# Better Group Behaviors using Rule-Based Roadmaps

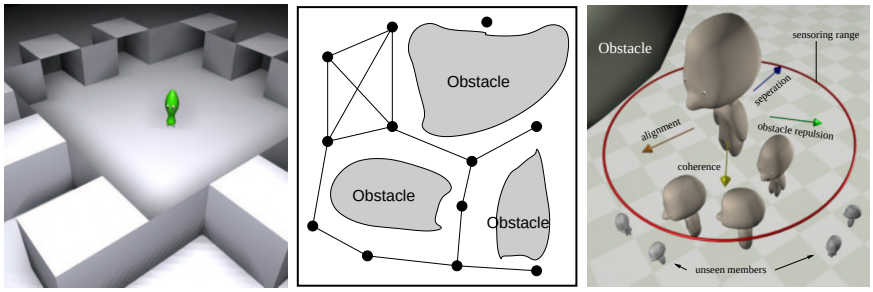
O. Burchan Bayazit, Jyh-Ming Lien, and Nancy M. Amato

Texas A&M University, College Station TX 77843, USA

**Abstract.** While techniques exist for simulating group behaviors, these methods usually only provide simplistic navigation and planning capabilities. In this work, we explore the benefits of integrating roadmap-based path planning methods with flocking techniques. We show how group behaviors such as exploring can be facilitated by using dynamic roadmaps (e.g., modifying edge weights) as an implicit means of communication between flock members. Extending ideas from cognitive modeling, we embed behavior rules in individual flock members and in the roadmap. These behavior rules enable the flock members to modify their actions based on their current location and state. We propose new techniques for three distinct group behaviors: homing, exploring (covering and goal searching) and passing through narrow areas. Animations of these behaviors can be viewed at `http://parasol.tamu.edu/dsmft`.

## 1 Introduction

Coordinating the movement of a group of robots plays an important role in robotics. Although techniques to achieve coordinated movements have attracted the attention of many researchers, most research has focused on techniques for modeling individual behavior of flock members inspired by Reynolds' *boids* [13]. Boids exhibit so-called *emergent behavior* in which characters only react to immediate events. Although initially developed for computer animation, this strategy has been used to control flocks of robots. Usually, the flock shows homogeneous behavior. Yu and Terzopoulos extended Reynolds' flocking system to create more realistic, self-animating characters. They use biomechanical modeling with a behavioral finite state machine, the intention generator [18], and propose the idea of *cognitive modeling* [5], which controls how characters gather knowledge and how they act. They develop the *cognitive modeling language, CML*, which eases an animator's task of specifying character behaviors and also enables heterogeneous behaviors to be included in flock members. Nevertheless, the ability to generate animations in complex environments is poor even if more elaborate behaviors are encoded in the CML. For example, in Figure 1(a), an alien (center) is trying to escape from a predator hiding in one of the rooms. Without memory, she cannot remember which room contains the predator she just encountered.




(a) (b) (c)

**Fig. 1.** (a) A character with emergent behaviors has difficulty navigating in symmetric environments. (b) A roadmap. (c) Basic flocking dynamics with obstacle avoidance.

Although, they can be coupled with simple methods for guiding global flock movement [14], existing methods have difficulty if complex navigation is required, such as in cities, through crowded rooms, or over rough terrain. In contrast, path planning algorithms developed in the robotics community are capable of navigation in complex environments [7]. In particular we note the *roadmap*-based methods which construct, usually during preprocessing, a network of representative feasible paths in the environment. While roadmap methods can efficiently support complex navigation, they have generally not been customized to support coordinated group behavior.

In this work, we explore the benefits of integrating roadmap-based path planning techniques with flocking techniques. We extend ideas from cognitive modeling [5], and embed behavior rules in individual flock members and in the nodes and edges of the roadmap. We find that the global information provided by our *rule-based roadmaps* improves the behavior of autonomous characters and enables more sophisticated group behaviors than are possible using traditional flocking methods. Key features of our approach include:

- The roadmap provides a convenient abstract representation of global environmental information.
- Dynamic roadmaps (e.g., modifiable node and edge weights) enable communication between agents.
- Associating rules with roadmap components enables local customization of behaviors.

We illustrate our approach by proposing new techniques for four behaviors: homing, covering, goal searching, and traversing narrow passages. Our techniques can be applied to an entire flock, to individual flock members, or to an external agent that may influence the flock (e.g., a sheep dog) [2].

