14

An Evolutionary Approach to Crowd Simulation

Tsai-Yen Li and Chih-Chien Wang

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

Summary. In previous work, virtual force has been used to simulate the motions of virtual creatures, such as birds or fish, in a crowd. However, how to set up the virtual forces to achieve desired effects remains empirical. In this work, we propose to use a genetic algorithm to generate an optimal set of weighting parameters for composing virtual forces according to the given environment and desired movement behaviour. A list of measures for composing the fitness function is proposed. We have conducted experiments in simulation for several environments and behaviours, and the results show that compelling examples can be generated with the parameters found automatically in this approach.

Keywords: Crowd simulation, Genetic algorithm, Robot formation, Multiple robot system.

14.1 Introduction

Formation control for multi-robot systems is a classical robotic problem that has attracted much attention in the literature [1]. In recent years, the techniques of simulating virtual crowd also have created potential applications in contexts such as a virtual mall, digital entertainment, transportation and urban planning. Most of the current systems for crowd simulation adopt a local control approach. One of the common approaches to this problem in computer animation adopts the virtual force model [10], whereby the movement of each robot is affected by virtual forces computed according to its spatial relation between itself and other robots or objects in the environment. However, designers still need to face the problem of how to choose the most appropriate forces and select the best weights. In addition, there is no objective way to evaluate the result.

Crowd motions can be generated by simulation or by planning. In this work, we focus on the approach of simulation. Bouvier [2], Brogan and Hodgins [3] used a physical-based particle system to simulate a crowd of athletes such as

T.-Y. Li and C.-C. Wang: *An Evolutionary Approach to Crowd Simulation*, Studies in Computational Intelligence (SCI) **76**, 119–126 (2007)
www.springerlink.com

runners or bikers competing in a field. Reynold [10,11] sought to apply virtual forces to individual robots to create steering behaviours for the whole crowd. Tu and Terzopoulos [4] used rule-based finite-state machine to construct a cognitive model of fish and succeeded in creating several interesting behaviours including flocking. Muse and Thalmann [9] used behaviour rules to design virtual characters and used scripts to construct complex social interactions.

In this chapter, we propose to model the problem of generating good examples of the movement of crowds by designing an appropriate parameterization and evaluation mode and adopt the genetic algorithm [7] to search for an optimal set of parameters. The parameterized virtual-force model is then evaluated by an array of measures to describe the desired crowd behaviours. We have also conducted experiments to generate parameter sets for several common crowd behaviours in various environments consisting of different spatial structures. The considered materials are derived from [6] where they were first presented and discussed.

14.2 Design of Movement Model and Virtual Forces

In this work, we have adopted the virtual force model proposed in [10] as the way to affect the movement of each individual robot. We assume that the robots move under the influence of virtual forces proposed in [11], and that they must respect maximal speed limits in both translation and rotation. In addition, a robot is given a view angle of 330 and a constant view distance that is 20 times the size of the robot. Because the virtual forces are computed locally for each robot, only the robots or obstacles that are within the range of view have effect on the computation of the virtual forces for the robot. The computed forces are used to update the next configuration of the robot. However, if the new configuration is in-collision, the system makes use of various local strategies to avoid such a collision.

The motions of the crowd in our simulation system are driven by five types of virtual forces: *separation, alignment, cohesion, following* and *collision*. First, the separation force (F_{sep}) is a repulsive force computed proportional to the distance between the robot and other neighbouring robots within the range of view. The resultant force is the summation of all of the forces exerted by each individual robot. The effect of the force is to maintain a safe distance between the robots. Second, the alignment force (F_{align}) is used by a robot to align its velocity and orientation with other neighbouring robots. An average velocity for the robots within the range of view is computed first. The alignment force is then computed according to how the velocity of the current robot deviates from the average one. Third, the cohesion force (F_{coh}) is computed according to the difference vector between the current position of the robot and the centre of the neighbouring robots within the range of view. Fourth, the following force (F_{fol}) 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. We have adopted the model proposed in [5] 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. Finally, the collision force (F_{col}) is a repulsive force exerted by the environmental obstacles when a condition of collision is predicted after certain period of time.

