

Evolving Crowd Motion Simulation with Genetic Algorithm

Chih-Chien Wang

Computer Science Department
National Chengchi University
g9022@cs.nccu.edu.tw

Tsai-Yen Li

Computer Science Department
National Chengchi University
li@nccu.edu.tw

Abstract

The problem of generating realistic crowd motion for robot fleet in robotics or virtual crowd in computer animation has attracted much attention in recent years. Previous work has succeeded in using virtual forces to simulate the motions of virtual creatures, such as birds or fishes, in a crowd. However, how to set up the virtual forces in order to generate desired motion behaviours remains empirical. In this paper, we propose to use the genetic algorithm to generate an optimal set of weighting parameters for composing virtual forces according to the given environment and desired movement behaviour. We have designed a list of measurement functions for composing the fitness function used in the genetic algorithm. Experiments in simulation have been conducted for several environments and behaviours. The results show that compelling examples can be generated with the parameters automatically found in this approach.

Keywords: Crowd Motion Simulation, Genetic Algorithm, Robot Formation, Multiple Robot System

1 Introduction

Formation control for multi-robot systems is a classical robotic problem that has attracted much attention in the literature. In recent years, due to the booming of the entertainment industry, especially in computer animation, the techniques of simulating desirable behaviors of virtual crowd also has created many potential applications of great value such as virtual mall, digital entertainment, etc. The techniques have also been used in scientific studies such as simulating crowd movement for transportation or urban planning.

According to the application contexts, the motion of a crowd can be structured in many ways with various degrees of organization. For example, in most robotic applications, the goal of formation control usually is to form a specific shape by means of referencing methods [13]. However, it is more common in computer animation applications to ask for a less structured crowd movement as long as it looks realistic according to our everyday experience. For example, one may desire to simulate a moving crowd as a cohesive group with consistent orientation, or a crowd following their leader/goal as closely as possible. We call these desirable visual effects the *movement behaviors* of a crowd.

The research of crowd motion simulation has been active for many years. The types of agents involved in the simulation include a flock of bird, a school of fish, or a crowd of people. Among these agents, simulating a human crowd is the most challenging because the behavior complexity that human possesses. We will call these agents *robots* in the rest of this paper. Most of the current systems adopt a local control approach. One of the common approaches to this problem in computer animation adopts the virtual force model

[13], where the movement of each robot is affected by virtual forces computed according to its spatial relation with other robots or objects in the environment.

Although several compelling examples have been created in computer animation with this virtual-force technique, how to choose appropriate forces and their weights remains a state of art that usually requires tuning of the designer in order to achieve good simulation results. In addition, there is no objective way to evaluate the result. In this paper, we propose to model the problem of generating good crowd movement behaviors by designing appropriate parameterization and evaluation model and adopt the genetic algorithm [6][8][10] to search for an optimal set of parameters. The parameterized virtual-force model is evaluated by an array of measures to describe the desired crowd behavior. We have conducted experiments to generate parameter sets for several common crowd behaviors in various environments of different spatial structures. We believe that this system will be able to automate the time-consuming parameter-tuning process required for simulating a desired behavior in a given environment.

We organize the rest of the paper as follows. In Section 2, we survey the work pertaining to the techniques that have been used in this research. In Section 3, we will describe the virtual force model for each individual robot. We will then propose how to formulate the problem with an evolutionary approach in Section 4. In Section 5, we will describe how we conduct the experiments and show the simulation results obtained from these experiments. We will then conclude the paper with some future research directions at the end.

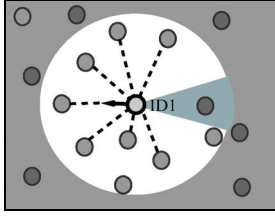


Figure 1: View range of a robot and its neighbours

2 Related Work

Crowd motions can be generated by simulation or by planning. Due to the complexity involved in such a system, simulation with distributed local computation is the most common approach. However, we can also find some work that attempts to plan the motion for a large crowd or use a hybrid approach in which only the motions of a few key robots (called leaders) are planned. We will first review the related work in simulation and then in planning.