To our knowledge, this is the first time global maps have been used to support group behavior. However, Parker's work supports our use of global information to enable sophisticated group behaviors [12]. In particular, she

concluded that global knowledge should be used to provide general guidance for the longer-term actions of an agent, whereas local knowledge influences the more short-term, reactive actions. She also suggested that local information should be used to ground global knowledge in the current situation. This allows agents to remain focused on the overall goals of their group while reacting to the dynamics of their current situations

## 2 Related Work

Reynolds' influential flocking simulation [13] showed that flocking is a dramatic example of emergent behavior in which global behavior arises from the interaction of simple local rules. Each individual member of the flock (boid), has a simple rule set stating that it should move with its neighbors. This concept has been used successfully by researchers both in computer graphics and robotics. Tu and Terzopoulos [18] used flocking behaviors with intention generators to simulate a school of fish. They also demonstrated shepherding behavior in which a T-Rex herds raptors out of its territory.

A number of related methods for achieving group behaviors have been proposed [1,3,4,10,11,17,16,20]. An interesting approach by Vaughan et al. [19] used a robotic external agent to steer a flock of real geese.

Although there is little research on path planning for flocks, many methods have been proposed for planning for multiple robots. These methods can be characterized as centralized or decoupled. Centralized methods consider all robots as one entity, while decoupled methods first find a path for each robot independently and then resolve conflicts. In work from Li et al. [8], each group of crowds is guided by a leader and the paths of the leaders are generated using a decoupled approach.

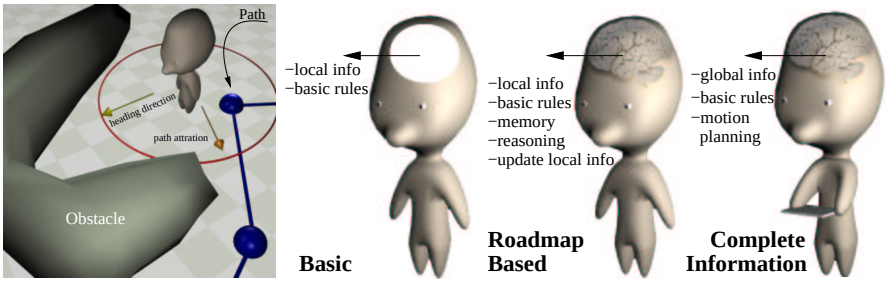
The observation of the behavior of ant colonies has inspired the ant colony optimization (ACO) meta-heuristic for discrete optimization. Dorigo et al. [9] exploit this ant-like behavior to optimize solutions for several NP-Complete problems. In our work, the flock's ability to explore comes from using an ACO-like approach to adaptively adjust roadmap edge weights

## 3 Roadmap Methods and Flocking Systems

A roadmap is a connectivity graph encoding representative feasible paths in the environment. Each node of the graph is a configuration of the robot that satisfies certain feasibility requirements (e.g., collision-free). The connections between nodes of the roadmap graph represent feasible (sub)paths. Several algorithms have been proposed to automatically generate roadmaps. Figure 1(b) shows an example of an automatically generated roadmap. We call a roadmap *adaptive* if its edge weights are updated according to new information gathered by flock members during simulation.

Basic flocking systems [13] model simple group behavior by providing individual members with simple rules that implement separation (to avoid collision with nearby neighbors), alignment (to move in the same direction as its neighbors) and coherence (to stay close to neighbors) maneuvers based on the positions and velocities of the flockmates inside the sensing range. Constraints are satisfied by generating forces for each rule and applying an integrated force to change the state of the flock member. In the presence of obstacles, force is also generated to push the flock member away. This basic system can be seen in Figure 1(c). More complicated behavior is usually simulated by adding other forces. As we will see in Section 6, flocking members sometimes are trapped in local minima of the potential field of these forces (see Figure 2(a)).

Flocking systems can be classified into three categories based on agent knowledge level (Figure 2(b)). In the basic system, an agent can only access information within its sensing range. On the other extreme, agents have complete information about the current state of the environment and can plan their motion. However, it is usually expensive to maintain complete information, and, more importantly, realistic simulation should not assume creatures always have such information and abilities. Like the basic system, agents in our roadmap-based system can only access local information. The global information (roadmap) is only locally available. This means each flock member can only access the environment and update the portion of the roadmap inside its sensing range. The agent's reasoning capabilities enable the agent to decide its next step based only on the local information. In addition, the agent is able to remember the places it has visited. With these simple mechanisms, we are able to simulate complex behaviors that are hard (expensive) to accomplish using the basic (complete information) system.




(a) (b)

**Fig. 2.** (a) This agent will be trapped in a local minimum of the path's attractive force. (b)Comparison between agents of different information and behaviors.

## 4 Rule-Based Roadmaps

The roadmaps encode global navigation information, but they usually do not include 'local' information for influencing behavior differently in different regions. We propose providing such information in the form of *behavior rules* which can be stored in the roadmap. When in the region of influence of a roadmap node (or edge), the agent's behavior is governed by the rule, if any, associated with that node. If no rule is specified, a default behavior will be followed. The embedded rules in the roadmap are analogous to the traffic signs on the streets which control cars.

