In *Proceedings of the Ninth International Conference on Swarm Intelligence* (ANTS14),
Brussels, Belgium, September 2014.

Influencing a Flock via Ad Hoc Teamwork

Katie Genter and Peter Stone

The University of Texas at Austin, Austin, TX, USA,
{katie,pstone}@cs.utexas.edu

Abstract. Flocking is an emergent behavior in which each individual agent follows a simple behavior rule that leads to a group behavior that appears cohesive and coordinated. In our work, we consider how to influence a flock using a set of ad hoc agents. Ad hoc agents are added to the flock and are able to influence the flock to adopt a desired behavior by acting as part of the flock. Specifically, we first examine how the ad hoc agents can behave to quickly orient a flock towards a target heading when given knowledge of, but no direct control over, the behavior of the flock. Then we consider how the ad hoc agents can behave to herd the flock through turns quickly but with minimal agents being separated from the flock as a result of these turns. We introduce an algorithm which the ad hoc agents can use to influence the flock. We also present detailed experimental results for our algorithm, concluding that in this setting, short-term lookahead planning improves significantly upon baseline methods and can be used to herd a flock through turns quickly while maintaining the composition of the flock.

1 Introduction

Consider a flock of migrating birds that is flying directly towards a dangerous area, such as an airport or a wind farm. It will be best for both the flock and the humans in the area if the path of the migratory birds is altered slightly such that the flock will avoid the dangerous area and reach their destination only slightly later than originally expected. However, there is no way to directly control the birds' flight. Rather, we must alter the environment so as to induce the flock to alter their path as desired.

The above scenario is a motivating example for our work on influencing a flock using ad hoc teamwork. We assume that each bird in the flock dynamically adjusts its heading based on that of its immediate neighbors. We assume further that we control one or more *ad hoc* agents — perhaps in the form of robotic birds or ultralight aircraft[1] — that are perceived by the rest of the flock as one of their own. It is through these *ad hoc* agents that we alter the birds' environment so as to induce them to alter their path. We are interested in *how* best to do so.

Flocking is an emergent behavior found in different species in nature including flocks of birds, schools of fish, and swarms of insects. In each of these cases, the animals follow a simple local behavior rule that results in a group behavior that

[1] www.operationmigration.org

appears well organized and stable. Research on flocking behavior has appeared in various disciplines such as physics [15], graphics [11], biology [3], and distributed control theory [7, 8, 13] but the research has focused mainly on characterizing the emergent behavior.

In this work, we are given a team of flocking agents following a known, well-defined rule characterizing their flocking behavior, and we wish to examine how the ad hoc agents should behave. Specifically, this paper addresses two questions: *How should ad hoc agents behave so as to (1) orient the rest of the flock towards a target heading as quickly as possible and (2) herd the rest of the flock through turns quickly but without compromising the composition of the flock?*

The remainder of this paper is organized as follows. Section 2 situates our research in the literature. Section 3 introduces our problem and necessary terminology. The main contribution of this paper is the 1-step lookahead algorithm for influencing a flock to travel in a particular direction; this algorithm is presented in Section 4. We present the results of running experiments using this algorithm in the MASON simulator [10] in Section 5 and then Section 6 concludes.

2 Related Work

Reynolds introduced the original flocking model that we use in this work [11]. His work focused on creating a computer model of flocking that looked and behaved like a real flock of birds. Reynolds' model consists of three simple steering behaviors that determine how each agent maneuvers based on the behavior of the agents around it (henceforth called *neighbors*): *Separation* steers the agent such that it avoids crowding its neighbors, *Alignment* steers the agent towards the average heading of its neighbors, and *Cohesion* steers the agent towards the average position of its neighbors. Vicsek *et al.* considered just the Alignment aspect of Reynolds' model in physics work that studied the emergence of self-ordered motion in flocking [15]. Some related research has also considered how different information provided to the flocking agents affects their behavior. Turgut *et al.* considered how noise in heading measurements, the number of neighbors, and the range of communication affect the self-organization of flocking robots [14]. However, none of these lines of research considered how to influence the flock to adopt a particular behavior by introducing additional agents into the flock.

