Better Group Behaviors in Complex Environments using Global Roadmaps*

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

Department of Computer Science, Texas A&M University, College Station, TX 77843-3112
{burchanb,neilien,amato}@cs.tamu.edu

Abstract

While many methods to simulate flocking behaviors have been proposed, these techniques usually only provide simplistic navigation and planning capabilities because each flock member's behavior depends only on its local environment. In this work, we investigate how the addition of global information in the form of a roadmap of the environment enables more sophisticated flocking behaviors and supports global navigation and planning. In this paper, we propose new techniques for four distinct group behaviors: homing, goal searching, traversing narrow areas and shepherding. Extending ideas from cognitive modeling, we embed behavior rules in individual flock members and in the roadmap. These embedded behaviors enable the creatures to modify their actions based on their current location and state. For example, the flock might move as an unordered group in open regions and in a follow-the-leader fashion through narrow passages. These behaviors exploit global knowledge of the environment and utilize information gathered by all flock members which is communicated by allowing individual flock members to dynamically update the shared roadmap to reflect (un)desirable routes or regions. We present experimental results showing how the judicious use of simple roadmaps of the environment enables complex behaviors to be obtained at minimal cost. Animations can be viewed at http://parasol.tamu.edu.

Introduction

The ability to simulate the coordinated movement of a group of creatures plays an important role in artificial life. For example, birds fly in flocks, fish swim in schools, and sheep move as a herd. An artificial representation of such creatures requires techniques for generating the motion of the individual entities within the flock and techniques for directing the global movement of the flock. While such methods have attracted much attention, most research has focused on techniques for modeling individual behavior within a flock, such as Reynolds' *boids* (Reynolds 1987). These techniques have been coupled with simple methods for guiding global flock movement, such as attractive potential fields centered at a goal location, to achieve realistic group movement in simple environments with few external obstacles, such as birds in the air or fish in the sea. Usually in this approach the flock shows homogeneous behavior. Tu and Terzopoulos extended Reynolds' flocking system to create more realistic, self-animating characters through biomechanical modeling with a behavioral finite state machine, the intention generator (Tu & Terzopoulos 1994). They also propose the idea of *cognitive modeling* (Funge, Tu, & Terzopoulos 1999), which controls how characters gather knowledge and how they act. Their *cognitive modeling language, CML*, eases an animator's task of specifying character behaviors. In brief, *cognitive modeling* provides an alternative to hard-coding for embedding behaviors into programs. It also enables heterogeneous flocks – different behaviors for different flock members. Unfortunately, these existing methods do not perform well if complex navigation is required, such as in cities, through crowded rooms, or over rough

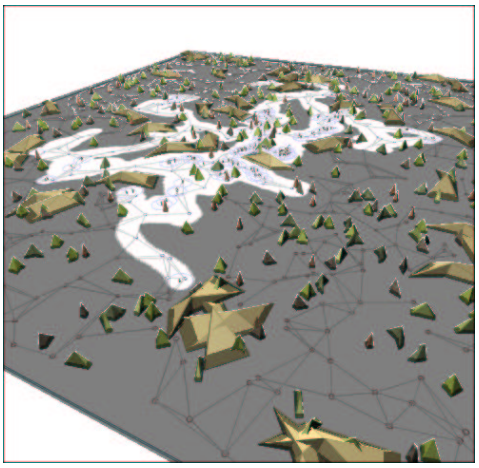


Figure 1: Complex behaviors, such as covering, can be improved using global information.

*This research supported in part by NSF CAREER Award CCR-9624315, NSF Grants IIS-9619850, ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACI-0113971, CCR-0113974, EIA-9810937, EIA-0079874, and by the Texas Higher Education Coordinating Board grant ARP-036327-017. Bayazit is supported in part by the Turkish Ministry of Education.

terrain. The ability to generate animations in complex environments is poor even if more elaborate behaviors are encoded in CML. This is due to, so-called, *emergent behavior* in which characters only react to immediate events.