2.1 Crowd Motion Simulation

The approaches to simulate crowd motion can be classified into three categories with different levels of abstraction: *particle system*, *flocking system*, and *behavioral system*. Bouvier[2], Brogan & Hodgins 0[4], used physical-based particle system to simulate a crowd of athletes such as runners or bikers competing in a field. Reynold [13][14] proposed to apply virtual forces to individual robots (called boids) to create steering behaviors for the whole crowd. Tu and Terzopoulos [5] used rule-based finite-state machine to construct the cognitive model of fish and succeeded in creating several interesting behaviors including flocking. Muse and Thalmann[11] used behavior rules to design virtual characters and used scripts to construct complex social interactions with various levels of control. They used the theory in social psychology to model crowd movement behaviors. In addition, Saiwaki, et al.[14] used the principle of chaos to design virtual crowd. However, great variations have been observed on the trajectory of the simulated crowd. It is also difficult to control the result of simulation precisely.

2.2 Motion Planning for Crowd

Several approaches have used the techniques of motion planning to generate motions for multiple robots. However, as the complexity of motion planning problem grows exponentially in the degrees of freedom that the system has, the problem of planning the motion for a crowd of robots with a systematic search dooms to be intractable. Therefore, most approaches use decoupled planning that generates crowd motions sequentially by planning only for one robot at a time. In addition, in order to reduce the complexity, only the motions of a few key robots are planned, and the other robots use local rules to follow the leaders. For

example, in [7], the authors have proposed a decoupled planning approach to generate the motions for leaders while taking the number of followers into account. The rest of the crowd follows the leaders with the virtual force model proposed in [13]. In contrast, the authors in [8] proposed a centralized approach to directly tackle the motion planning problem for a crowd of over 200 robots. However, since reaching goal is the only criterion, the crowd behavior in the course of the movement remains difficult to control.

3 Design of Movement Model and Virtual Forces

In this research, we have adopted the virtual force model proposed in [13] as the way to affect how each individual robot moves. We will first describe the model of movement and perception of a robot and then the virtual force model.

3.1 Design of Movement Model for an Individual Robot

We assume that the robots move under the influence of virtual forces proposed in [14], and they must respect some maximal speed limits in translation and rotation. In addition, a robot has a limited view range. In our system, a robot is given a view angle of 330 and a constant view distance that is 20 times of the size of the robot, as shown in Figure 1. Since the virtual forces are computed locally, only the robots or obstacles that are within the view range have effects on the computation of the virtual forces.

In the simulation system, each individual robot computes the virtual forces described in the next subsection according to its relative position to other robots or environmental obstacles in its view range. The computed virtual forces are used to update the next configuration of the robot. However, the new configuration is not guaranteed to be collision-free. In the case of being in-collision, the system uses a collision handling procedure that makes use of various local strategies to escape the collision situation.

3.2 Design of Virtual Forces

Virtual forces are used to drive the movement in our crowd simulation system. Five types of virtual forces have been used in this work: *separation*, *alignment*, *cohesion*, *following*, and *collision*. The first three are originally proposed in [13] and computed according to the objects in the view range while the other two were added in [7].

Separation force (F_{sep}): the repulsive force is computed proportionally according to the distance between this robot and other neighboring robots within the view range. The resultant force is the summation of all forces exerted by individual robots. The force is to maintain a safe distance between robots, and the

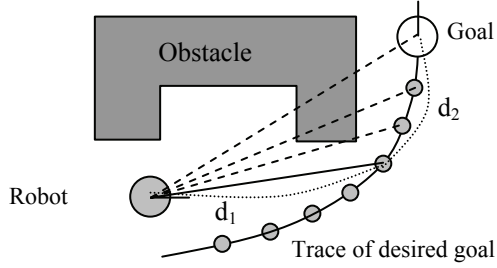


Figure 2: Following the trace of a desired goal

desirable strength should be different for different behavior requirements and in different environments.