Jadbabaie *et al.* considered the impact of adding a controllable agent to a flock [8]. They used the Alignment aspect of Reynolds' model and showed that a flock with a controllable agent will always converge to the controllable agent's heading. Su *et al.* also presented work that is concerned with using a controllable agent to make the flock converge [13]. [2] used the same model as [14] and extended it to include informed agents that guide the flock by their preference for a particular direction. Our work is different from these three lines of research in that while they influence the flock to converge to a target heading eventually, we influence the flock to converge quickly.

Couzin *et al.* considered how grouping animals make informed unanimous decision [3]. They showed that only a very small proportion of informed agents is required, and that the larger the group the smaller the proportion of informed individuals needed to orient the group. Cucker and Huepe proposed two Laplacian-based models for a consensus term that balances the trade-off between an informed individuals preference to go in a particular direction and the desire for social interaction [4]. Ferrante *et al.* utilized communication for coordinating movement of a flock towards a common goal [5]. Specifically, informed robots communicated the goal direction while uniformed robots communicated the average of messages received from their neighbors. Yu *et al.* proposed an implicit leadership algorithm that allows all agents to follow a single rule and reach a common group decision without any complex coordination methods [16]. However, none of these lines of research consider how to control some agents from the perspective of knowing and planning for how the other agents will react. Instead, each agent behaves in a fixed way that is pre-decided or based on its type.

Han *et al.* studied how one agent can quickly influence the direction in which an entire flock of agents is moving [7]. In their work each agent follows a simple control rule based on its neighbors, but they only consider one ad hoc agent with unlimited, non-constant velocity. This allows their ad hoc agent to move to any position in the environment within one time step, whereas in our work we assume the agents have bounded velocity.

In our previous work, we considered the problem of leading a flock of agents to a desired orientation using ad hoc agents [6]. In that work we set bounds on the extent to which both stationary and non-stationary ad hoc agents could influence an otherwise stationary team to orient to a desired orientation. The work presented in this paper is substantially different in that we consider a completely non-stationary flock and we present a more advanced algorithm for the ad hoc agents.

Overall, to the best of our knowledge, the work presented in this paper is the first that uses knowledge of how other agents will react to design controllable agents with bounded velocities to influence a flock in motion to converge quickly to a desired heading.

3 Background and Problem Definition

In this section we introduce the concept of ad hoc teamwork and define our problem.

3.1 Ad Hoc Teamwork

Ad hoc teamwork is a relatively new multiagent systems research area [1, 9, 12] that examines how an agent ought to act when placed on a team with other agents such that there was no prior opportunity to coordinate behaviors. As agents and robots are used with increasing frequency in various cooperative domains, designing agents capable of reasoning about ad hoc teamwork is becoming increasingly important. Ad hoc agents can cooperate within a team without using explicit communication or previously coordinating behaviors among teammates. One aspect of ad hoc teamwork involves *leading* teammates. Consider a case in which we want to influence a given team of agents to alter their actions in order to maximize the team utility. One way of doing so is by adding one or more agents to the team in order to *lead* them to perform the desired actions.

3.2 Problem Definition

In this work we use a simplified version of Reynolds' Boid algorithm for flocking [11]. Specifically, similarly to other studies such as [8, 15], we only consider the Alignment aspect of Reynolds' model. We assume that each agent calculates its orientation for the next time step to be the average heading of its *neighbors*. Throughout this paper, an agent's *neighbors* are the agents located within some set radius of the agent. An agent is not considered to be a neighbor of itself, so an agent's current heading is not considered when calculating its orientation for the next time step. In order to calculate its orientation for the next time step, each agent computes the vector sum of the velocity vectors of each of its neighbors and adopts a scaled version of the resulting vector as its new orientation. Figure 1 shows an example of how an agent's new velocity vector is calculated. At each time step, each agent moves one step in the direction of its current vector and then calculates its new heading based on those of its neighbors, keeping a constant speed.