Rules may be as simple as "Go To Next Node in Your Path" or as complex as "Wait For The Others, Select a Leader, Follow The Leader." Rules are assigned to nodes automatically in our implementation. For example, narrow passage behaviors (rules) are assigned to nodes on path segments that have small clearance. This process can be done manually as well, enabling customization.

### 4.1 System Overview

The general components of our system are one or more Agents and an Environment. Environment-aware Agent behavior is specified in Rules.

**System Components** Each Agent is a flock member, and it has several attributes associated with it, such as `position`, `goal`, `role`, and `path`. These are stored in variables called `AgentVar`s. When an Agent reaches a node, it starts executing that node's rule script. These rules might be executed only once, or at each time step that the Agent is in the region of influence of that node. Agents may also have dynamic (temporary) variables which are instantiated when rules are executed.

The Environment stores global information about the environment in the roadmap which is shared by all Agents. The roadmap nodes store Node attributes, such as node `position` and `edge`s and functions to assist navigation (e.g., returning the minimum weight edge leaving the node) in `NodeVar`s. The `NodeVar`s and functions are shared by all Agents running the Node's script (when they are within the Node's influence area). A Node may also have dynamic `NodeVar`s instantiated by the Node's script.

Rules describe Agent behavior. They are associated with regions of the Environment and are stored in roadmap Nodes. They can contain variables, assignments, control statements and loops. Triggers are special rules which are invoked to initialize and finalize variables governing behaviors. For example, in the narrow passage behavior, the first Agent to reach the passage is selected as the leader by the Trigger for that narrow passage. Triggers may be run only once, the first time an Agent reaches the associated Node, or they may run continuously until all Agents have left the region.

**System Control Structure** Our system is a single program. There is a main loop in the program which, at each time step, assigns AGENTs to NODEs, runs any necessary TRIGGERs, and then calls individual agent rule scripts. Since all the executions are sequential, the system is thread-safe, i.e., an AGENT's rule script won't be executed while another AGENT's rule script is running. For parallel or distributed implementation, synchronization issues have to be considered.

Rule scripts are stored with roadmap nodes, and are executed by AGENTs that are assigned to those nodes by the control loop (i.e., that are in the sphere of influence of those nodes). The structure of a rule script is as follows:

```
Definition of NodeVars
Definition of AgentVars
Definition of TRIGGERs
RULEs....
```

In our scripting language, the predefined variable `agent` refers to the AGENT running the script. All AGENTs have `agent.position` and `agent.goal` attributes providing their current position and goal position, and an `agent.role` attribute defining their current role, which is *boid* by default. The AGENT attribute `agent.path` has functions such as `currentPosition,nextPosition, index, push, pop`, etc. The predefined variable `node` in the script represents the NODE with which that script is associated. Each edge belongs to an `edge` structure made up of `start, end` and `weight` values. AGENTs can access the roadmap (or ENVIRONMENT) information through the predefined `node` variable or by calling functions which return information available in the `node` or `edge` structures, e.g., `node.edge(i)` which returns the $i$th edge.

## 5 Roadmap-Based Group Behavior

In this section, we show how roadmap-based techniques can be used to achieve different behaviors. We consider several behaviors: *homing*, *exploring (covering and goal searching)* and *traversing narrow passages*. The first two behaviors influence *where* the flock goes – reaching a pre-defined goal (homing), attempting to cover (visit) all reachable areas of the environment (covering) or search for a goal whose location is not known. The narrow passage behavior influences *how* the flock members position themselves relative to each other when they move through the passage.

### 5.1 Homing Behavior

Homing behavior is usually simulated by adding an attractive force toward the goal [6]. However, this method may easily be trapped in a local minimum even in a simple environment. A method commonly used in computer games

requiring motion of a group of objects is a grid-based $A^*$ search [15]. In this approach, the environment is discretized to small grid cells and the search for the flock's path is based on expanding toward the most promising neighbor of already visited positions. Although $A^*$ search finds shortest paths and it is usually fairly fast, it does have drawbacks. Of particular note here is the necessity of finding a completely new path for each new goal which reduces efficiency and increases the computation time for complex environments.

In contrast, roadmap-based path planning methods work on a global scale and once the roadmap is generated, finding new paths is fast and efficient. In our approach, we use MAPRM [21] to find a path for the flock. One of the advantages of MAPRM is that the paths we find tend to have large clearances from obstacles. Once a path is found, individual flock members follow the path. The path is partitioned into a set of subpaths (identified by subgoals) based on the individual flock member's sensor range. Each member keeps track of subgoals and as soon as a subgoal comes within sensory range the next subgoal becomes the steering direction for the global goal.