In contrast, path planning algorithms developed in the robotics community are capable of navigation in complex environments (Latombe 1991). In particular we note the *roadmap methods* which can quickly answer many diverse path planning queries using a map, typically constructed during preprocessing, containing a network of representative feasible paths in the environment. In essence, these maps function similarly to driving maps in that one plans a route by first locating their initial and final positions and then selecting a route connecting them from the roads and highways shown on the map. While many good path planning algorithms exist, they have traditionally only been used to plan paths for a single moving object (the 'robot'). That is, roadmap methods have not been used to support group behavior.

Our Contribution

In this work, we explore the benefits of integrating roadmap-based path planning techniques with flocking techniques. We extend ideas from cognitive modeling (Funge, Tu, & Terzopoulos 1999), 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 in particular, enables more sophisticated group behaviors than are possible using traditional (local) flocking methods. Key features of our approach include:

- The roadmap provides a convenient abstract representation of global information in complex environments.
- Adaptive roadmaps (e.g., modifying node and edge weights) enable communication between agents.
- Associating rules with roadmap nodes and edges enables local customization of behaviors.

We illustrate the power of our approach by proposing new approaches for four behaviors: homing, goal searching, traversing narrow passages and shepherding. Our new 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).

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 (Parker 1993). 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.

Related Work

Reynolds' influential flocking simulation (Reynolds 1987) established the feasibility of modeling such a system. His work 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 (1994) used flocking behaviors with intention generators to simulate a school of fish in artificial life. Later, they implemented a search over possible situations expressed in formal logic (Funge, Tu, & Terzopoulos 1999). They also demonstrated shepherding behavior in which a T-Rex herds raptors out of its territory.

Nishimura and Ikegami (1997) used flocking dynamics to investigate collective strategies in a "prey-predator" game model. Ward et al. (2001) studied an evolving sensory controller for producing schooling behavior based on "boids". Brogan and Hodgins (1997) investigated group behavior with significant dynamics, such as human-like bicycle riders. Sun et al. (2001) achieve swarm behaviors based on a biological immune system. Balch and Hybinette (2000) propose a behavior-based solution to the robot formation-keeping problem. Fukuda et al. (1999) describe group behavior for a Micro Autonomous Robotics System. Mataric (1994) classifies a basic set of group behaviors which can be used to create more complex behaviors including flocking and herding. Saiwaki et al. (1997) use a chaos model to simulate a moving crowd. An interesting approach by Vaughan et al. (2000) 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. (2001), 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. (1999) 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.

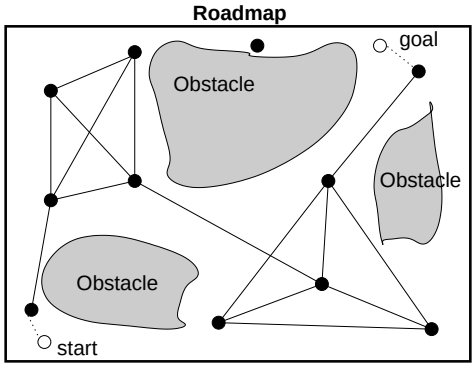


Figure 2: Querying a roadmap.

Rule-Based Roadmap Path Planning

Roadmap methods are among the most effective motion planning methods (Latombe 1991). These methods are based upon a map, usually computed during preprocessing, encoding representative feasible paths in the environment. Paths are planned by first locating their start and goal positions in the roadmap and then determining a route between them from the pathways in the roadmap. (See Figure 2.)

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 pre-defined *behavior rules* which can be stored in the roadmap. When in the region of influence of a roadmap node (or edge), the group members' behavior is governed by the rule, if any, associated with that node. If no rule is specified, a default behavior will be followed. Rules may be as simple as

RULE: Go To Next Node in Your Path

or as complex as

RULE: 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.

System Overview

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

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. In addition to pre-defined variables common to all Agents, recording properties such as position, 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). In addition to the pre-defined `NodeVar`s, 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 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.

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 `currentPosition, index, nextPosition, 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 ith edge.

Roadmap-Based Group Behaviors

In this section, we show how rule-based roadmap techniques can be used to achieve different behaviors. We consider four behaviors: *homing*, *goal searching*, *traversing narrow passages* and *shepherding*. The first two behaviors influence *where* the flock goes – reaching a pre-defined goal (homing) or searching for a goal (goal searching). The narrow passage behavior influences *how* the flock members position themselves relative to each other when they move through the passage. In the shepherding behavior an external agent influences the flock.