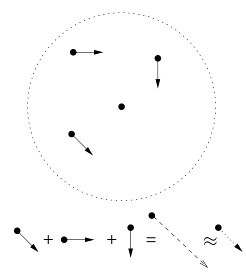


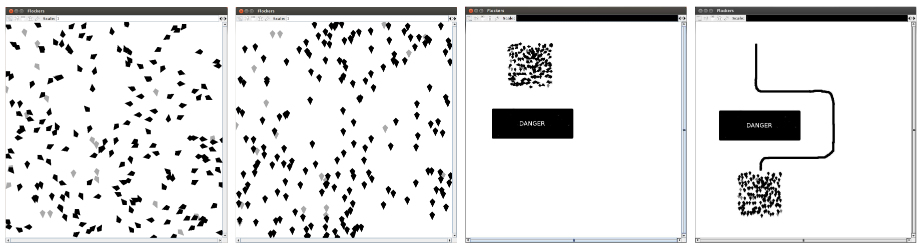
Fig. 1: An example showing how an agent's new velocity vector is calculated. The black dot without an arrow represents the agent in question, the dots with arrows represent the agent's neighbors and their velocity vectors, and the dotted circle represents the boundary of the agent's neighborhood. The agent's new velocity vector is calculated as shown at the bottom of the figure.

Over time, agents behaving as described above will naturally gather into one or more groups, and these groups will each travel in some direction. However, in this work we add a small number of *ad hoc agents* to the flock. These ad hoc agents attempt to influence the flock to travel in a pre-defined direction — we refer to this direction as θ^*. This paper addresses two questions: how to *orient* the flock to a target heading and how to *herd* a flock through turns. Hence, throughout this paper we consider two specific cases. In the **Orient** case, the ad hoc agents attempt to influence the flock to travel towards θ^*. In the **Herd** case, the ad hoc agents attempt to influence the flock to travel as a cohesive unit through multiple turns — this can be thought of as influencing the flock towards a frequently changing θ^*.

Note that the challenge of designing ad hoc agent behaviors in a dynamic flocking system is difficult because the action space is continuous. Hence, in our work we make the simplifying assumption of only considering a limited number (*numAngles*) of discrete angle choices for each ad hoc agent.

3.3 Simulation Environment

We situate our research on flocking using ad hoc teamwork within the MASON simulator [10]. Pictures of the Flockers domain are shown in Figure 2. Each agent points and moves in the direction of its current velocity vector.



(a) (b) (c) (d)

Fig. 2: Images of (a) the start of an **Orient** trial, (b) the end of an **Orient** trial, (c) the start of a **Herd** trial, and (d) the end of a **Herd** trial (the black line shows the approximate path that the flock travelled to reach their current location) in the MASON Flockers simulation environment. The grey agents are ad hoc agents while the black agents are other members of the flock.

Videos showing the simulator in action in both cases are available on our web page[2]. Our experimental setup using the MASON simulator is described in much more detail in Section 5.2.

4 1-Step Lookahead Behavior

As specified in Section 3, the variable under our control is the heading of each ad hoc agent at every time step of the simulation.

In this section we present Algorithm 1, a 1-step lookahead algorithm for determining the individual behavior of each ad hoc agent. This behavior considers *all* of the influences on neighbors of the ad hoc agent, such that the ad hoc agent can determine the best orientation to adopt based on this information. The 1-step lookahead behavior is a greedy, myopic approach for determining the best individual behavior for each ad hoc agent, where 'best' is defined as the behavior that will exert the most influence on the next time step.

