Composable Group Behaviors [*]

Jyh-Ming Lien[†] neilien@cs.tamu.edu
Samuel Rodríguez[†] sor8786@cs.tamu.edu
Xinyu Tang[†] xinyut@cs.tamu.edu

John Maffei[†] jmaffei@cs.tamu.edu
Daniel Corlette[†] corletted@tamu.edu
Arnaud Masciotra [‡] masciotra@imerir.com
Nancy M. Amato[†] amato@cs.tamu.edu

Technical Report TR05-006
Parasol Lab.
Department of Computer Science
Texas A&M University

September 21, 2005

Abstract

Creating complex and realistic group behaviors can be a difficult and time consuming task. Automated approaches for motion generation typically involve explicitly defining a set of possible agent behaviors, associating appropriate behaviors with all environmental events, and setting the priorities among various behaviors in every possible situation. Generally, such approaches are pre-tuned to particular situations and are difficult to adapt for other scenarios or for different sets of behaviors. In this paper, we investigate methods to facilitate the generation of complex group behaviors for applications such as games, virtual reality, robotics and biological/ecological simulation. Our general approach is to provide a framework that automatically combines simple composable behaviors into more complex behaviors. Adaptation to new environments and specialization for new tasks or new agent abilities is supported by a "learning" process through which agents select their current behavior based on their prior experiences. We illustrate how our framework can be applied to pursuit/evasion and laser tag scenarios.

[*]This research supported in part by NSF Grants ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACR-0081510, ACR-0113971, CCR-0113974, EIA-9810937, EIA-0079874. Masciotra's work performed at the Parasol Lab at Texas A&M during a research internship in summer 2004.

[†]Parasol Lab., Dept. of Computer Science, Texas A&M University.

[‡]Mediterranean Institute of Studies and Research in Computing and Robotics (IMERIR), University of Perpignan, France.

1 Introduction

Simulating the coordinated behavior of multiple agents has been studied in many fields including robotics, computer animation and games. However, creating complex and realistic behaviors for a group of interacting agents remains a difficult and time consuming task. Most automated approaches for this task define a set of basic behaviors, and then set up an exhaustive prioritization among the behaviors that is used to determine which behavior(s) should be active at any given time [4,16,25]. Unfortunately, this level of specification results in architectures that can only be used for a specific workspace and task with a fixed set of behaviors. Moving to another workspace, requiring the agents to perform different tasks, and adding, removing or replacing any of the existing behaviors all takes significant effort, often requiring the redesign of the entire system.

In this research, we investigate techniques for easing the process of generating realistic and complex group behaviors. We provide a "learning" framework that allows these behaviors to automatically adapt to new environments and to new tasks. Our strategy is to enable the composition of a set of simple and basic user defined behaviors into a much larger set of complicated and more interesting behaviors. More specifically, we design our agents so that during the simulation they select their own behaviors to optimize a task-specific utility function based on their current status and their own prior experiences.

The idea of composing basic behaviors into complex behaviors is not new. In fact, combining simple behaviors is one of the most natural and general ways to generate realistic behaviors. Research on biology and animal behaviors has shown that the concept of basic behaviors exists ubiquitously ranging from lower level basic biomechanic motor controls [21] to high level ethological basis behaviors [9]. Composing simple behaviors into complex behaviors is also proposed in studies developing behavior based robotics [4]. In computer graphics, flocking behavior, which is generally composed of three simple rules (avoidance, alignment and coherence), is also a type of composed behavior [25].

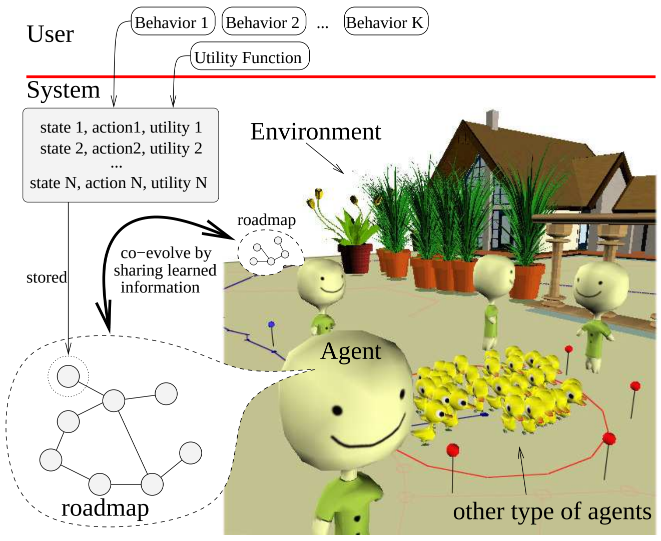

User
Behavior 1
Behavior 2
...
Behavior K
Utility Function
System
state 1, action1, utility 1
state 2, action2, utility 2
...
state N, action N, utility N
Environment
roadmap
co−evolve by sharing learned information
stored
Agent
roadmap
other type of agents

Figure 1: System overview.