Homing Behavior

Homing behavior consists of two sub-models, one representing the individual behavior of flock members and the other influencing the global behavior of the flock. *"Boid"* dynamics (Reynolds 1987) sufficiently model individual behavior in most cases. In this model, individual members should: (i) avoid collision with neighboring flockmates, (ii) match velocity with them, and (iii) stay close to their neighbors. The neighborhood is defined by a distance, and an individual member of the flock is steered by angle and directional vectors satisfying the above criteria.

Global behavior is usually simulated using potential field methods by adding two directional vectors (Khatib 1986): one toward a goal and one away from nearby obstacles. However, this method may easily be trapped in a local minimum in an environment crowded with obstacles. A method commonly used in computer games requiring motion of a group of objects is a grid-based A^* search (Russell & Norvig 1994). 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 the efficiency of this approach and increases the computation time for complex environments.

In contrast, roadmap-based path planning methods have a global view and once the roadmap is generated, finding new paths is fast and efficient. In our approach, we use a *probabilistic roadmap method* (PRM) (Amato *et al.* 1998; Kavraki *et al.* 1996) motion planning method called MAPRM (Wilmarth, Amato, & Stiller 1999), to build the roadmap automatically and 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 discretized to subgoals based on an individual flock member's sensor range. Each member keeps track of subgoals and as soon as a subgoal comes within the 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 behavior is summarized in Algorithm I.

Algorithm I HOMING

```
01. for (each individual flock member)
02.     if (goal is in view range)
03.         stay near goal
04.     else if  (current subgoal is in view range)
05.         set next subgoal as the target
06.     else
07.         steer toward the target
08.     endif
09. endfor
```

The script to implement this behavior would be:

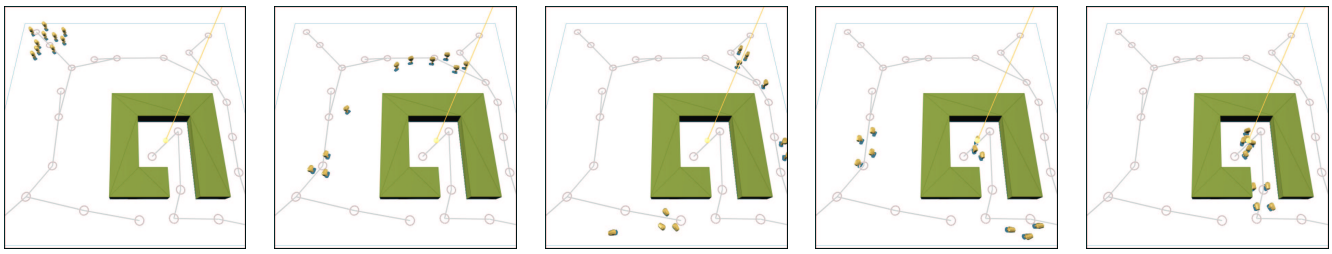
```
//Homing Script
agent.goal = agent.path.next();
stop; //don't run again at this node
```

Thus, as soon as an AGENT reaches a node within the path, the node's rule selects the next node in the AGENT's path as the goal.

Goal Searching Behavior

Goal searching is a type of exploring behavior. We assume the environment is known and the objective is to search for a goal and then move toward it. We achieve this behavior using a roadmap graph with adaptive edge weights. Each individual member behaves independently from its flock mates and uses the roadmap to wander around. Specifically, they follow roadmap edges and there are no predefined paths. If they reach a roadmap node with several roadmap edges, they probabilistically choose a roadmap edge to follow based on the weight of the edge. The edge weights represent any preferences for the current task, i.e., searching for and reaching the goal.