Note that if we only considered the current orientations of the neighbors (instead of the influences on these neighbors) when determining the next orientation for the ad hoc agent to adopt, we would only be estimating the state of each neighbor and hence the resulting orientation adopted by the ad hoc agent would not be 'best'.

The variables used throughout Algorithm 1 are defined in Table 1. Two functions are used in Algorithm 1: *neighbor.vel* returns the velocity vector of neighbor while *neighbor.neighbors* returns a set containing the neighbors of neighbor. Note

[2] `http://ants14-flocking.blogspot.com/`

that Algorithm 1 is called on each ad hoc agent at each time step, and that the neighbors of the ad hoc agent at that time step are provided as a parameter to the algorithm. The output from the algorithm is the orientation that, if adopted by this ad hoc agent, is predicted to influence its neighbors to face closer to θ^* than any of the other *numAngles* discrete ad hoc orientations considered.

Algorithm 1 bestOrient = 1StepLookahead(neighOfAH)

```
1:  bestOrient ← (0, 0)
2:  bestDiff ← ∞
3:  for each ad hoc agent orientation vector ahOrient do
4:      nOrients ← ∅
5:      for n ∈ neighOfAH do
6:          nOrient ← (0, 0)
7:          for n' ∈ n.neighbors do
8:              if n' is an ad hoc agent then
9:                  nOrient ← nOrient + ahOrient
10:             else
11:                 nOrient ← nOrient + n'.vel
12:         nOrient ← nOrient / |n.neighbors|
13:         nOrients ← {nOrient} ∪ nOrients
14:     diff ← average difference between the vectors of nOrients and θ*
15:     if diff < bestDiff then
16:         bestDiff ← diff
17:         bestOrient ← ahOrient
18: return bestOrient
```

Conceptually, Algorithm 1 is concerned with how the neighbors of the ad hoc agent are influenced if the ad hoc agent adopts a particular orientation at this time step. Figure 3 presents a pictorial explanation of the calculation of *nOrient* (lines 6-12 in Algorithm 1). In the figure, *nOrient*, the predicted next step orientation vector of neighbor *n* of the ad hoc agent, is calculated to be the average of *n*'s neighbors (both marked *n'*) as shown below the diagram. In the example shown, *n* is the only neighbor of the ad hoc agent, so *nOrients* would only contain this one *nOrient*. However, *numAngles* ad hoc agent orientations would be considered by Algorithm 1, resulting in *numAngles* different *nOrient* vectors competing to be *bestOrient*.

Variable	**Definition**
bestDiff	the smallest difference found so far between the average orientation vectors of *neighOfAH* and θ^*
bestOrient	the vector representing the orientation adopted by the ad hoc agent to obtain *bestDiff*
neighOfAH	the neighbors of the ad hoc agent
nOrient	the predicted next step orientation vector of neighbor *n* of the ad hoc agent if the ad hoc agent adopts *ahOrient*
nOrients	a set containing the predicted next step orientation vectors of all of the neighbors of the ad hoc agent, assuming the ad hoc agent adopts *ahOrient*

Table 1: Variables used in Algorithm 1.

Now let us walk through the algorithm in more detail. Algorithm 1 considers each of the *numAngles* discrete ad hoc agent orientation vectors. For each orientation vector, we consider how each of the neighbors of the ad hoc agent will be influenced if the ad hoc agent adopts that orientation vector (lines 3-13). Hence, we consider all of the neighbors of each neighbor of the ad hoc agent (lines 7-11) — if the neighbor of the neighbor of the ad hoc agent is an ad hoc agent, we assume that it has the same orientation as the ad hoc agent (even though, in fact, each ad hoc agent orients itself based on a different set of neighbors, line

9), whereas if it is not an ad hoc agent, we calculate its orientation vector based on its current velocity (line 11). Using this information, we calculate how each neighbor of the ad hoc agent will be influenced by averaging the orientation vectors of the each neighbor's neighbors (lines 12-13). We then pick the ad hoc agent orientation vector that results in the least difference between θ^* and the neighbors' current orientation vectors (lines 14-18).