Alignment force (F_{align}): A robot uses this force to align its velocity and orientation with other neighboring robots. An average velocity for the robots within the view range is computed first. The alignment force is then computed according to how the velocity of the current robot deviates from the average one.

Cohesion force (F_{coh}): The center of the robots within the view range is first computed according to their position vectors. Then the cohesion force is computed based on the vector between the center and the current position of the robot. This force has the effect of keeping the robots as a group.

Following force (F_{fol}): A moving crowd follows a specific goal, which may not be a physically existing leader. The following force is an attractive force that drives the crowd to its goal. This force is computed according to the distance from the goal, which could also be moving. However, since there could be obstacles in the environment, the vector connecting their positions may not reflect the correct direction and distance for the following action. We often find that robots can easily get stuck at some local cavity under the attractive following force. Therefore, we have adopted the model proposed in [7] to compute a collision-free following path by making the robot head to a point along the trace of the leader that does not cause collisions with obstacles, as illustrated in Fig. 2.

Collision force (F_{col}): The repulsive collision force is exerted by the environmental obstacles when a collision situation is predicted after certain period of time. The direction of this force is opposite to the moving direction of the robot, and this force takes effect to resolve collisions when the robot is close to the obstacle boundary.

Our system uses a linear combination of the normalized component forces as shown in Eq.(1).

$$F = s_1 * F_{sep} + s_2 * F_{align} + s_3 * F_{coh} + s_4 * F_{fol} + s_5 * F_{col} \quad (1)$$

The set of weights (s_1, s_2, s_3, s_4, s_5) determines how the forces are composed to affect the behaviour of the robot. How to set up the weights appropriately re-

mains empirical for the designer, which is the reason that we propose to automate the search process for an optimal solution with the genetic algorithm as described in the next section.

4 Evolution of Crowd Motion Behavior

In crowd simulation, the trajectory of a crowd may vary greatly according to the scenario and the environment where the crowd is situated. For example, in outdoor open space, people tend to keep a larger distance amongst others. In contrast, people may have to jam-pack into a narrow passage in order to reach their goal as early as possible. In the framework of using virtual forces to simulate crowd motions, the weights of the component forces are the parameters for the designer to tune in order to achieve the desirable results. The complexity of the parameter space and the time-consuming simulation process make it a difficult optimization problem for human or for machine.

We propose to solve the problem with the genetic algorithm, commonly used to solve the problem of searching for an optimal solution in a large search space [9][10]. In a genetic algorithm, the parameters that control the performance of a system are first encoded, and fitness functions for evaluating the solutions are provided by the designer of the system. By evaluating the population in a generation with the fitness function, good parameter sets are selected for continuing evolution in the subsequent generations. Evolution operations such as crossover and mutation are also often used in the evolution process to find global optimal solutions instead of the local optima. In the following subsection, we will describe in more details how we encode the crowd simulation problem and how we design the fitness functions for evaluating the crowd behaviors.

4.1 Problem Formulating with Genetic Algorithm

The set of weights described in the previous section are used as the genes for encoding. Each of genes is encoded into a bit string of length 10. Since we have five parameters (genes) in our system, the total length of the chromosome is 50 bits. In the current system, the population is set to 200. Each chromosome is tested in the simulation and evaluated with the fitness function. In this work, we have chosen the roulette wheel selection mechanism to select the samples that can survive in the next generation. In this selection mechanism, each sample occupies a region that is proportional to the value returned by the fitness function. That is, the sample that performs better occupies a larger region in the wheel and will have a higher probability of being selected. Then sample points are randomly selected in the wheel, and the selected samples will be put into the crossover pool for the crossover operation. We have also chosen the one-point

crossover mechanism for the crossover operation, where a random cut point is selected. In each generation, we also perform the mutation operation on samples selected with the probability of 0.01. The mutation operation switches only one randomly selected bit of the selected sample.