With other interacting forces from neighboring flock members and obstacles, steering toward the subgoal has the lowest priority, so individual members still move together while moving toward goal. This results in a flocking toward the goal and avoids getting trapped in local minima. The homing behavior is shown in Algorithm 5.1. The script to implement this algorithm in our system is Script 5.1.

**Algorithm 5.1** Homing

```
1: if (goal is in view range) then
2:     set goal as target.
3: else if (target is in view range) then
4:     set next subgoal as the target.
5: end if
6: steer toward the target.
```

(a)

**Script 5.1** Homing

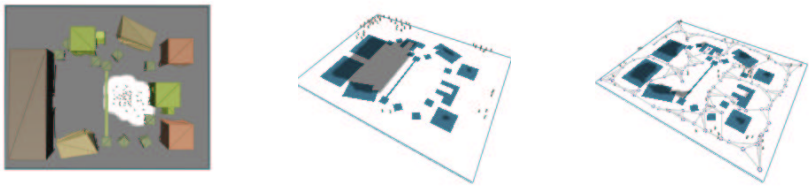
```
agent.goal = agent.path.next();

//don't run again at
//this node

stop;
```

(b)

**Fig. 3.** Homing Behavior: (a) algorithm, (b) script to implement algorithm.

### 5.2 Exploring Behavior

There are several variants of exploring. For example, flocks may have good *a priori* data about the environment and they may be looking for objects whose locations are unknown, or the environment may be unknown and the objective of the exploring behavior is to gather information about the environment. For our work, we assume the environment is known. We consider two



(a) (b) (c)

**Fig. 4.** Covering behavior. Unexplored regions are shaded. (a) Initial configuration of flock. (b) Normal flocking system. (c) Roadmap-based method.

different exploring behaviors: (i) covering the environment, and (ii) searching for a goal and moving toward the goal once it is found. To achieve these behaviors, we use a roadmap with adaptive edge weights. Each agent moves independently in the roadmap. At each roadmap node, they choose an edge to follow based on the weights of the incident edges. The weights represent how relevant the edge is to the current task, either covering the environment or searching for and reaching the goal. This is similar to *ant colony optimization (ACO)* where the ants deposit pheromones to indicate routes.

**Covering Behavior** The objective in this behavior is for some member of the flock to have visited every location in the environment. We assume that we start with a roadmap covering all relevant portions of the environment, and our goal is to have some flock member visit every edge and vertex of the roadmap. We implement this behavior using a roadmap graph with adaptive edge weights. Initially, all edges have the same weight. As they traverse a roadmap edge they increase its weight – this is similar to ant pheromones which increase as more ants follow the same path. Since our goal is to cover the environment, individual flock members are biased toward relatively unexplored areas of the roadmap, i.e., to roadmap edges that have smaller weights. That is, they select low weight edges to follow with some higher probability. The covering behavior is shown in Algorithm 5.2 and the corresponding script is Script 5.2. The first two lines of Script 5.2 define local variables. The third line calls a function of the predefined `node` variable which returns, with some probability, the minimum weight outgoing edge. Then, the next goal is set to that edge's second endpoint and the edge weight is increased.

**Goal Searching Behavior** Although the individual flock members have access to environmental information through the roadmap (at least locally), they don't know the location of the goal. If an individual reaches a location where the goal is within sensor range, its location is communicated to the other members, perhaps indirectly, and they then attempt to reach the goal as well. We implemented this behavior using adaptive roadmap edge weights.

**Algorithm 5.2** Covering the Env.

```
1: if (reach target t) then
2:    Push t onto agent's stack
3:    if (no outgoing edge) then
4:       t = topmost stack node with un-
         visited edge
5:    else
6:       select edge e and increase wt(e)
7:       t = endpoint(e)
8:    end if
9: end if
10: steer toward the target
```

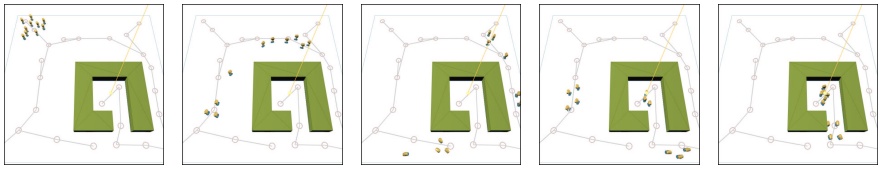
(a)

**Script 5.2** Covering the Env.

```
AgentVar NODE nextNode;
AgentVar EDGE edge;

edge = node.probMinEdge();
nextNode = edge.end;
agent.goal = nextNode.position;
edge.weight = edge.weight+1;
```

(b)

**Fig. 5.** Covering Behavior: (a) algorithm, (b) script to implement algorithm.



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

**Fig. 6.** 10 flock members are searching an unknown goal. (a) flock is facing a split point, (b) since two branches have same edge weight, flock splits into two groups, (c) members on the lower (and upper right) part encountered dead end, and weight of edges leading to dead end will decrease, (d) some members find goal, weight of edges leading to the goal will increase, (e) rest of members reach the goal.