Our system uses a linear combination of the normalized component forces described above to compute the final virtual force as shown in (14.1).

$$F = s_1 * F_{sep} + s_2 * F_{align} + s_3 * F_{coh} + s_4 * F_{fol} + s_5 * F_{col} \tag{14.1}$$

The set of weights $s = (s_1,\ s_2,\ s_3,\ s_4,\ s_5)$ determines how the forces are composed to affect the behaviour of the robot. The appropriate setting of the weights remains as an empirical problem for the designer, which is the reason that we propose to automate the search process for an optimal solution with the genetic algorithm described in the next section.

14.3 Evolution of Motion Behaviour of Crowd

In a crowd simulation, the trajectory of the crowd may vary greatly according to the scenario and the environment in which the crowd is situated. Therefore, the designer is required to tune the weights of the component forces in order to achieve the results desired within the framework of virtual forces. The complexity of the parameter space and the time-consuming process of developing the simulation make it a difficult optimization problem for human or for machine. Therefore, we propose to solve the problem by using a genetic algorithm, commonly used to solve the problem of searching for an optimal solution in a large search space [7].

In our problem, 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. We have chosen the roulette wheel selection mechanism to select the samples that will survive in the next generation. Sample points are randomly selected in the wheel for the one-point crossover operation. In each generation, we also perform the mutation operation (switching a random bit) on samples selected with the probability of 0.01.

The desired fitness function in our experiments may also be different for different scenes and different types of behaviours. Instead of designing a specific fitness function for each type of behaviour, we designed several elementary fitness functions that can be used to compose the final fitness function of a behaviour. Most of the elementary fitness functions are computed based on the spatial relation between a robot and its neighbours, defined as the k-nearest robots. Following Miller [8], we currently set the value of k as seven in the

current system. We describe the elementary fitness functions used in this work as follows.

Inter-robot distance (G_m). The inter-robot distance for a robot is defined as the average distance of its k-nearest neighbours. The relative distance to robot i is denoted as r_i, and their average distance between each other is denoted as R_i as shown in (14.2). For a given frame, the difference between the average and the user-specified value is the main performance index. This value is normalized by the quantization factor Q_d to make it fall in the interval $I_f = [0, 1]$, as shown in (14.3). The overall system performance is computed as the average of all robots over the whole path as shown in (14.4).

$$R_j = \frac{\sum_{i=1}^{i=k} r_i}{k} \tag{14.2}$$

$$G_j = \frac{Q_d}{|R_j - R_e|} \ , \text{ s.t. } 0 < G_j \leq 1 \tag{14.3}$$

$$R_m = \frac{\sum_{j=1}^{j=N} G_j}{N} \ , \ G_m = \frac{\sum_{m=1}^{m=L} R_m}{L} \tag{14.4}$$

Distance to the goal (G_g). The calculation is similar to that for inter-robot distance described above except for that the distance is computed with respect to the goal instead of each robot. In the interest of saving space, we do not repeat the formula here. In addition, instead of being a physically existing object, the goal could also be a designated position at a point behind a possibly moving leader.

Number of collisions (G_c). Assume that N_c denotes the number of robots that are in collision with other robots or obstacles, and that 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, and the overall elementary fitness function is then defined as the average of this ratio in movement over the whole path as shown in (14.5).

$$s_j = \frac{N - N_c}{N}, \ G_{\mathrm{c}} = \frac{\sum_{j=1}^{j=L} s_j}{L} \tag{14.5}$$

Consistency in orientation (G_a). In (14.6), we define the average difference in the orientation of a robot with respect to other neighbouring robots and the consistency in orientation for a single robot. The fitness function for the whole crowd is then defined as the average of all robots over the whole path as shown in (14.7).

$$A_j = \frac{\sum_{i=1}^{i=k} |\theta_j - \theta_i|}{k}, i \neq j \qquad B_j = (1 - \frac{\mathrm{A}_j}{\pi}) \tag{14.6}$$

$$B_m = \frac{\sum_{j=1}^{j=N} B_j}{N}, G_a = \frac{\sum_{m=1}^{m=L} B_m}{L} \tag{14.7}$$

Consistency in distance (G_d). The average distance and standard deviation of a robot's distances to its neighbours are first computed according to (14.8).