4.2 Definitions of Fitness Functions

Fitness functions provide the basis for selecting good samples that can survive in the next generation. For different scenes and desired behaviours, the desired fitness function may also be different. However, instead of designing a specific fitness function for each behaviour, we have designed several elementary fitness functions that can be used to compose the final fitness function for the desired behaviour. The system allows the designer to use linear combination of the elementary fitness functions to specify how the desirable behaviour should be. Most of these elementary fitness functions are computed based on the spatial relation between a robot and its neighbours. The neighbours of a robot are defined as the k -nearest robots, where k is specified by the user. According to the psychological study in [11], we have set the value of k seven in the current system. Next, we describe the elementary fitness functions used in this work.

1) *Inter-robot distance* (G_m): For each robot, the inter-robot distance is defined as the average distance of the k -nearest neighbours around the robot itself. The relative distance to robot i is denoted as r_i , and their average is denoted as R_i as shown in Eq. (2). For a given frame, the difference between the average and the user-specified value is the main performance index. This value is normalized by some quantization factor Q_d to make it fall in the interval $I_f=[0,1]$, as shown in Eq. (3). The overall system performance is computed as the average of all robots over the whole path as shown in Eq. (4)

$$R_j = \frac{\sum_{i=1}^{i=k} r_i}{k} \quad (2)$$

$$G_j = \frac{Q_d}{|R_j - R_c|}, \text{ s.t. } 0 < G_j \leq 1 \quad (3)$$

$$R_m = \frac{\sum_{j=1}^{j=N} R_j}{N}, G_m = \frac{\sum_{m=1}^{m=L} R_m}{L} \quad (4)$$

2) *Distance to the goal* (G_g): The calculation is similar to the inter-robot distance described above except for that the distance is computed with respect to the goal instead of each robot. To save space, we do not repeat the formula here. Instead of being a physically existing object, the goal could also be a designated position behind a possibly moving leader. Again, the value returned by this fitness function is also normalized to the interval I_f .

3) *Number of collisions* (G_c): Collisions are considered as an undesirable effect in crowd simulation. Therefore, the larger the number of collisions, the smaller the value that this fundamental fitness func-

tion should return. Assume that N_c denotes the number of robots that is in collision with other robots or obstacles, and N denotes the total number of robots in the simulation. Then, the collision ratio s_j is defined as the percentage of robots that are in collision as shown in Eq. (5). The overall elementary fitness function is then defined as the average of this ratio over the whole path as shown in Eq. (6).

$$s_j = \frac{N - N_c}{N} \quad (5)$$

$$G_c = \frac{\sum_{j=1}^{j=L} s_j}{L} \quad (6)$$

4) *Orientation consistency* (G_a): This elementary fitness function is defined to measure orientation consistency with neighbouring robots. The average orientation difference of the robot with respect to other neighbouring robots is computed in Eq. (7). Orientation consistency for a single robot, as shown in Eq. (8), is defined as the complement of the orientation difference. The fitness function for the whole crowd is then defined as the average of all robots over the whole path as shown in Eq. (9).

$$A_j = \frac{\sum_{i=1}^{i=k} |\theta_j - \theta_i|}{k}, i \neq j \quad (7)$$

$$B_j = (1 - \frac{A_j}{\pi}) \quad (8)$$

$$B_m = \frac{\sum_{j=1}^{j=N} B_j}{N}, G_a = \frac{\sum_{m=1}^{m=L} B_m}{L} \quad (9)$$

5) *Distance consistency* (G_d): The consistency of the distances of a robot from other neighbouring robots is defined as the standard deviation of these distances. The average distance and standard deviation are first computed according to Eq. (10) and (11). The distance consistency for a robot is then defined as the inverse of standard deviation multiplied by some quantization factor Q_σ (Eq.(12)). In addition, we need to make sure that the consistency value will always fall into the range I_f . The overall performance index is computed as the average of all robots over the whole path as shown in Eq. (13).