As an agent traverses a path in the roadmap, it remembers the route it has taken. Then, when it reaches a goal, it increases the weight of the edges on the route it took. If the individual reaches a roadmap node without any outgoing connections (i.e., with only one edge) or a node already contained in the current path (i.e., a cycle), the weight of the edges it followed will be decreased. The algorithm for this approach is summarized in Algorithm 5.3.

This algorithm can be implemented using two different rule sets. The first rule set is for all roadmap nodes except the (hidden) goal, it adds the visited nodes to the AGENT's path. The second rule set will only be executed if the agent reaches the goal, and the task of this set is to increase the weight of edges this agent used to get there. Script 5.3.I. implements the first set of rules. The first two lines define local variables. The third line calls a function of the predefined `node` variable which returns, with some probability, the maximum weight outgoing edge. Then, the next goal is set to that edge's second endpoint and this information is added to path so that the promising

**Algorithm 5.3** Goal Searching

```
1: if (goal found) then
2:    increase weight of edges to goal.
3: end if
4: if (reach target t) then
5:    Push t onto agent's stack
6:    if (no outgoing edge) then
7:       t = topmost stack node with
         unvisited edge. Decrease edge
         weights leading to t
8:    else
9:       select edge e and increase wt(e)
10:      t = endpoint(e).
11:   end if
12: end if
13: steer toward target t.
```

(a)

**Script 5.3.I** Goal Searching

```
AgentVar Node nextNode;
AgentVar Edge edge;
Trigger checkDeadEnd;
edge = node.probMaxEdge();
nextNode = edge.end;
agent.goal = nextNode.position;
agent.addPath(nextNode.position);
```

(b)

**Script 5.3.II** Hidden Goal

```
// Update the passed edges
agent.path.increaseEdgeWeights();
stop;
```

(c)

**Fig. 7.** Goal Searching: (a) algorithm, scripts for (b) searching, and (c) goal nodes.

edges can be updated if the Agent reaches the goal. The Trigger function checkDeadEnd will run once for every node and would check the path so far to see if there is a dead end, and if so decrease the weight of edges and remove them from the path. Script 5.3.II, which only resides in the node closest to the goal, will increase the weight of the edges on the path Agent took to the goal.

### 5.3 Narrow Passage Behavior

Sometimes the flock's behavior should depend on the surrounding environment. For example, different group formations may be used in relatively open areas than are used when passing through narrow regions.

A naive way to achieve narrow passage traversal by the flock is to use the homing behavior and to select two nodes as goals, first a node in front of the entrance to the passage and then a node outside the exit from the passage. One drawback of this approach is that flock members may bunch up and conflict with each other as they try to move through the passage.

A *follow-the-leader* strategy may avoid the congestion problems of the naive strategy. In this strategy, we first assemble the flock in front of the narrow passage, and then select the closest agent to the narrow passage entrance as the leader. Then, the remaining flock members are arranged into a queue that follows the leader. Their position in the queue depends on their distance to the entrance of the narrow corridor. They can be kept from crowding each

other by selecting appropriate values for the repulsive force from other flock members.

Note that different behaviors can be achieved by using a different criterion to select the next flock member in line 6 of Algorithm 5.4. For example, instead of selecting the next closest flock member to the narrow passage, one might select the farthest, which would create a 'milling around' effect at the entrance to the passage.

**Algorithm 5.4** Narrow Passage

```
 1: while (not all flock members in gathering area) do
 2:    set individual members' goal to gathering area
 3: end while
 4: set leader to NIL
 5: while (there are flock members outside passage) do
 6:    select the closest unselected member as Current
 7:    if (Leader is NIL) then
 8:       set Leader to Current and set Leader's goal to next step in the path
 9:    else
10:       set Current's goal to Previous
11:    end if
12:    set Previous to Current
13:    increase neighbor avoidance threshold
14: end while
```

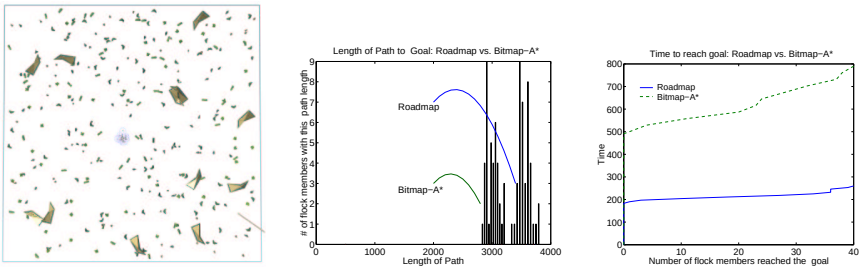
**Fig. 8.** Narrow Passage Behavior algorithm.