While current technology in game and animation development can produce complex group behaviors, the process of creating such a system can take a significant amount of time and the resulting system is usually restricted. More flexible methods, in which agents can adapt to their environments and behaviors, for simulating group behaviors have been developed in the robotics community. However, these methods are generally designed to handle a small number of robots and usually cannot handle the complex scenarios we study. Different from these methods, our framework can simulate a large variety of agents which can have complex and adaptive behaviors.

Main features of our framework for generating group behaviors include:

- Users only need to provide a set of simple behaviors and the system will compose them into complex behaviors.

- Users do not need to define the relationships between each behavior, i.e., these simple behaviors can be developed independently without knowing what other behaviors are in the system.
- The relationships between simple behaviors, and the complex behaviors themselves, can evolve as the agents learn from their past experiences.

We demonstrate the utility, flexibility and power of our system by varying the allowable behaviors and measuring the performance of the system on multiple applications, including pursuit/evasion scenarios and laser tag.

2 Preliminaries

In this section, we define terms and concepts used in this paper. A *flock* is a collection of agents. Each agent can only sense a limited surrounding range, follows some basic rules [25], and creates control forces to change its state. Additional user-defined behaviors will be imposed on agents dynamically during the simulation.

A *roadmap* is an abstract representation of the reachable space in a given environment. A roadmap can be created manually or automatically using one of the existing roadmap-based motion planning methods [2, 18]. There are two types, global and local, of roadmaps in our system. A *global* roadmap is accessible to all agents. Thus, a global roadmap can be used as an indirect communication medium as shown in Figure 2. An agent can also have its own *local* copy of the roadmap. This local roadmap may be used to store specific information for this agent, such as behavior performances.

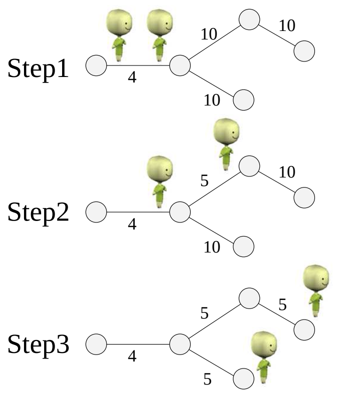


Figure 2: Flocking behavior using a global roadmap. As the agents explore the roadmap, weights are updated along edges that are traversed, e.g., between Steps 1 and 2 and between Steps 2 and 3.

A *state* of the simulation is a snapshot of all variables in the system, such as the configurations and velocities of all agents and the weights of the roadmap edges. Because each agent can sense locally, it can only obtain partial information about the current state, such as the current location of the agent itself, the location of neighboring agents and the surrounding obstacles. Even though an agent can have an entire roadmap, the agent has only restricted access to the surrounding roadmap nodes and edges. This restriction promotes realistic agent behaviors.

3 Related Work

Reynolds' influential flocking simulation [25] showed that flocking is a dramatic example of *emergent behavior* – complex behavior arising from the interaction of simple local rules and established the feasibility of modeling such a system. Each individual member of the flock has a simple rule set stating how it should move with its neighbors. This concept has been used successfully by researchers both in computer graphics [3, 7, 12, 29] and robotics [17, 22, 27, 31].

Animation. Frameworks for composing easier to design sub-modules into more complex systems can be found in recent studies on generating realistic animations. With the objective of creating realistic whole body motion for an articulated character, Faloutsos et al. [11] proposed a system that composes a set of primitive motion controllers to produce proper forces and torques for balancing, standing up, running and walking motions. Because a good motion controller is hard to design, such a system can allow users and designers to share and reuse motion controllers. In simulating group behaviors, Sung et al. [28] proposed a crowd simulation system that allows the agents in the system to select their behaviors online. A behavior is selected based on how likely the behavior can be performed given the agent's current state. If there are multiple behaviors available at the same time, the agent will pick one behavior at random. Because their goal is to generate animation for non-task-specific motions, it is the overall motion of the crowd that is important and the sequences of motions that an individual agent takes can be less realistic. In their system, transitioning between behaviors needs to be setup manually.

Robotics and Artificial life. Several architectures have been proposed in robotics to define the behaviors of a robot. Two of the most popular approaches (for "reactive" robots) are subsumption [8] and motor schema [4]. The subsumption architectures arrange behaviors in a hierarchical way, i.e., higher level behaviors can suppress and inhibit lower level behaviors. Motor schema architectures activate behaviors according to sensory inputs and add all forces generated from activated behaviors to control the robot. Both types of architectures have strengths and weakness. Several hybrid systems that attempt to draw from both the subsumption and motor schema architectures have been proposed, such as Mataric's concept of "basis behaviors" [19].