$$R_{mean} = \frac{\sum_{i=1}^{i=k} r_i}{k} \quad (10)$$

$$\sigma_j = \frac{\sqrt{\sum_{i=1}^{i=k} (R_i - R_{mean})^2}}{k} \quad (11)$$

$$F_j = \frac{Q_\sigma}{\sigma_j}, \text{ s.t. } 0 < F_j \leq 1 \quad (12)$$

$$F_i = \frac{\sum_{j=1}^{j=N} F_j}{N}, G_d = \frac{\sum_{m=1}^{m=L} F_i}{L} \quad (13)$$

The overall fitness function (F_{sum}) for evaluating the performance of a crowd simulation is computed based on a linear combination of the elementary fitness functions defined above. The formula is shown in Eq. (14).

$$G_{sum} = S_m * G_m + S_g * G_g + S_c * G_c + S_a * G_a + S_d * G_d \quad (14)$$

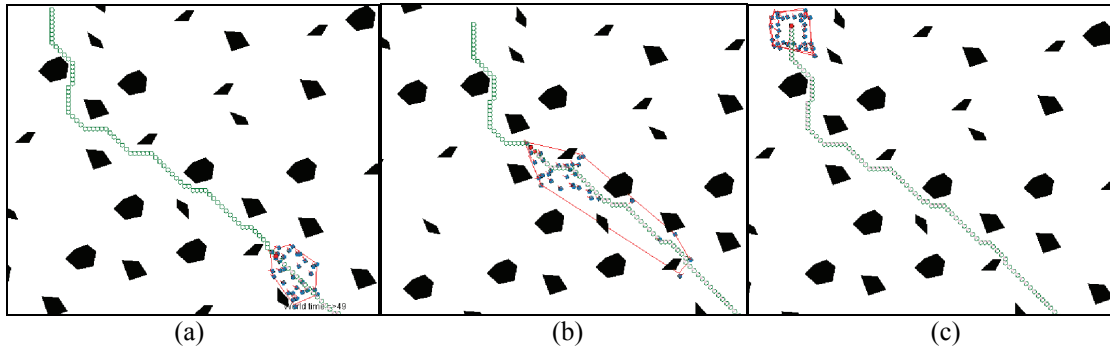


Figure 3: Example of simulation results on the following behaviour (B2) with a small inter-robot distance in a space cluttered with small obstacles (E3)

Table 1: Optimal weights generated by the genetic algorithm for various environments and behaviours (B: Behaviour, E: Environment)

	s_1	s_2	s_3	s_4	s_5
B1-E1	0.42	0.03	0.15	0.84	0.44
B1-E2	0.87	0.31	0.99	0.49	0.16
B1-E3	0.65	0.12	0.19	0.01	0.12
B2-E1	0.53	0.15	0.77	0.36	0.62
B2-E2	0.90	0.39	0.80	0.06	0.16
B2-E3	0.95	0.21	0.93	0.07	0.04
B3-E1	0.94	0.42	0.90	0.14	0.59
B3-E2	0.98	0.43	0.95	0.75	0.71
B3-E3	0.97	0.32	0.63	0.09	0.27

According to the desired behaviour, a designer makes use of the weights (S_m, S_g, S_c, S_a, S_d) to compose the final fitness function from the elementary ones. We assume that these weights are more intuitive to set compared to the weights in Eq.(1) and they should be the same for the same desired behaviour of crowd movement. However, the optimal weights in Eq.(1) may be scene specific, as will be shown in the next section.

5 Experimental Design and Results

We have implemented the simulation system and the genetic algorithm in Java. In the conducted experiments that we report in this section, the total length of the simulation and the desired distance between the robots are specified by the user. In addition, only one group of crowd is considered at a time.