Our goal searching behavior is similar to *ant colony optimization (ACO)*. Although the individual flock members know the environment, 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



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

Figure 3: Ten flock members are searching for an unknown goal. (a) The flock faces a branch point. (b) Since both edges have the same weight, the flock splits into two groups. (c) After dead ends are encountered in the lower left and upper right, edge weights leading to them are decreased. (d) As some members find the goal, edge weights leading to it are increased. (e) The remaining members reach the goal.

attempt to reach the goal as well. We implemented this behavior using adaptive edge weights. The weight of an edge indicated how important it is believed to be, and edges leaving roadmap nodes are selected with some probability based on their weights. As an individual 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. See Figure 3.

The algorithm for this approach is summarized below.

Algorithm II GOAL SEARCHING

```
01. for (each flock member)
02.     if (goal found)
03.         increase edge weights on path to goal
04.     else if (dead end found)
05.         pop stack until a new branch is found
06.         decrease weight of edge corr. to popped node
07.     else
08.         select a neighboring node of the current node
09.         push this node onto the stack
10.     endif
11. endfor
```

This algorithm can be implemented using two different rule sets: one for all roadmap nodes except the node with the goal, and one for the roadmap node with the goal. The first rule set adds the visited nodes to the AGENT's path. The second set, which is only executed if the agent reaches the goal, will increase the weight of the edges the agent used to reach the goal. The script to implement the first set of rules is:

```
// Searching Script
AGENTVAR Node nextNode;
AGENTVAR Edge edge;
TRIGGER checkDeadEnd;
edge = node.probMaxOutEdge();
nextNode = edge.end;
agent.goal = nextNode.position;
agent.addPath(nextNode.position);
```

The first three lines define local variables and a trigger function. The fourth line calls a function of the predefined `node` variable which returns, with some probability, the maximum weight outgoing edge. Then, the AGENT's next goal is set to that edge's second endpoint and this information is added to path so that the AGENT can remember the route it has taken.

The goal searching behavior differs from the homing behaviors in that we need to monitor the AGENTS' path to deal with dead ends. We use special RULES called TRIGGERS for this purpose. TRIGGERS are instantiated when an AGENT reaches the associated NODE, and they may be run only once, or they may run *continuously* until some termination condition is met. For example, in this case, the TRIGGER function *checkDeadEnd* will run once for AGENTS in its influence range. It will check to see if the path so far contains a dead end, and if so, will remove the edges leading to it from the path and decrease their weights so other agents will be less likely to traverse them.

The following rule, which only resides in the node closest to the goal, will increase the weight of the edges on the path the AGENT took to the goal.

```
// Update the passed edges
agent.path.increaseEdgeWeights();
stop; //don't run again at this node
```

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 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 flock member to the entrance to the narrow passage 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. This strategy provides more of a flow effect.

Note that different behaviors can be achieved by using a different criterion to select the next flock member in line 6 of Algorithm III. 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 III NARROW PASSAGE

```
01. while NOT all flock members in gathering area
02.    set individual members' goal to gathering area
03. endwhile
04. set leader to NIL
05. while there are flock members outside passage
06.    select the closest unselected member as Current
07.    if Leader is NIL
08.       set Leader to Current
09.       set Leader's goal to next step in the path
10.    else
11.       set Current's goal to Previous
12.    endif
13.    set Previous to Current
14.    increase neighbor avoidance threshold
15. endwhile
```

A script for the NARROW PASSAGE behavior is given below. Initially, all agents are *boids*. 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.

```
NODEVAR previous=-1; //define node vars
NODEVAR closest;
NODEVAR state=wait; //define initial state
TRIGGER continuous:  waitGathering;
TRIGGER continuous:  selectClosest;
  IF (agent.role==boid && state==wait) { //wait
    agent.goal=node.position; //go to gathering area
  }
  ELSE IF (agent.role==leader) { //if leader
    agent.goal=agent.path.next(); //follow path
    stop; //don't run again at this node
  }
  ELSE IF (agent.role==follower) { //if follower
    agent.path.next(); //update path (not goal)
    stop; //don't run again at this node
  }
  ELSE { // agent is boid and state is not wait
    IF(closest==agent.id) { //agent is closest
      IF (previous==-1) { //no leader yet
       agent.role=leader;
       agent.goal=agent.path.next();//follow path
      }
      ELSE { //join queue - follow previous
       agent.role=follower;
       agent.goal=agent(previous);
       agent.path.next(); //update path
      }
      previous=agent.id; //update previous
      stop; //don't run again at this node
    }
  }
```

The *waitGathering* TRIGGER function continuously checks if all AGENTs have reached the gathering area. When they are there, it sets the NODEVAR state so that the follow behavior starts. The *selectClosest* TRIGGER function continuously runs. It can access and update variables applicable to all AGENTs within the region of the NODE (i.e., previous and closest) as well as private variables for each AGENT (i.e., role). At each iteration, it finds a new closest AGENT and sets the *closest* variable. 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.