Behavior simulation research has also been reported for applications such as entertainment robots using ethograms [1] and simulating visibility based pursuit and evasion behaviors [26]. Pursuit and evasion is a common behavior seen in nature and robotics. It involves one group of agents that chases another group of agents. A simplified version of this behavior that considers only one pursuer and one evader has been studied for decades using methods such as game theory [15], genetic algorithms [24], and neural networks [10] (see [20] for a good survey).

While agents in these frameworks can be highly adaptive, these methods have usually considered only a small number of robots and have not been used for simulating complex behaviors. In contrast, our system scales to large number of agents and can seamlessly compose user defined basic behaviors into a much larger set of complicated and more interesting behaviors.

Computer games. Game developers have been able to create complex individual and group behaviors for characters in games. In many games, the complex behaviors set up for the game framework are designed for combat purposes and often involve meta-behaviors such as searching, combat, self-preservation or flight [16]. These meta-behaviors are usually linked in a simple finite state machine for easy transitions between behaviors and behaviors are selected from predefined behavior trees. Adding new behaviors requires updating this behavior structure. Though the behaviors can look realistic, adding more behaviors requires changes to the behavior structure and predefined behaviors often end up looking predictable.

In contrast, our framework supports the addition of basic behaviors without updating any behavior structure. Also, many of the behaviors we include in our framework involve basic behaviors between groups of agents which end up looking like more complex behaviors.

Communication. Group behavior simulations can also be viewed as multi-agent systems comprised of agents forming teams. Here the concept of a team is defined as a group of agents working together to maximize a commonly shared utility function. Many frameworks have been proposed to facilitate team-work in multi-agent systems within a belief-desire-intentions (BDI) framework [23]. These systems differ from the type of system proposed in this paper in that they allow agents to hold and reason with abstract notions of the world in which they are situated. However they do share one important trait with the type of framework proposed here, the need for agents to communicate with each other. Communication, either explicit or implicit, is central to groups of agents showing collective intention or collaborative group behavior [13].

The kind of communication that exists between agents in the framework described in this paper is implicit in that agents modify their behavior based on the proximity and location of neighboring agents. The ability for agents to communicate state information exists, however it is not central to the framework. Within the framework proposed, communication links or connections exist and information is transferred when agents fall within the visibility of other agents.

Connectivity in natural systems. The ability for agents within a system to communicate can be viewed as those agents sharing a connection. The study of natural systems that develop through a competitive evolutionary process can be revealing in regards to what might constitute a successful connectivity structure for systems wishing to simulate or exhibit properties of natural life. It has been shown that many naturally occurring complex systems

Table 1: System of the composable group behaviors. Notation is adapted from [30].

Name	Type	Description	Details
A: Agents	an index set	K agents in the system	see Section 4.1
T: Time Steps	an index set	time steps of the game (simulation)	
X: System States	a set	all possible states in the system. A state of the game at time n is denoted as x_n	see Section 4.2
B_n^a: Composable Behavior	a set	behaviors available for the a-th agent at time n	see Section 4.3
S_n^a: Agent State	a set	sensor state available for the a-th agent at time n	see Section 4.2
I_n^a: Learned Experiences	a set	actions and observations from any previous time step	see Section 4.6
Γ_n^a: Game Strategy	mappings $\gamma_n^a : I_n^a \to B_n^a$	the strategies available to the a-th agent at time n	See Section 4.5
U^a: Game Utility	a function	a utility function of the a-th agent $U^a : (X \times B_1^1 \times B_1^2 \times \cdots \times B_1^K)\times$ $(X \times B_2^1 \times B_2^2 \times \cdots \times B_2^K) \times \cdots \times$ $(X \times B_N^1 \times B_N^2 \times \cdots \times B_N^K) \to \mathbb{R}$	see Section 4.4

from differing domains show a similar connectivity structure. Barabasi et.al. showed that many social networks, biological networks, and man made networks share the property that their underlying network structure is "scale-free" [5]. For most of the twentieth century most complex systems modeled in graph theory were represented by an exponential distribution. Some of the systems which Barabasi found to contain this scale-free structure include cellular metabolism, protein regulatory networks, sexual relationships, actors database (IMDB) for Co-Stars, research collaborations (papers citations), internet (routers), and the World Wide Web (Web links) [5].

4 The framework

Our system is built on top of a *roadmap* based system. As shown in Figure 1, the proposed framework consists of several main components, i.e., agents, roadmaps, user defined behaviors and utility functions. In this work, an agent can store agent-specific information, such as the action that the agent takes, in a node of an agent owned local roadmap and store information in a global roadmap when inter-agent communication is needed. The result of an action will be evaluated by a user defined utility function. The agent will select an action based on the past performance of the action. The learned information can also be shared between agents when their relative states allow communication (e.g., proximity and other values).

Although a straight forward application of game theory is not able to handle complex behaviors at the levels that we are interested in for this paper, it provides a good framework for studying our problems. In Table 1, we use notation borrowed from game-theory-based motion planning [30] to assist our discussion in the remainder of this paper.

