dominance group. AVs in the more dominant groups should push through and reach the goal first, while the less dominant AVs will have to wait until almost no AVs are near the hallway opening in order to get through. Figure 6 shows the increase in the rate at which the AVs pass through the hallway, but as predicted, this does not come free, because the number of collisions increases drastically. What is interesting, and unexpected about these experiments is how the AVs are able to align themselves based on their groupings. This example of emergent behavior can be seen in Figure 7. No central coordinator is present to force the AVs to create such a well structured alignment, yet due to the local social interactions at the hallway opening, this behavior emerges. Because of the high number of collisions, this approach would not be practical if the AVs are fragile, as they would get damaged, but if the AVs are durable, an approach similar to this one could be used. This type of SSS might also be useful when one wishes to get the AVs through certain openings in a particular order.

So far, two vastly different SSSs have been studied. The results of the first experiment did not differ significantly, if at all, with the non-SSS experiments, whereas the second experiment took full advantage of the dominance hierarchical system and created a completely heterogeneous group of AVs. Is there a balance between these two extremes? We believe so, and we test this by breaking the AVs into five groups of ten. One group will be more dominant than the other four, the second group will be more dominant than three of the groups, and so on. We expect that, on average, the most dominant group will reach the goal first, with the least dominant group having to wait until all the others pass through the hall before proceeding. Figure 8 shows that the AVs do pass through the doorway almost as fast as the AVs with all different dominance levels, but the average number of collisions is not as low as we had hoped. This configuration is still too chaotic, and shows that dividing the AVs into optimal groupings is no simple task and may require some sort of learning algorithm.

VI. Conclusion

In this paper, we investigate the impact of introducing a Synthetic Social Structure into a multi-agent system. With the addition of the SSS, we hope to witness emergent behavior that is not present when no SSS is included. Our Synthetic Social Structure is based on a linear dominance hierarchy. AVs are given a dominance level, and it is this dominance level that determines which AVs avoid or ignore which other AVs. We test three different types of dominance structures in this work.

In our first experiment, we make one random AV extremely dominant while the other 49 have equal lower dominance. This dominance structure increases the collisions slightly over not having a social structure at all; however, the average number of AVs through the goal does not change. In our second experiment, a more drastic approach is taken,

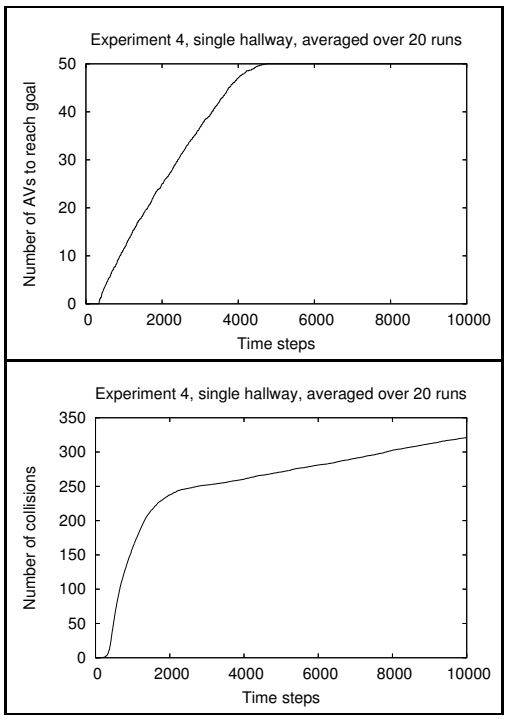

Fig. 8. When compared to Figure 6, we see that both the rate at which the AVs pass through the hallway and the number of collisions are decreased.

where each AV is given a different dominance level. This approach creates a choatic scene where an extremely large amount of collisions take place in the first 2,000 time steps. After the 2,000 time steps, most of the AVs have passed through the hallway, and the ones who have made it through arrange themselves in what appears to be a priority queue (see Figure 7). As a third experiment, we want a structure that is not as simple as the first but not as chaotic as the second, so we divide the 50 AVs into five groups of ten. This strategy reduces collisions as compared to the second experiment, and increases the average number of AVs through the hallway as compared to the first experiment.

An extremely large number of possible experiments exist using this Synthetic Social Structure, and in no way did we cover them all. Our future work will examine the application of learning algorithms to learning the parameters that define these social structures. The learning algorithm could, for example, determine the optimal groupings for a particular problem. We believe that no one set of parameters will perform optimally in every scenario, and that the performance of the system will be based on the problem at hand. If the AVs are fragile or expensive, then it would not be beneficial to use settings that cause many collisions, but if the AVs are durable, and the focus is on getting them through the hallway as fast as possible, an appropriate set of parameters should be used.

REFERENCES

Bayazit, O. B., J.-M. Lien, and N. M. Amato (2004). Swarming behavior using probabilistic roadmap techniques. In *Proc. Simulation of Adaptive Behavior (SAB) 2004 Int'l Workshop on Swarm Robotics*, Volume 3342 of *Lecture Notes in Computer Science (LNCS)*, pp. 112–125.

Blumenthal, H. J. and G. B. Parker (2004). Co-evolving team capture strategies for dissimilar robots. In *American Association for Artificial Intelligence Fall Symposium Series 2004*.

McLurkin, J. and J. Smith (2004). Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *7th Int'l Symposium on Distributed Autonomous Robotic Systems*.

Payton, D., R. Estkowski, and M. Howard (2004). Pheromone robotics and the logic of virtual pheromones. In *Proc. SAB 2004 Int'l Workshop Swarm Robotics*, Volume 3342 of *LNCS*, pp. 45–57.

Seyfried, J., M. Szymanski, N. Bender, R. Estaña, M. Thiel, and H. Wörn (2004). The i-swarm project: Intelligent small world autonomous robots for micromanipulation. In *Proc. SAB 2004 Int'l Workshop on Swarm Robotics*, Volume 3342 of *LNCS*, pp. 70–83.

Spears, W. M., D. F. Spears, R. Heil, W. Kerr, and S. Hettiarachchi (2004). An overview of physicomimetics. In *Proc. SAB 2004 Int'l Workshop on Swarm Robotics*, Volume 3342 of *LNCS*, pp. 84–97.