If we assume that there are *numAgents* of agents in the flock, we can calculate the worst-case complexity of Algorithm 1 as follows. Line 3 executes *numAngles* times, line 5 executes at most *numAgents* times, and line 7 executes at most *numAgents*. Hence, the complexity for Algorithm 1 is $O(numAngles * numAgents^2)$.

Results regarding how Algorithm 1 performs in both the **Orient** case and the **Herd** case can be found in Section 5.

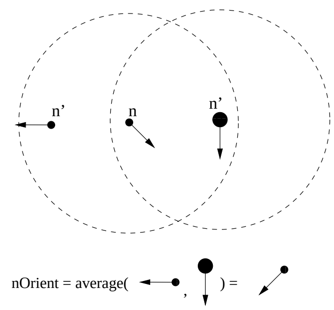


Fig. 3: Diagram illustrating how *nOrient* is calculated in Algorithm 1. Each agent is shown as a dot with an arrow pointing towards its heading. The ad hoc agent is the agent with the larger dot. The dotted circles represent the neighborhood of the agent at the center of the circle.

5 Experiments

In this section we describe our experiments testing the ad hoc agent behavior presented in Section 4 against a baseline method. We describe experiments for both the **Orient** case and the **Herd** case.

5.1 Baseline Ad Hoc Agent Behavior

In this subsection we describe the *Face Desired Orientation* heuristic behavior, which serves as our baseline for comparison. When following this behavior, the ad hoc agents always orient towards θ^*. Note that under this behavior the ad hoc agents do not consider their neighbors or anything about their environment when determining how to behave.

This behavior is modeled after work by Jadbabaie *et al.* [8]. They show that a flock with a controllable agent will eventually converge to the controllable agent's heading. The *Face Desired Orientation* ad hoc agent behavior is essentially the behavior described in their work, except that in our experiments we include multiple controllable agents facing θ^*.

5.2 Experimental Setup

We utilize the MASON simulator [10] for our experiments in this paper. The MASON simulator was introduced in Section 3.3, but in this section we present the details of the environment that are important for completely understanding and replicating our experimental setup.

The baseline experimental settings for variables are given in Table 2 for both the **Orient** case and the **Herd** case. We chose for 10% of the flock to be ad hoc agents as a trade-off between providing enough ad hoc agents to influence the flock and keeping the ad hoc agents few enough to require intelligent behavior in order to influence the flock effectively.

Variable	**Orient Default**	**Herd Default**
toroidal domain	*yes*	no
domain height	*150*	300
domain width	*150*	300
units moved by each agent per time step	*0.7*	0.2
number of agents in flock (*numAgents*)	*200*	*200*
% of flock that are ad hoc agents	10%	10%
neighborhood for each agent (diameter)	*20*	*20*

Table 2: Experimental settings for variables in the **Orient** and **Herd** cases. Italicized values were default settings for the simulator.

For the **Orient** case, the domain is toroidal. This means that agents that move off one edge of our domain reappear on the opposite edge moving in the same direction. However, for the **Herd** case we removed the toroidal nature of the domain so as to make the domain more realistic. Hence, if agents move off one edge of our domain in the **Herd** case, they will not reappear.

For the **Orient** case, agents are initially randomly placed with random initial headings throughout the domain. For the **Herd** case, agents are initially randomly placed within a square in the top left of the domain, where this square occupies 4% of the domain. Agents are assigned random headings that are within 90 degrees of the initial θ^* for the **Herd** case.

We only consider *numAngles* discrete angle choices for each ad hoc agent. In all of our experiments, *numAngles* is 50, meaning that the unit circle is equally divided into 50 segments beginning at 0 radians and each of these orientations is considered as a possible orientation for each ad hoc agent. *numAngles*=50 was chosen after some experimentation using the **Orient** case in which *numAngles*=20 resulted in a higher average number of steps for the flock to converge to θ^* and *numAngles*=100 did not require significantly fewer steps for convergence.