Similar to a game, the goal of an agent (a game player) studied in this paper is to generate a series of behaviors (actions) so that the outcome state(s) of the agent will maximize its utility function. The plan for generating the behaviors to achieve the goal is called the "game strategy" of the agent. Instead of solving differential equations, which quickly become too complex to handle even for a game with a small number of players [15], our system "solves" the optimization problem by allowing agents to learn via the provided utility function and to learn from other agents' experiences. This is similar to the concept of co-evolution of agents studied in [10, 24]. The major components of our system are summarized in Table 1, and are described in more detail below.

4.1 $A = \{a_k\}$: agents

An agent in the system is a member of a flock and therefore is equipped with basic flocking dynamics. In addition to these basic behaviors, an agent can perform behaviors defined by users. An agent can also access and modify information stored in global and local roadmaps. The global roadmap is accessible to all agents and can be used as an indirect medium to communicate with other agents. A local roadmap is used to store the (private) knowledge and memory of an agent.

4.2 S_n^a: system and agent states

A state $s \in S_n^a$ for an agent a at time n consists of a nearest roadmap node to the agent and a list of neighboring agents. All states of an agent, with the behaviors performed under these states and their associated utilities, are stored in the associated local roadmap nodes. Thus, a state in our system is used as an index to retrieve information about an agent's prior experiences.

4.3 B_n^a: composable simple behaviors

A behavior is a set of rules that determines what the next desired location and direction of the agent should be and produces control forces to move the agent there. To make a (simple) behavior composable in our system, the only requirement is to provide a function for the system that determines if the behavior can be applied to a given state of a flock. For example, a "running away from opponent" behavior is only applicable to an agent if there are opponents in its vicinity or an "approach flock" behavior can only be applied if a flock is in view.

Note that, unlike most other systems [4, 16, 25], our system does not require users to explicitly specify the relationships, e.g., a dependency graph between behaviors specifying relationships such as behavior $\mathcal{B}_2$ can only be performed after behavior $\mathcal{B}_1$. The "applicability" function associated with each behavior implicitly provides this information so adding new behaviors to our system will not change the system itself and other existing behaviors in the system.

4.4 U^a (game utility): evaluating the result of an action

Each available action (behavior) at a state is associated with a utility value. A higher utility is correlated with a higher probability that the behavior will be selected subsequently. Initially, each action has the same utility and therefore has equal probability of being selected. This utility value evolves during the simulation. Intuitively, the more successful a behavior has been in the past, the higher utility value it should have.

Every time the agent performs an action (a behavior at a state), a utility function (sometimes called the loss function) is used to evaluate the performance of this action. The utility function is provided by the user and depends on the goal of the simulation. For example, in a game of tag, this function might evaluate the duration of an agent being *it*, in a predator-prey simulation, this function may compute the distances between predators and prey and evaluate how many times an agent has captured or been captured, or in the shepherding behaviors, this function might evaluate how many times the controlled flock is separated into sub-groups. More detail regarding these utility functions will be discussed in the experiments in Section 5.

In our case, the utility associated with each behavior is learned during the simulation. We note that utility can also be provided by the user or determined from the previously learned utility. For example, for applications like games in which directability is important, we can allow the actions to be very predictable by assigning very high utilities to certain behaviors. For applications such as robotics in which adaptability is important, we can interject unpredictability in the simulation by assigning similar evaluation and rewards for behaviors and randomly selecting behaviors based on the cumulative utilities throughout the simulation.

4.5 Γ_n^a (game strategy): mapping from a state to an action

An agent determines its next action by selecting one or more of the allowable behaviors from a behavior list. In our current implementation, this action selection process is done by picking one of the available behaviors at random according to their utilities accumulated in the past time steps. Thus, an action with higher utility for the current state has higher probability of being chosen.

Each time an agent needs to select its behavior, it first finds the state in its roadmap nodes that is similar to its current state. After the state is found, the agent selects the next behavior from a list of performed behaviors associated with that state (which have been evaluated by their outcomes) and from a list of available behaviors at the current state that have not been performed before (which still have the initial utility); see Figure 3.

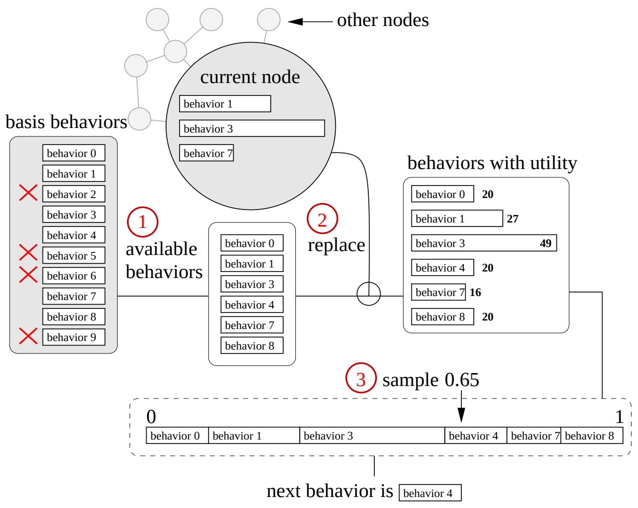


Figure 3: The process of selecting the next behavior. Three steps: (1) Determine available behaviors for a given state. (2) Replace the utilities of available behaviors with the utilities of performed behaviors. (3) Randomly pick one of the behaviors based on their normalized utilities.