5.1 Scene and Behavior Variations

Generally speaking, it is difficult to classify environments or define typical scenes. Nevertheless, we have defined and tested three types of scenes in our current experiments including an open space without obstacles (E1), a scene with a narrow passage (E2), and one cluttered with small obstacles (E3).

Three types of behaviors have been used in our experiments: *group moving* (B1), *following* (B2), and *guarding* (B3). The group moving behavior is to keep the crowd moving with a given inter-robot distance in a group. As for the following behavior, the objective is to pursue a possibly moving goal as closely as possible. As for the guarding behavior, the crowd is supposed to surround the possibly moving goal, which could also be another leader robot.

For each different behavior, we have used the elementary fitness functions to compose the final fitness function used in the genetic algorithm. We assume that these weights are used to express the designer's intention for ideal behaviours and should be independent of the environments. Nevertheless, the user needs to give the inter-robot distance in order for the simulation to converge into the desirable behaviours.

5.2 Experimental Results

We have conducted experiments with the genetic algorithm to acquire the set of parameters for the desired behavior for each given environment. The set of weighting parameters generated by the system are given in Table 1. In order to validate the parameters, we used the parameters obtained for E1 to run simulations in E2 and E3. The overall scores G_{sum} returned by the fitness function are 295, 150, and 262, respectively. If we use the optimal parameters generated for E2 and E3, respectively, to run the experiments again, the scores are improved to 271 and 284, respectively. Although the scores in the cluttered environments (E2 and E3) are not as good as E1 as expected, the scores have been greatly improved when the optimal parameters are used. This experiment reveals that the optimal weighting parameters for the virtual force are scene dependent.

The simulation results are illustrated in Figure 3 to Figure 5. A convex hull (in red) is computed to illustrate the boundary of a crowd (20 robots) in each example. In Figure 3, we show an example of the following behavior (B1) for the crowd with a small inter-robot distance in a space cluttered with obstacles (E3). In Figure 4, we show a group moving example (B1) where the desired inter-robot distance is set to a

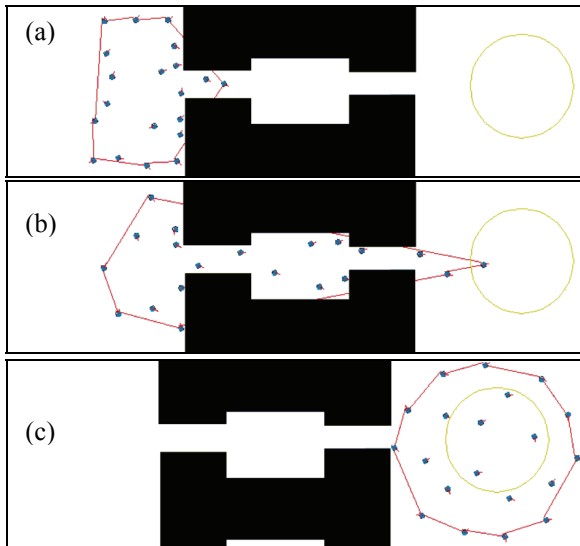


Figure 4: Example of simulation results for the group moving behaviour (B1) with a large inter-robot distance passing a narrow passage (E2)

higher value, and the crowd needs to pass the narrow passage (E2) in order to reach the goal. In Figure 5, we show the case of body guarding a specific leader robot depicted in red (B3) in an open space (E1). The crowd succeeds in surrounding the leader all the way while they are moving.

6 Conclusion

As the applications of virtual crowd simulation increase in recent years, the demands for better simulation tools for creating desirable results also increase. In this paper, we have proposed to formulate the problem as an optimization problem on the parameters of virtual forces with the genetic algorithm. The fitness functions used in the genetic algorithm is composed according to the desired behaviors from elementary ones designed for evaluating a specific aspect of simulation. Our preliminary experiments reveal that the genetic algorithm is a good way to automate the time-consuming process of generating the optimal set of parameters for virtual forces for a given scene and desired movement behavior. In the future, we will conduct experiments with more types of environments and behaviors in order to shed some light on the limitation of using this type of virtual force mechanism to simulate crowd motions.