In our experiments, we conclude that the flock has converged to θ^* when every agent (that is not an ad hoc agent) is facing within 0.1 radians of θ^*. Other stopping criteria, such as when 90% of the agents are facing within 0.1 radians of θ^*, could have also been used. We tested this alternate stopping criteria in the **Orient** case, but found that using it did not qualitatively alter the results.

In all of our **Orient** experiments, we run 50 trials for each experimental setting. In our **Herd** experiments we run 100 trials for each experimental setting. In the **Orient** case we use the same 50 random seeds for each set of experiments for the purpose of variance reduction, where in the **Herd** case we use the same 100 random seeds. The random seeds are used to determine the exact placement and orientation of all of the agents at the start of a simulation run.

5.3 Orient Experimental Results

Figure 4 shows the number of time steps needed for the flock to converge to θ^* for the baseline algorithm and the 1-step lookahead algorithm presented in Algorithm 1 using the experimental setup described in Section 5.2 as well as a few variants on this baseline setup. In order to further investigate the dynamics of this domain, in one variant we alter the percentage of the flock that are ad hoc agents while in the other variant we alter the number of agents in the flock. Note that although multiple metrics will be used to judge performance in the **Herd** case, only time to convergence is used in this case since in a toroidal world agents can not become permanently separated from the flock unless they are also travelling towards θ^*.

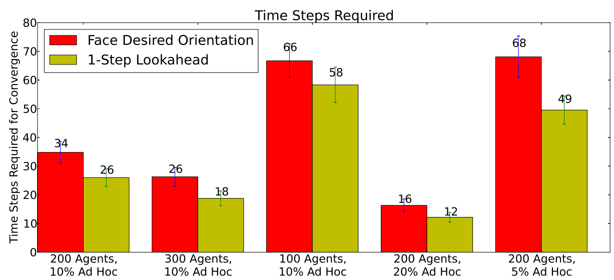


Fig. 4: Results from experiments using the experimental setup described in Section 5.2 as well as four variants on this experimental set-up. The results shown in the figure are averaged over 50 trials and the error bars represent the 95% confidence interval.

The results shown in Figure 4 clearly show that the 1-Step Lookahead Behavior performs significantly better than the baseline method in all of our experiments except when the flock size was decreased from 200 agents to 100 agents. In this experiment, although our algorithm did perform better than the baseline, we believe it did not significantly improve over the baseline because the agents were too sparse in the environment to have a strong effect on each other.

Altering the percentage of ad hoc agents in the flock clearly alters the amount of agents we can control, which affects the amount of influence we can exert over the flock. Hence, as can be seen in Figure 4, flocks with higher percentages of ad hoc agents will, on average, converge to θ^* in fewer time steps than flocks with lower percentages of ad hoc agents.

5.4 Herd Experimental Results

In our **Herd** experiments, we started all of the agents in a square occupying 4% of the domain in the upper left corner (see Figure 2.c for a picture representing a sample starting configuration). Then the ad hoc agents influenced the flock to travel downward for 300 time steps, then rightward for 300 time steps, then downward for 300 time steps, then leftward for 300 time steps, and finally downward — this path represented the path a flock might need to take to avoid an obstacle in its path.

Different numbers of time steps were used by the ad hoc agents to influence the flock to turn in these four turns. The ad hoc agents were always influencing the flock to orient towards θ^*, so during the turns the value of θ^* was interpolated linearly between the values of θ^* on the surrounding straightaways according to the number of time steps allowed for the turn. Hence, θ^* changed more rapidly when fewer time steps were allowed.

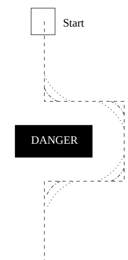


Fig. 5: Diagram showing the approximate path along which the flock is influenced to travel. The dashed line shows the path if turns were instantaneous, and two arcs are shown at each turn to show what the path looks like when 100 or 200 time steps are used to turn. The square shows the flock's starting area.