Shepherding Behavior

In the previous sections we have observed distinct flocking behaviors In the homing and narrow passage behaviors, the flock members were moving toward a goal together, i.e., as a flock. The motion was planned for the flock. In the goal searching behavior, the flock members were exploring and planning their motions individually. In a sense, the flock had control of the motion in the first case and individual flock members had control in the second case. In our last scenario, control is ceded to an outside agent who guides, or shepherds them. In the simulation snapshots shown in Figure 6, the external agent is a dog whose objective is to move the flock of sheep toward the goal. The only motion control for the flock is to move away from the dog. Vaughan et al. (2000) provide a similar implementation in which a robot was programmed to move geese toward a goal position. We would like to implement a similar algorithm where a subgoal will be a roadmap node found in the path (starting at the center of the flock). Until the subgoal is reached, the shepherd will move the flock toward that goal and then will choose the next roadmap node on the path as the next subgoal

To move the flock toward the subgoal, the dog steers the flock from the rear (see Figure 6, second and third columns, the robot is behind the flock's covering circle). If any subgroup separates from the flock, it is the dog's job to move the subgroup back to the flock (third column

in Figure 6).

Algorithm IV SHEPHERDING (for dog)

```
01. Find a path on roadmap
02. while (goal not reached )
03.    Select the next node on the path as subgoal
04.    while (subgoal not reached)
05.       Move to rear of flock on the far side of subgoal
06.       if flock separates
07.          Move the subgroup that is farthest from
               subgoal toward other subgroups
08.       endif
09.    endwhile
10. endwhile
```

The script to implement shepherding behavior is shown below. Note that it should be applied everywhere in the environment, and hence every roadmap node should have this rule.

```
//Shepherding Script
AGENTVAR steeringLocation;
AGENTVARVAR dogAvoidance;
TRIGGER continuous:  findSteering;
TRIGGER continuous:  avoidDog;
  IF (agent.role==dog )
    agent.goal=steeringLocation;
  ELSE
    agent.externalForce=dogAvoidance;
```

The *findSteering* TRIGGER function continuously finds a steering location for the dog by checking the location of the sheep. The *avoidDog* trigger continuously computes an extra directional vector to avoid the dog (*dogAvoidance*). The internal variable *externalForce* is added to the boid rules.

Experimental Results

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

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.

All of our experiments were run on a Linux system with Athlon 1.33 GHz processor and 256MB memory. While noting that our techniques could use any roadmap, our current implementation is based on the Medial-Axis Probablistic Roadmap Method, or MAPRM (Wilmarth, Amato, & Stiller 1999). MAPRM probabilistically generates a roadmap which is an approximation of the medial axis of the environment in two dimensional space.

Homing Behavior

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

The environment is a square with sides measuring 420 meters that contains six types of obstacles (see Figure 4(a)). A total of 301 obstacles are randomly placed in the environment. 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. The experiment involves 40 flock members, which are initially placed according to a Gaussian distribution around the center of the square environment. The simulation is updated every 100 ms.

For the grid-based A^* behavior, a bitmap of the environment of 914×914 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 roadmap-based 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.

Homing behavior: Basic v.s. Roadmap

METHOD	#flockmate reaching the goal
Basic	10
grid-based A^*	40
Roadmap-based	40

Table 1: This table shows the number of the 40 flock members that reach their home within 30 seconds using the basic flocking behavior, the grid-based A^* behavior, and the roadmap-based behavior.