A script for the NARROW PASSAGE behavior can be found in [2]. Initially, all agents are in the same class. After they all have gathered outside the passage, one of them is selected as the *leader* and the others become *followers*; these roles are maintained while traversing the passage. There is a TRIGGER function that continuously checks if all AGENTs have reached the gathering area. When they are there, the TRIGGER updates a `NodeVar` variable which keeps track of the state so that the follow behavior starts. A second TRIGGER function continuously finds the closest AGENT to the passage and updates its role (the first time a closest AGENT is found, it is the *leader*, the successive ones are *followers*). Then, the AGENTs running the script compare themselves with the closest, if they are closest then they either follow their pre-assigned path (the leader) or follow the previous AGENT.

**Table 1.** The number of the 40 flock members reaching their home within 30 seconds, the initialization time, the average time to find a path, and the total time spent by all flockmates escaping local minima using basic flocking, grid-based $A^*$, and rule-based roadmap behaviors.

**Homing behavior: Comparison of different methods**

| Behavior Method | #flockmate reaching the goal | init time | find path time | local minima # | local minima escape(s) |
|---|---|---|---|---|---|
| Basic | 10 | N/A | N/A | N/A | N/A |
| grid-based A* | 40 | 6.02 | 5.757 | 2005 | 1035.43 |
| roadmap-based | 40 | 0.88 | 0.652 | 255 | 22.99 |




(a) (b) (c)

**Fig. 9.** Homing behavior: the number of flock members reaching (a) goals with respect to the length of the paths they took, and (b) homes with the evolution of time. Although the grid-based $A^*$ behavior finds shorter paths, the flock spends less time to reach the goals with the roadmap-based behavior.

## 6 Experimental Results

In this section we evaluate our rule-based roadmap techniques for the homing, covering, and narrow passage behaviors described previously. Movies can be found on our webpage (http://parasol.tamu.edu/dsmft/).

Our experiments are designed to compare our roadmap-based techniques with more traditional approaches for simulating flocking behavior and to study the improvements possible by incorporating global information about the environment as encoded in a roadmap. To study the efficiency of our covering technique, we also compare our roadmap-based method with an 'ideal' variant in which the location of all unvisited nodes and edges in the (known) roadmap is known at all times.

All of our experiments were run on a Linux system with Athlon 1.33 processor and 256MB memory.

### 6.1 Homing Behavior

For the homing behavior, our rule-based roadmap technique, described in Section 5.1, is compared with a basic flocking behavior using a potential field and a grid-based $A^*$ search behavior.

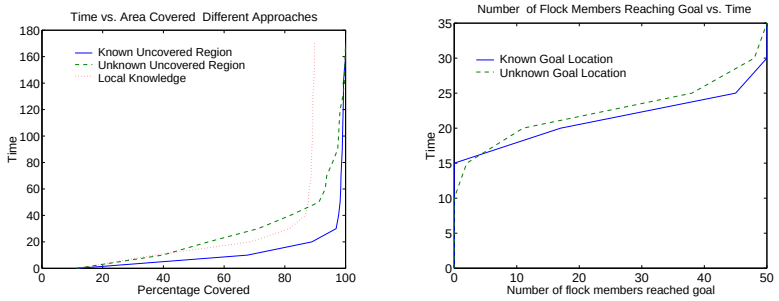
The $420 \times 420$ environment is populated with 301 randomly placed obstacles (of six types) (see Figure 9(a)). At any given time there is one goal, and when all flock members reach it, a new goal is randomly generated; this process continues until eight goals have been generated and reached. There are 40 flock members which are initially placed according to a Gaussian distribution around the center of the environment. The simulation is updated every 100 ms.

For the grid-based $A^*$ behavior, a bitmap of the environment of $914 \times 914$ cells is constructed; the length of a side of each square cell is equal to the diameter of a flock member. Cells are classified as free cells and collision cells. Paths are found in this bitmap using $A^*$ search. For the rule-based roadmap behavior, the roadmap is built using the MAPRM method to generate 400 roadmap nodes and we attempt to connect each node to its 4 nearest neighbors.

In our experiments we first tried to evaluate the effect of the global information. In order to do that, we first compared different methods to see how many flock members can reach a given goal with predetermined time (See Table 1, second column). It is clear from Table 1, without global information, that only 25% of the flock members reach the goal and most of them are trapped in local minima. On the other hand, when global navigation information is utilized, either with the grid-based $A^*$ method or our rule-based roadmap method, all flock members reach the goal. Based on these results, we further evaluated the grid-based $A^*$ and the roadmap-based method. In Table 1, columns 3–6, we show the time spent searching for paths, the number of local minima encountered along all paths, and the total time spent escaping from local minima. This offers some insight into the methods studied, as can be seen more clearly in Figure 9. Although the flock takes a shorter path with the grid-based $A^*$ search than with the roadmap-based method (Figure 9(b)), the flock reaches the final goal faster with the roadmap-based method (Figure 9(c)). As $A^*$ search is known to be fast and always finds shortest paths, this example illustrates that our roadmap-based method indeed is a competitor for grid-based $A^*$ methods – while the paths found are a bit longer, they are found faster.