Figure 5 depicts the approximate path along which the flock is influenced to travel, including a depiction of how turns of different lengths affect this path. We maintain approximately the same time to complete all four turns by shortening the straightaway times depending on the amount of time allocated to turning. Flocks that are influenced by the ad hoc agents to turn quicker will inherently have the opportunity to finish their last turn quicker (as can be seen in Figure 5). Hence, *steps-optimal* represents the minimal number of time steps that could be spent by an agent to complete the four required straightaways and turns.

In the **Herd** experiments, we consider three metrics when determining how much controllability the ad hoc agents exerted on the flock: (1) the average total number of time steps required for the flock to converge to facing downward at the end of the path (*steps-converge*), (2) the difference between *steps-converge* and *steps-optimal* (*diff*), and (3) the average number of agents that become separated from the flock and do not return to the flock before the flock converges to facing downward at the end of the path (*lost*). We also report the number of trials in which at least one agent was separated from the flock and did not return before the flock converged to facing downward at the end of the path, as this makes *lost* easier to interpret.

Table 3 shows results of both the baseline algorithm and the 1-step lookahead algorithm using the experimental setup described above for the **Herd** case. As can be seen in the table, usage of the 1-step lookahead algorithm results in significantly better *steps-converge* and *diff* than the baseline algorithm for each of the turn times tested in the experiment. On average, flocks that are influenced to turn quicker are more likely to have a greater average *diff*. Additionally, note that given this experimental setup, the ad hoc agents would do best to use around 30 time steps to influence the flock through each turn, as *steps-converge* is least when 30 time steps are used for each turn.

Experiments were run in which the percentage of ad hoc agents in the flock was altered to 5% of the flock and 20% of the flock. Results were comparable to those presented in Table 3, but did differ in two significant ways. First, when 20% of the flock consisted of ad hoc agents, no agents were lost during our experiments. Second, when 20% of the flock consisted of ad hoc agents, turns

	Steps-Converge	Steps-Optimal	Diff	Lost	Times Lost
10 Steps to Turn - Baseline	1243.0 (4.6)	1205	38.0	17.0	1
30 Steps to Turn - Baseline	1242.3 (2.6)	1215	27.3	17.0	1
50 Steps to Turn - Baseline	1245.8 (2.2)	1225	20.8	0	0
100 Steps to Turn - Baseline	1261.0 (1.6)	1250	11.0	17.0	1
200 Steps to Turn - Baseline	1301.9 (1.0)	1300	1.9	17.0	1
10 Steps to Turn - 1-Step Lookahead	1237.0 (5.4)	1205	32.0	13.5	2
30 Steps to Turn - 1-Step Lookahead	1236.5 (4.6)	1215	21.5	17.0	1
50 Steps to Turn - 1-Step Lookahead	1238.6 (3.0)	1225	13.6	17.0	1
100 Steps to Turn - 1-Step Lookahead	1254.5 (1.3)	1250	4.5	0	0
200 Steps to Turn - 1-Step Lookahead	1300.6 (0.6)	1300	0.6	17.0	1

Table 3: Results for the baseline algorithm and the 1-step lookahead algorithm when using the experimental setup described for the **Herd** case. The numbers in parentheses show the 95% confidence interval.

lasting 10 steps had the least *steps-converge* but were still able to maintain the consistency of the flock. When only 5% of the flock consisted of ad hoc agents, more ad hoc agents were lost on quicker turns — specifically, 22 out of 100 runs lost some agents when turns lasted for 10 time steps, and on average 63.7 out of 200 agents were lost on these runs. When 5% of the flock consisted of ad hoc agents, turns lasting about 50 steps were best in terms of *steps-converge*.