4.6 Co-Evolution Using Roadmap

All information obtained by the agent is stored in a roadmap. In the current implementation, each agent can have its own copy of the roadmap. We can say that a local roadmap is a learned knowledge base that contains a list of learned experiences during the simulation. This knowledge base can be shared between any two agents by merging information stored in the roadmap nodes when these two agents are in states that allow them to communicate.

Note that, although we allow the agent to have a complete copy of the roadmap, our framework does not require an agent to have complete knowledge of the roadmap. In fact, a complete roadmap is generally not necessary and may not be a realistic assumption. This constraint can be easily relaxed by allowing the agents to have the explored portion of the roadmap as in our previous work [6].

4.7 Putting it All Together

At the beginning of a simulation, a global roadmap is automatically created according to the description of the environment or loaded into the system from a file. This global roadmap is replicated and is provided as a local roadmap to each agent. The roadmap nodes, at this point, do not have any stored information and all behaviors are assigned a default utility value.

At each time step, an agent will check if its current behavior is still applicable to the current state. If so, the agent will continue to exhibit the current behavior. Otherwise, the current behavior is evaluated (by the utility functions provided by users) and a new behavior is chosen. Selecting an applicable behavior involves the following steps:

1. Match the current state to a state s in the local roadmap. If no such state exists in the roadmap, let the current state be s and insert s into the roadmap. Two states are a match if both states have the same roadmap node and the agent in both states can see a similar number of neighboring flock members.
2. Add the current behavior and its associated utility to s.
3. Retrieve the behaviors $\mathcal{B}_1$ and their utilities (from the past) that are associated with s.
4. Find applicable behaviors $\mathcal{B}_2$ that have not been performed in the past, i.e., behaviors that are not in s.
5. Replace the current behavior by randomly picking a behavior from $\mathcal{B}_1 \cup \mathcal{B}_2$ according to their utilities.

This process is illustrated in Figure 4.

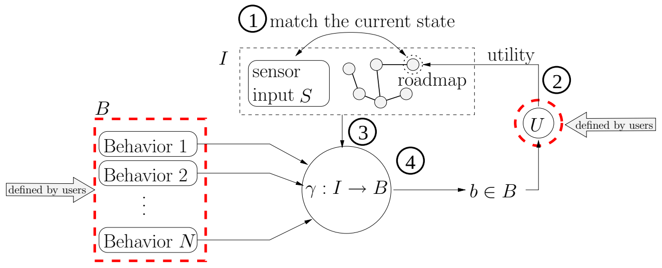


Figure 4: The process of changing to the next behavior. I is the actions and observations of current and all previous time steps which are stored in roadmap nodes. U is the utility function provided by users. The function γ is shown in Figure 3. This process is discussed in detail in Section 4.7.

5 Experiments

We studied two types of behaviors: a general pursuit/evasion behavior and a pursuit/evasion behavior in a laser tag game. In each behavior, we discuss the primitive behaviors and the associated utility function. We also compare the performance of agents with different types behaviors and show that the agents can learn to adapt to the environment and learn to adapt to their constraints by selecting the behavior that maximizes the utility under the given circumstances. Movies illustrating the behaviors are available on the project website.

5.1 Pursuit/Evasion Behaviors

Table 2: Basic Behaviors for Pursuit and Evasion.

Type	Name	Applicable if
Evasion	in a straight line	there are predators in sight
	zigzag	
	circle	
	random walk	
Hiding	behind a near-by obstacle	there are predators and obstacles in sight
	behind a same color obstacle	
Waiting	at the current node	there is no predator or prey in sight
	patrolling between two nodes	
Pursuing	in a straight line	there is prey in sight
	by predicting prey's direction	
Searching	random walk	there is no predator or prey in sight
	least visited areas	
Following	a friend	there are friends in sight

Pursuit and evasion are common behaviors which are frequently found in nature. Computational models of pursuit and evasion have been studied for more than five decades [14]. Typically, in these behaviors, a group of agents (the predators) attempt to catch one or more members of another group of agents (the prey) while the prey try to avoid the predators. Table 2 lists a more specific set of basic behaviors for the pursuit and evasion behaviors studied in this paper. These behaviors are given to the simulated agents and will be composed via the framework discussed in the previous sections.