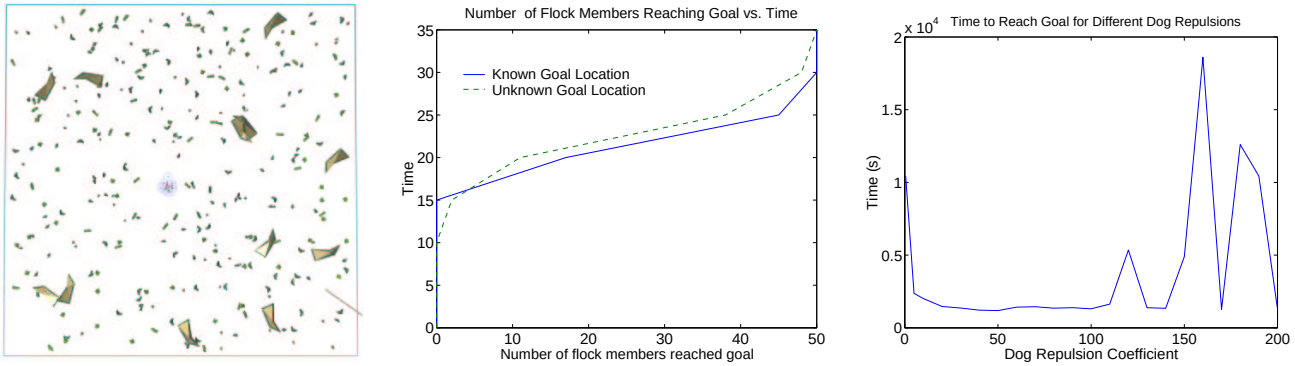
Table 1 shows that, without global information, only 25% of the flock members reach the goal and most of the others are trapped in local minima. On the other hand, when global navigation information is utilized, either with the grid-based A^* method or our roadmap-based method, all flock members reach the goal.

Homing behavior: Roadmap v.s. grid-based A^*

BEHAVIOR METHOD	init time	find path time	local minima #	local minima escape (s)
roadmap-based	0.88	0.652	255	22.99
grid-based A^*	6.02	5.757	2005	1035.43

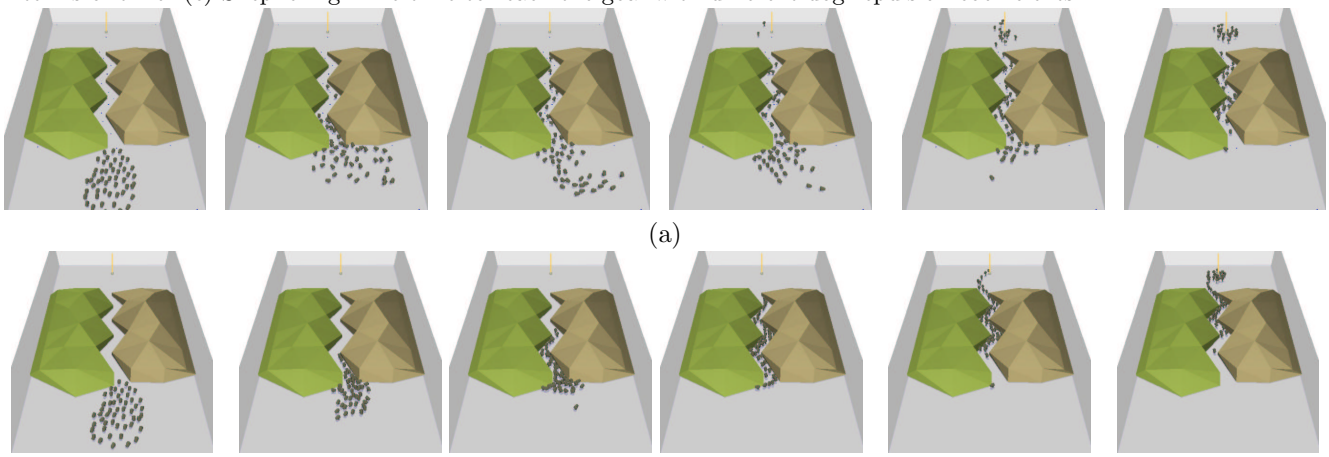
Table 2: This table shows the time for initialization, the average time to find a path, and the total time spent by all flockmates escaping local minima.