### 6.2 Covering Behavior

We next consider the covering behavior. This behavior is tested on the environment shown in Figure 4, which requires flock members to pass through narrow passages to access undiscovered areas. In this experiment, we compare basic flocking behavior, rule-based roadmap behavior, and an ideal variant of the roadmap-based behavior which has dynamic knowledge of the undiscovered regions. The former comparison enables us to establish the value of the global information stored in the roadmap, while the latter comparison enables us to determine how far our methods are from achieving optimal performance.




(a) (b)

**Fig. 10.** Exploring behavior: (a) Covering: the percentage of the environmental covered in terms of time, (b) Goal Searching: the number of flock members reaching the goal area in terms of time.

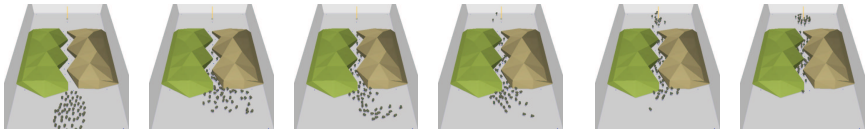


**Fig. 11.** Passing through a narrow passage using the FOLLOW THE LEADER behavior (Algorithm 5.4).

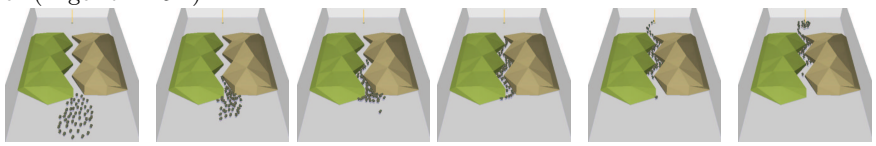


**Fig. 12.** Passing through a narrow passage using the naive strategy – homing to the exit node of the passage.

The environment ($80 \times 100$) is populated with 16 obstacles (6 types of obstacles) and in total 24% of the environment is occupied by obstacles. 50 flock members are simulated and states are updated every 100ms. A bitmap is built to record discovered/undiscovered information. A bitmap cell is discovered when it is inside the sensory range of any flock member. We set the radius of the sensory circle at 5m. For the roadmap, 120 nodes are sampled and connections are attempted to each node's 4 nearest neighbors.

The roadmap-based covering behavior is as described previously. The basic behavior uses only local information, and is essentially a random walk through the environment. It shows that the lack of global knowledge results in some areas never being discovered, especially those nearly surrounded by obstacles. The ideal behavior with complete knowledge of the undiscovered locations uses the roadmap to find paths from a flockmate's current position to the closest unexplored spot. Although such knowledge is generally not available, this variant indicates how fast an environment can be covered in the best case.

As seen in Figure 10(a), the ideal behavior rapidly covered all of the free environment in the first 30 seconds. The rule-based roadmap behavior, using indirect communication (adaptive edge weights) takes about three times as long (90 seconds), to reach a similar coverage. Nevertheless, like the ideal behavior, the rule-based roadmap behavior found most reachable areas, while the basic flocking behavior had difficulty covering more than 89% of the free environment. However, it is interesting to note that the basic flocking behavior found more undiscovered areas than the rule-based roadmap approach in the first 40 seconds; this is due to the flockmates' with basic behavior which tend to bounce around and discover 'easy' areas very quickly.

### 6.3 Goal Searching Behavior

In this experiment, the rule-based roadmap behavior is compared with simple flocking behavior that has only local information about the environment and no knowledge of the goal position and with an ideal variant of the roadmap-based behavior that has *a priori* knowledge of the position of the goal. The environment is the same as in the covering experiment (See Figure 4).

We are interested in the number of flock members that reach the goal and how fast they get there. As mentioned before, the behavior with complete knowledge is used to establish a best case (lower bound) for the simulation efficiency, and the basic behavior using only local information is used to illustrate the importance of global knowledge.

The results of some experiments are shown in Figure 10(b). Flocks using the basic behavior did not discover any goals within 35 seconds, and hence this behavior is not shown in the plot. Overall, the roadmap-based behavior is competitive with the ideal roadmap-based behavior – only taking 5 seconds longer than the method in which the position of the goal is known *a priori*. In addition, it is surprising to note that two of the flock members in the roadmap-based method reach the goal earlier than any of their flockmates in the ideal roadmap-based behavior. While we expect the roadmap-based method to continue to perform well in more complex environments, we expect its efficiency relative to the ideal method to decline somewhat.