The framework can incorporate any behavior that can determine if it is applicable in a given state. Our implementation of all of the basic behaviors listed in Table 2 is made up by combining and reusing a set of more primitive behaviors. For example, as shown in Figure 5, the Pursuing behaviors are a combination of the Locomotion (a straight line, zigzag or random walk) and Towards Target primitives, the Waiting behaviors are formed by a series of Locomotion and Stop primitives, and the Evasion behaviors are a combination of the Locomotion and Away from Target primitives.

In a specific situation, only a subset of basic behaviors will be applicable to a flock agent. For example, the

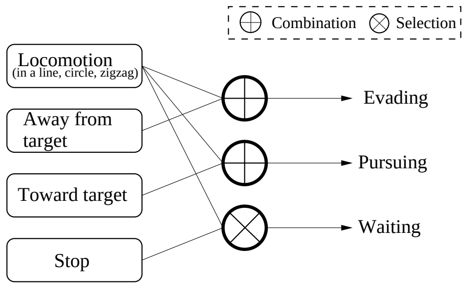


Figure 5: The basic behaviors are made up from a set of primitive behaviors.

flock agent will pursue only when it sees prey (which is in his food chain); the flock agent will be waiting only when it does not see a predator. Below is a more detailed explanation of the basic behaviors.

- **Pursuit Behavior.** Whenever a flock agent sees prey, it will attempt to pursue and attack it. By knowing the prey's current position at each time step, we compute the force that attracts the predator towards the prey. The predator can adopt any strategy (including straight line or target prediction) to approach the prey. It will continue chasing until it either reaches the prey or it can no longer see the prey and gives up.
- **Evasion Behavior.** Running Away is a simple way to avoid a predator. Whenever the agent sees one or more predators, it will start to evade. A repulsive force is computed from each visible predator. The agent can select any locomotion strategy (straight line, zigzag, random walk, or a circular movement) to try to move to a safe area without being caught by the predator.
- **Hiding Behavior.** In the hiding behavior the prey avoids a predator by hiding behind an obstacle. Similar to the running away behavior, the prey knows the position of the closest hunter. Instead of calculating a repulsive force, the prey determines the closest obstacle to the prey and computes a position next to this obstacle that is not visible to the predator. We then compute a path for the prey to go to this position and wait there.
- **Waiting Behavior.** The waiting behavior can be applied only when a flock agent does not see any opposing agents (either predators or prey). It will stay within a small area (near one or several roadmap nodes).
- **Searching Behavior.** This behavior is used when the flock member doesn't know the location of a target. It will explore the area until a target is found. This searching behavior can either choose to explore unvisited areas or random places in the environment.
- **Following Behavior.** This behavior is similar to pursuit. However, instead of chasing a prey, it will follow a friendly agent (which is neither its prey nor predator). This behavior can be used to gather flock members into a group (e.g., for group attacking).

Utility function. The utility function of the pursuit and evasion behavior is designed to encourage the agent to catch prey and to avoid being caught by the predators. It simply counts the number of times an agent has caught and has been caught by other agents. When an agent discovers or catches (or is caught by) a prey (a predator), the behavior that leads the agent to this state will be rewarded with a positive (resp., negative) utility.

Experimental setup. In this experiment, we show that an agent's performance can improve dramatically by simply adding more behaviors to it. Figure 6 shows the environment for this experiment. In this environment, there are two groups of flocking agents, predators and their prey. The prey are designed to only have the patrolling and evasion behaviors and normally walk back and forth along the patrolling path and start running away when predators are nearby. The predators can choose to either wait or search for prey in the entire environment if they do not see any. When they see a prey, they will start pursuing and then attacking it. Both predators and prey have 100 life points at the beginning. The life points of predators decrease as time goes by and increase after catching a prey. The prey can run four times faster and see twice as far as the predator so that the prey become hard to catch.

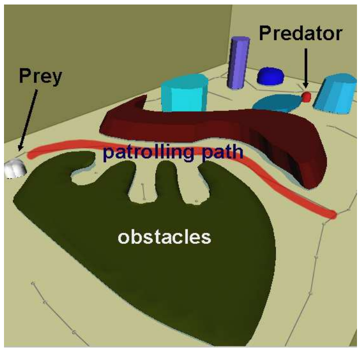
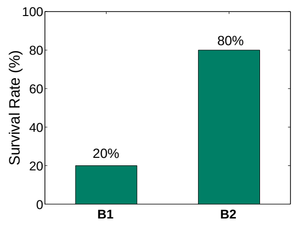
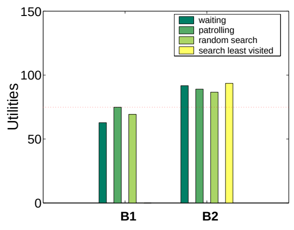


Figure 6: Environment for pursuit and evasion experiments.