Experiments were also run in which only one of the 200 agents was an ad hoc agent. Hence, one ad hoc agent was attempting to influence the entire flock through the series of four turns. In these experiments, we found that when using the 1-step lookahead algorithm, a neighborhood of 2000 in diameter was sufficient to not lose any agents on any of our 100 runs and it also obtained *steps-optimal*. We also tested a neighborhood of 200 in diameter, and in this experiment 86 of the 100 trials lost at least one agents (and in fact, all 86 lost all of the other 199 agents in the flock). Likewise, it is interesting to consider that the baseline algorithm was not able to maintain the consistency of the flock even with a 2000 diameter neighborhood, instead losing all of the other 199 agents in all 100 runs.

6 Conclusion

In this work, we set out to determine how ad hoc agents should behave in order to orient a flock towards a target heading as quickly as possible and to herd a flock around turns quickly but while still maintaining the flock. Our work is situated in a limited ad hoc teamwork domain, so although we have knowledge of the behavior of the flock, we are only able to influence them indirectly via the behavior of the ad hoc agents within the flock. This paper introduces an algorithm that the ad hoc agents can use to influence the flock — a greedy lookahead behavior. We ran extensive experiments using this algorithm in a simulated flocking domain, where we observed that in such a setting, a greedy lookahead behavior is an effective behavior for the ad hoc agents to adopt.

There are plenty of avenues for extensions to this work. We could consider other types of algorithms for the ad hoc agents, such as deeper lookahead searches or algorithms in which the ad hoc agents coordinate their behaviors. Additionally, as this work focused on a limited version of Reynolds' flocking model in which agents calculate their next heading to be the average heading of their neighbors, a promising direction for future work is to extend the algorithms presented in this work to Reynolds' complete flocking model in which agents also consider separation and cohesion when calculating the next heading.

7 Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287), ONR (21C184-01), and AFOSR (FA8750-14-1-0070).

References

1. M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, pages 53–58, 2005.
2. H. Celikkanat and E. Sahin. Steering self-organized robot flocks through externally guided individuals. *Neural Computing & Applications*, 19(6):849–865, Sep 2010.
3. I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513–516, Feb 2005.
4. F. Cucker and C. Huepe. Flocking with informed agents. *MathematicS In Action*, 1(1):1–25, 2008.
5. E. Ferrante, A. E. Turgut, N. Mathews, M. Birattari, and M. Dorigo. Flocking in stationary and non-stationary environments: A novel communication strategy for heading alignment. In *PPSN*, pages 331–340. Springer-Verlag, 2010.
6. K. Genter, N. Agmon, and P. Stone. Ad hoc teamwork for leading a flock. In *AAMAS*, May 2013.
7. J. Han, M. Li, and L. Guo. Soft control on collective behavior of a group of autonomous agents by a shill agent. *Systems Science and Complexity*, 19:54–62, 2006.
8. A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988 – 1001, Jun 2003.
9. E. Jones, B. Browning, M. B. Dias, B. Argall, M. M. Veloso, and A. T. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *ICRA*, pages 570 – 575, 2006.
10. S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. Mason: A multi-agent simulation environment. *Simulation: Transactions of the Society for Modeling and Simulation International*, 81(7):517–527, 2005.
11. C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH*, 21:25–34, Aug 1987.
12. P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, 2010.
13. H. Su, X. Wang, and Z. Lin. Flocking of multi-agents with a virtual leader. *IEEE Transactions on Automatic Control*, 54(2):293–307, Feb 2009.
14. A. Turgut, H. Celikkanat, F. Gokce, and E. Sahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2-4):97–120, 2008.
15. T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Sochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6), 1995.
16. C.-H. Yu, J. Werfel, and R. Nagpal. Collective decision-making in multi-agent systems by implicit leadership. In W. van der Hoek, G. A. Kaminka, Y. Lesprance, M. Luck, and S. Sen, editors, *AAMAS*, pages 1189–1196, 2010.