In Table 2 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. Note that our roadmap-based method is faster than the



(a) Homing (b) Goal searching (c) Shepherding

Figure 4: (a) Environment for homing experiments. (b) Goal Searching: the number of flock members reaching the goal area in terms of time. (c) Shephering: The time to reach the goal with different dog repulsion coefficients.



(a)

(b)

Figure 5: Passing through a narrow passage using (a) the follow-the-leader behavior, and (b) the naive strategy (homing to the exit node of the passage).

grid-based A^* method, mainly because it spends less time escaping from local minima.

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 (80×100) 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. 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. 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 4(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.

Narrow Passage Behavior

The narrow passage environment shown in Figure 5 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 which attempts to generate high clearance nodes on the medial axis of the free space. We consider the naive and the follow-the-leader strategies described earlier.

In the follow-the-leader strategy, a node is automatically assigned the narrow passage rule if its clearance is below some threshold. All other nodes are assigned homing rules 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 5(a). 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 naive strategy, 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. Snapshots of the agent movement are shown in Figure 5(b). Note how the agents clump together and interfere with each other as they move through the passage.

Shepherding Behavior

In shepherding, we try to compare the roadmap-based method with the grid-based A^* search and we investigate how the magnitude of the dog's repulsive force affects the simulation. We conducted two sets of experiments.

In the first set of experiments, the flock consists of 30 sheep and the sheep dog is an external agent. The experiment starts the flock in a random location and the objective is to move it to a randomly selected goal. When the sheep reach the goal, the experiment is repeated again by selecting a new starting position and goal position at random.

We compare our method to a grid-based A^* method. The basic method using local information was not considered due to its observed inadequacies in the previous experiments. The environment is the same and the experiments are similar as for the homing experiments (Figure 4(a)). In our grid-based A^* implementation, the search for the path to the goal and the dog's path to the steering position use A^* search on a bitmap with 914×914 cells. The roadmap method used the MAPRM method with 400 roadmap nodes, with each node connected to its 4 nearest neighbors.

Our results, shown in Table 3, include initialization times, the number of simulation steps required to reach the goal, and the number of local minima encountered. All values reported are averages over 40 experiments. We see that fewer local minima are encountered here than in the previous behaviors. This is influenced by the fact that MAPRM tries to generate paths that have high clearance from obstacles and because as the dog moves to the steering position it influences the individual members and increases the entropy of the system, resulting in relatively more randomness. Thus, even though some members are stuck in local minima, the dog would come and retrieve them. The table shows that the roadmap-based shepherding behavior performed better than the grid-based A^* search.

Shepherding behavior: Roadmap v.s. A^*

METHOD	init (s)	#steps	#local min.
Roadmap-based	0.88	2348.17	7.8
A^*-based	6.02	10612.08	32.2

Table 3: Shepherding behavior. This table shows time for initialization, the average number of simulation steps required to reach the goal, and the average number of local minima encountered.

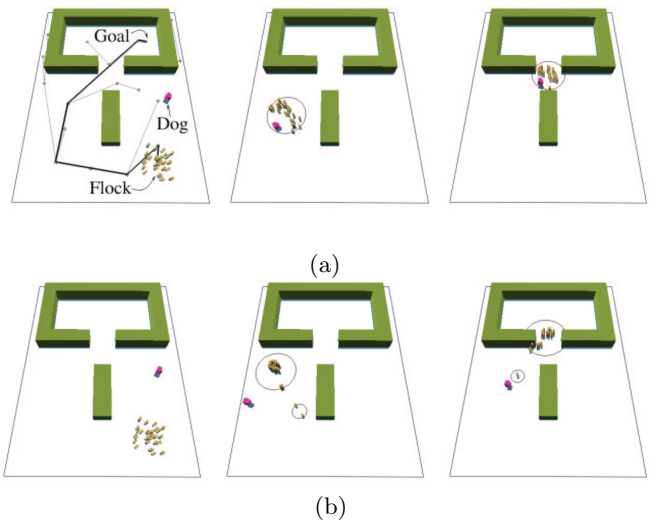


Figure 6: Shepherding behavior snapshots where the dog's repulsive force is (a) lower or (b) higher. Note that as the repulsive force increases, the flock may separate and become harder to herd.