We study how well a predator performs with different behaviors. In the end of each experiment, the predator will either die or capture all the prey. Thus the performance of the predator is measured as the number of times the predator survives. In this environment, the predator will need to search for prey and stay close to the patrolling path of the prey in order to capture all the prey (because the prey can quickly run out of the sensory range of the predator). Figure 7(a) shows that the survival rate of the predator increases four times from 20% to 80% when the predator has the ability to search least visited areas. Note that the utilities of all behaviors of the predator increase after the predator can search least visited areas. See Figure 7(b). As the utility of a behavior positively correlates with the performance of the behavior, this increase in the utilities of all behaviors indicates that both searching for the prey and waiting in that narrow passage are necessary to capture all the prey.



(a)



(b)

Figure 7: Results for pursuit and evasion experiments. (a) Survival rate of the predator. (b) Utilities of the basic behaviors. B1 and B2 are two sets of basic behaviors for the predator. B1: Waiting, patrolling, random searching and pursuing. B2: All behaviors in B1 and also the searching least visited areas behavior.

5.2 Laser Tag

Table 3: Basic Behaviors for Pursuit and Evasion.

Type	**Name**	**Applicable if**
Attacking	shooting at opponents	there are opponents in sight

In the laser tag game, agents work as teams in order to score points against agents from opposing teams. An agent's goal is to shoot at opponents, with virtual laser shots, in order to score hits against the opponents. After a specified number of hits against an agent, that agent can no longer participate in the game.

In our simulation, the probability of hitting an opposing agent is determined by a probability distribution within the visible area as seen in Figure 8. This probability is determined by the distance an agent is from the prey as well as the angle between the heading direction and the direction to the prey. Although this is just a simple approximation of an actual shooting scenario, it is a straightforward way to approximate the likelihood of a successful shot, which could possibly be determined by many factors.

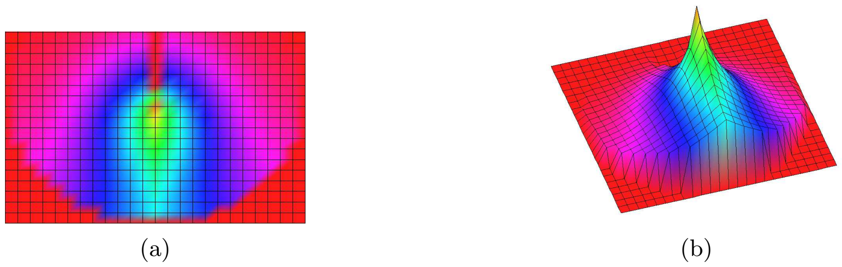


Figure 8: Shot distribution. (a) Green indicates a higher probability of being hit. (b) is a 3D projection of (a).

With the exception of the attacking behavior, the other basic behaviors available to the agents are the same as in the pursuit-evasion example in Table 2. The attacking behavior shown in Table 3 is different in that as soon as an agent sees an opposing agent, the agent can either shoot or position itself for a better, possibly more successful shot. Since the probability of a successful shot depends on the distance to the agent and the angle to that agent, it may be more beneficial for the shooter to close the distance to the opposing agent or line up with the opposing agent for a better chance of a successful shot.

Utility function. The utility function of the laser tag game is the same as that of the pursuit and evasion game.

Experimental setup. Our laser tag environment, shown in Figure 9, consists of two flock groups. Both groups have the ability to shoot at opposing agents. The flock groups have different properties making the simulation somewhat more realistic. The first flock group (called snipers) is equipped with a longer range of sight but a more restricted viewing angle. The viewing angle is the total angle in the heading direction that the flock member can see. The second flock group (called infantry) has a shorter range of sight but a much larger view angle.

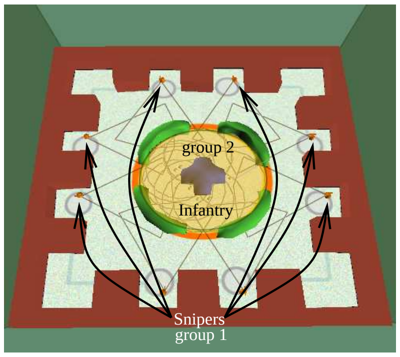
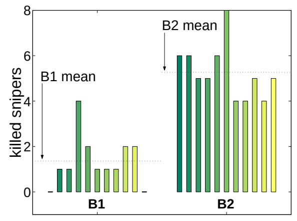


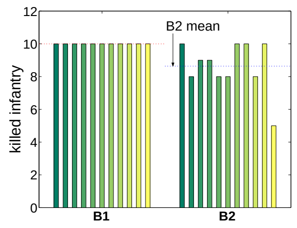
Figure 9: Environment for laser tag experiments.