### 6.4 Narrow Passage Behavior

The narrow passage environment in Figures 11 and 12 contains 50 flock members and two mountain-like obstacles. Agents are asked to reach the goal on the other side of environment. The only way to reach their destination is to pass through the narrow passage between these two obstacles. As before, the roadmap is generated using MAPRM [21] which attempts to generate high clearance nodes on the medial axis of the free space. To demonstrate our rules in Section 5.3, we consider two experiments.

In the first experiment, a node is automatically assigned the narrow passage rule described in Section 5.3 if its clearance is below some threshold. All

other nodes are assigned homing rules as described in Section 5.1 with the goal set as a node outside the exit of the narrow passage. The homing rule script also sets `agent.role = boid`, so that all AGENTs are boids when they gather outside the entrance to the narrow passage. Snapshots of the agent movement are shown in Figure 11. Note how the agents maintain clearance from each other as they move through the passage. The amount of clearance is adjusted by modifying the repulsive force between agents, which can be set in the rule scripts stored in narrow passage nodes.

In the second experiment, homing rules are assigned to all nodes, with the goal of the homing rules set to the node at the exit of the narrow passage. This is the naive narrow passage behavior mentioned in Section 5.3, which is essentially a homing behavior. Snapshots of the agent movement are shown in Figure 12. Note how the agents clump together and interfere with each other as they move through the passage.

## 7 Conclusion

In this paper, we have shown that complex group behaviors can be generated if some global environmental information is available in a roadmap. The information the roadmap contains, such as topological information and adaptive edge weights, enables the flock to achieve behaviors that cannot be modeled with local information alone. Moreover, since in many cases global knowledge involves high communication costs between individuals, indirect communication though dynamic updates of the roadmap's edge weights provides a less expensive means of obtaining global information. The behavior rules embedded in our roadmaps and agents enable the agents to modify their actions based on their current location and state. For example, the flock can move as an unordered group in open regions and in a follow-the-leader fashion through narrow passages. Our simulation results for the three types of behaviors studied show that the performance of the rule-based roadmap behaviors is very close to ideal behaviors that have complete knowledge.

## References

1. T. Balch and M. Hybinette. Social potentials for scalable multirobot formations. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 73–80, 2000.
2. O. B. Bayazit, J.-M. Lien, and Nancy M. Amato. Better group behaviors in complex environments using global roadmaps. In *Artif. Life*, Dec 2002. To appear.
3. D. C. Brogan and J. K. Hodgins. Group behaviors for systems with significant dynamics. In *Autonomous Robots*, pages 137–153, 1997.
4. T. Fukuda, H. Mizoguchi, K. Sekiyama, and F. Arai. Group behavior control for MARS (micro autonomous robotic system). In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1550–1555, 1999.

5. J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for interlligent characters. In *Computer Graphics*, pages 29–38, 1999.
6. O. Khatib. Real–time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5(1):90–98, 1986.
7. J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
8. T. Y. Li, Y. J. Jeng, and S. I. Chang. Simulating virtual human crowds with a leader-follower model. In *Proceedings of 2001 Computer Animation Conference*, 2001.
9. Dorigo M., G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. In *Artificial Life*, pages 137–172, 1999.
10. M. J. Mataric. *Interaction and Intelligent Behavior*. PhD thesis, MIT EECS, 1994.
11. S.I. Nishimura and T. Ikegami. Emergence of collective strategies in prey-predator game model. *Artif. Life*, 3:243–260, 1997.
12. L. E. Parker. Designing control laws for cooperative agent teams. In *IEEE International Conference on Robotics and Automation*, pages 582–587, 1993.
13. C. W. Reynolds. Flocks, herds, and schools: A distributed behaviroal model. In *Computer Graphics*, pages 25–34, 1987.
14. C. W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, 1999.
15. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1th edition, 1994.
16. N. Saiwaki, T. Komatsu, T. Yoshida, and S. Nishida. Automatic generation of moving crowd using chaos model. In *IEEE Int. Conference on System, Man and Cybernetics*, pages 3715–3721, 1997.
17. S.-J. Sun and D.-W Lee K.-B. Sim. Artificial immune-based swarm behaviors of distributed autonomous robotic systems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3993–3998, 2001.
18. X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Computer Graphics*, pages 24–29, 1994.
19. R. T. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron. Experiments in automatic flock control. *J. Robot. and Autonom. Sys.*, 31:109–117, 2000.
20. C.R. Ward, F. Gobet, and G. Kendall. Evolving collective behavior in an artificial ecology. *Artif. Life*, 7:191–209, 2001.
21. S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1024–1031, 1999.