7 Acknowledgement

This work was supported by National Science Council under contract NSC94-2213-E-004-006.

References

- [1] T. Balch and R.C. Arkin, "Behaviour-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, 14(6), 926-939, (1998).
- [2] E. Bouvier, E. Cohen, and L. Najman, "From crowd

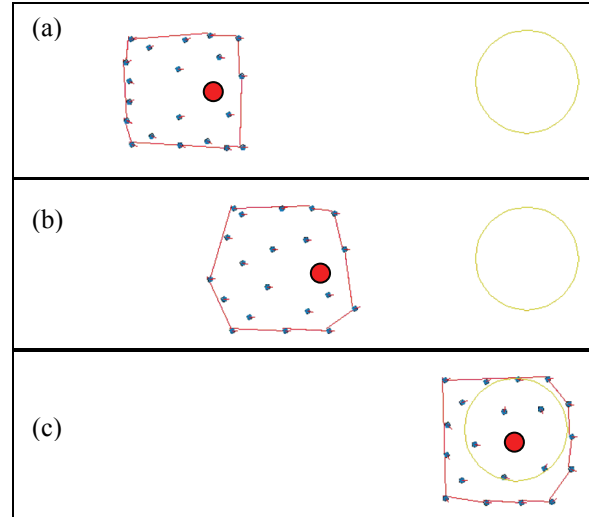


Figure 5: Example of simulation results for the guarding behaviour (B3) in open space (E1)

simulation to airbag deployment: particle systems, a new paradigm of simulation," in *Journal of Electronic Imaging*, Vol. 6, No.1, 94-107, (1997).

- [3] D.C. Brogan and J. Hodgins, "Group Behaviors for Systems with Significant Dynamics," in *Autonomous Robots*, 4, 137-153, (1997).
- [4] D.C. Brogan, R.A. Metoyer and J.K. Hodgins, "Dynamically simulated characters in virtual environments," in *IEEE Computer Graphics and Applications*, Vol.18, No5, 58-69, (1998).
- [5] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive Model: Knowledge, Reasoning, and Planning for Intelligent Characters," in *Proc. of ACM SIGGRAPH*, 29-38, (1999).
- [6] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA, MIT Press, (1992).
- [7] T.Y. Li, Y.J. Jeng, and S.I. Chang, "Simulating virtual human crowds with a leader-follower model," in *Proc. of 2001 Computer Animation Conf.*, (2001).
- [8] T.Y. Li and H.C. Chou, "Motion Planning for a Crowd of Robots," in *Proc. of the 2003 Intl. Conf. on Robotics and Automation*, (2003).
- [9] M. Mitchell, *An Introduction to Genetic Algorithms*, Cambridge, MA, MIT Press, (1998).
- [10] G. Mitsuo and C. Runwei, *Genetic Algorithms & Engineering Design*, John Wiley & Sons, Inc., (1997).
- [11] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63, pp. 81-97, (1956).
- [12] S. R. Musse and D. Thalmann, "Hierarchical model for real time simulation of virtual human crowds," in *IEEE Trans. on Visualization and Computer Graphics*, 7(2):152-164, (2001).
- [13] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioural model," *Computer Graphics*, 25-34, (1987).
- [14] C. W. Reynolds, "Steering behaviours for autonomous characters," in *Proc. Of Game Developers Conf.*, (1999).
- [15] N. Saiwaki, T. Komatsu, T. Yoshida, and S. Nishida, "Automatic generation of moving crowd using chaos model," in *Proc. of IEEE Int. Conf. on System, Man and Cybernetics*, 3715-3721, (1997).