The first flock (sniper) group is positioned along the circumference, initially surrounding the second flock group which starts at the center of the environment. The sniper group will only have two available behaviors, the attacking or shooting and waiting behaviors, meaning they will be stationed at their initial locations waiting for opposing agents. The second flock (infantry) group will be given a more complete set of behaviors along with the shooting behavior including searching the least visited areas or searching randomly and the evading behaviors.

Figure 10 shows results of this set of experiments. We compare the performance of the agents with and without the following behavior. Although the following behavior does not seem to have a direct relation to this laser tag game, it allows the agents to form a larger group. The advantages of forming a larger group include reducing the chance of getting shot and increasing the chance of eliminating sniper agents. This kind of formation is critical for this experimental setup in which the sniper agents can shoot the infantry agents long before the infantry agents can even see the sniper agents. We see that the agents learn this key factor from the results shown in Figure 10 in which the agents with the following behavior can kill significantly more snipers than the agents without the following behavior and they also have noticeably fewer losses of group members. Thus, we see that the agents learn to use the following behavior to increase the utility of the laser tag game.



(a)



(b)

Figure 10: Results for laser tag experiments. B1 and B2 are two sets of behaviors for the infantry group. Each bar in these plots shows the result of a single simulation run. B1: shooting and random searching. B2: All behaviors in B1 and additionally following neighboring agents.

6 Conclusion

In this work, we propose a framework to ease the process of creating natural and complex group behaviors. Users can use this framework to compose more complex behaviors from a set of simple behaviors. The agents in the system adapt to the given situation as shown in our experimental results. One benefit is that it provides a convenient medium for enabling different researchers to share independently developed behaviors.

There are several ways to extend the current implementation of the framework. A simple extension of the framework is to let the agent store and regain previously learnt experiences if the learning process is not critical to the applications, such as in an animation or in a game.

References

[1] K. Ali and R. Arkin. Implementing schema-theoretic methods of animal behavior in robotics systems. In *5th International Workshop on Advanced Motion Control*, pages 246–253, Coimbra, Portugal, 1998. ACM Press.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, pages 155–168, Natick, MA, 1998. A.K. Peters. Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[3] M. Anderson, E. McDaniel, and S. Chenney. Constrained animation of flocks. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 286–297. Eurographics Association, 2003.

[4] R. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)"*.

[5] A. Barabasi and E. Bonabeau. Scale-free networks. *Scientific American*, (288):60–69, 2003.

[6] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Better group behaviors in complex environments using global roadmaps. In *Artif. Life*, pages 362–370, Dec 2002.

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

[8] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Trans. Robot. Automat.*, RA-2(1):12–23, 1986.

[9] F. Cervantes-Perez. Schema theory as a common language to study sensori-motor coordination. In *Visuomotor Coordination: Amphibians, Comparisons, Models, and Robots*, pages 421–450, 1989.

[10] D. Cliff and G. F. Miller. Co-evolution of pursuit and evasion II: Simulation methods and results. In P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack, and S. W. Wilson, editors, *From animals to animats 4*, pages 506–515, Cambridge, MA, 1996. MIT Press.

[11] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 251–260, New York, NY, USA, 2001. ACM Press.

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

[13] B. Grosz and L. Hunsberger. The dynamics of intentions in collaborative intentionality.

[14] R. Isaacs. Games of pursuit. Technical Report P-257, Rand Corporation, 1951.

[15] R. Isaacs. *Differential games: A mathematical theory with applications to warfare and pursuit and optimization.* John Wiley, 1965.

[16] D. Isla. Managing complexity in the halo 2 ai system. In *Game Developers Conference*, 2005.

[17] T.-Y. Li and H.-C. Chou. Motion planning for a crowd of robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[18] J.-M. Lien, S. L. Thomas, and N. M. Amato. A general framework for sampling on the medial axis of the free space. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[19] M. J. Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):51–80, Dec 1995.

[20] G. Miller and D. Cliff. Co-evolution of pursuit and evasion i: Biological and game-theoretic foundations. Technical Report CSRP311, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1994.

[21] F. Mussa-Ivaldi and S. Giszter. Vector field approximation: a computational paradigm for motor control and learning. *Biol Cybern.*, 67(6):491–500, 1992.

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

[23] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[24] C. W. Reynolds. Competition, coevolution and the game of tag.

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

[26] S. Sachs, S. M. LaValle, and S. Rajko. Visibility-based pursuit-evasion in an unknown planar environment. *Int. J. Robot. Res.*, 23(1):3–26, 2004.

[27] A. C. Schultz, J. J. Grefenstette, and W. Adams. Robo-shepherd: Learning complex robotic behaviors. In *Proceedings of the International Symposium on Robotics and Automation*, pages 763–768, May 1996.

[28] M. Sung, M. Gleicher, and S. Chenney. Scalable behaviors for crowd simulation. In *Eurographics*, volume 23, 2004.

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

[30] S. L. Valle. Robot motion planning: A game-theoretic foundation. *Algorithmica*, 26(3):430–465, 2000.

[31] 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.