In the second set of experiments, we have varied the repulsive force coefficient of the dog between 0.1 and 200. The coefficient represents the degree to which the sheep will be repulsed from the dog. Note that, although the greater the coefficient the more repulsion occurs, it does not mean that the total force on the sheep is proportional to the coefficient since there are other forces on the individual sheep (to represent boid behavior). Snapshots of this experiment in a simplified version of the environment can be seen in Figure 6, and Figure 4(c) shows the relation between the magnitude of the dog's repulsive force and the time needed to herd the flock to the goal. If the repulsive force is very low, it takes longer for the dog to push the sheep to the goal. As the repulsive force increases, the dog controls the sheep better.

However, after some point, the system becomes chaotic. This behavior can be explained by the fact that as the sheep become more repulsed, it becomes difficult for the dog to collect them into the herd. This can be seen in the second and third columns of Figure 6(b) where some group members are separated from the flock.

Conclusion

In this paper, we have shown that complex group behaviors can be generated using a roadmap providing global environment information. 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 thier 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 four types of behaviors studied show that the performance of the rule-based roadmap behaviors is very close to ideal behaviors that have complete knowledge.

In summary, we believe the techniques presented in this paper can be applied easily and efficiently to many diverse domains and have the potential to improve upon existing approaches.

References

Amato, N. M.; Bayazit, O. B.; Dale, L. K.; Jones, C. V.; and Vallejo, D. 1998. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 155–168.

Balch, T., and Hybinette, M. 2000. Social potentials for scalable multirobot formations. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 73–80.

Brogan, D. C., and Hodgins, J. K. 1997. Group behaviors for systems with significant dynamics. In *Autonomous Robots*, 137–153.

Fukuda, T.; Mizoguchi, H.; Sekiyama, K.; and Arai, F. 1999. Group behavior control for MARS (micro autonomous robotic system). In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1550–1555.

Funge, J.; Tu, X.; and Terzopoulos, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for interlligent characters. In *Computer Graphics*, 29–38.

Kavraki, L.; Svestka, P.; Latombe, J. C.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.* 12(4):566–580.

Khatib, O. 1986. Real–time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* 5(1):90–98.

Latombe, J. C. 1991. *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers.

Li, T. Y.; Jeng, Y. J.; and Chang, S. I. 2001. Simulating virtual human crowds with a leader-follower model. In *Proceedings of 2001 Computer Animation Conference*.

M., D.; Caro, G. D.; and Gambardella, L. M. 1999. Ant algorithms for discrete optimization. In *Artificial Life*, 137–172.

Mataric, M. J. 1994. *Interaction and Intelligent Behavior*. Ph.D. Dissertation, MIT EECS.

Nishimura, S., and Ikegami, T. 1997. Emergence of collective strategies in prey-predator game model. *Artif. Life* 3:243–260.

Parker, L. E. 1993. Designing control laws for cooperative agent teams. In *IEEE International Conference on Robotics and Automation*, 582–587.

Reynolds, C. W. 1987. Flocks, herds, and schools: A distributed behaviroal model. In *Computer Graphics*, 25–34.

Russell, S., and Norvig, P. 1994. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1th edition.

Saiwaki, N.; Komatsu, T.; Yoshida, T.; and Nishida, S. 1997. Automatic generation of moving crowd using chaos model. In *IEEE Int. Conference on System, Man and Cybernetics*, 3715–3721.

Sun, S.-J., and Sim, D.-W. L. K.-B. 2001. Artificial immune-based swarm behaviors of distributed autonomous robotic systems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 3993–3998.

Tu, X., and Terzopoulos, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Computer Graphics*, 24–29.

Vaughan, R. T.; Sumpter, N.; Henderson, J.; Frost, A.; and Cameron, S. 2000. Experiments in automatic flock control. *J. Robot. and Autonom. Sys.* 31:109–117.

Ward, C.; Gobet, F.; and Kendall, G. 2001. Evolving collective behavior in an artificial ecology. *Artif. Life* 7:191–209.

Wilmarth, S. A.; Amato, N. M.; and Stiller, P. F. 1999. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1024–1031.