The consistency in distance for a robot is then defined as the inverse of the standard deviation multiplied by the quantization factor Q_σ in (14.9). The overall performance index is computed as the average of the consistency in differences between all robots over the whole path as shown in (14.10).

$$R_{mean} = \frac{\sum_{i=1}^{i=k} r_i}{k} \ , \sigma_j = \frac{\sqrt{\sum_{i=1}^{i=k} (R_i - R_{mean})^2}}{k} \tag{14.8}$$

$$F_j = \frac{Q_\sigma}{\sigma_j} \ , \ \text{s.t. } 0 < F_j \leq 1 \tag{14.9}$$

$$F_t = \frac{\sum_{j=1}^{j=N} F_j}{N} \ , G_d = \frac{\sum_{t=1}^{t=L} F_t}{L} \tag{14.10}$$

The overall fitness function (F_{sum}), as shown in (14.11), is computed based on a linear combination of the elementary fitness functions defined above.

$$G_{\text{sum}} = S_m * G_m + S_g * G_{\text{g}} + S_c * G_{\text{c}} + S_a * G_a + S_d * G_d \tag{14.11}$$

A designer makes use of the weights, $S = (S_m,\ S_g,\ S_c,\ S_a,\ S_d)$, according to the nature of the desired behaviour, 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 (1) and that they should be the same for the same desired behaviour of crowd movement. However, the optimal weights in (1) may be scene specific, as shown in the next section.

14.4 Experimental Design and Results

We implemented the simulation system and the genetic algorithm in Java. The evolution terminates when the maximal number of generations is reached or the optimal solution converges and do not change for five generations. For each environment and behaviour, we ran the same experiment with three different initial settings to see if they all converge to the same optimal solution. Our observation is that most of the experiments converged after 17–19 generations.

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

Three types of behaviours were tested in our experiments: *group moving* (B1), *following* (B2) and *guarding* (B3). The group moving behaviour refers to keeping the crowd moving with a given inter-robot distance in a group. The following behaviour refers to pursuing a possibly moving goal as closely as possible, and the guarding behaviour refers to surrounding a possibly moving goal. For each different type of behaviours, we used a different set of elementary fitness functions to compose the final fitness function in order to express the designer's intention.

We conducted experiments with the genetic algorithm described in the previous section to acquire the set of parameters for the desired behaviour for each given environment. The set of weighting parameters generated by the system are given in Table 14.1. These optimal parameter sets vary greatly for different scenes and for different types of environment. For example, for the behaviour B1, the parameter s_3 varies from 0.15 to 0.99 for different environments. For the environment E1, the parameter s_3 varies from 0.19 to 0.95. The generated weights for the following force (s_4) and the collision forces (s_5) for the cluttered environment (E3) are relatively smaller than other environments since these two forces tend to jam-pack the crowd in such an environment.

In order to validate the parameters, we used the parameters obtained for E1 to run simulations with E2 and E3. The overall scores G_{sum} returned by the fitness function are 295, 150 and 262, respectively. When we used the optimal parameters generated for E2 and E3, respectively, to run the experiments again, the scores improved to 271 and 284, respectively. Although the scores in the cluttered environments (E2 and E3) are not as good as might have been anticipated from the result for E1, the scores were greatly improved when the optimal parameters were used. This experiment reveals that the optimal weighting parameters for the virtual force are scene dependent.

In Fig. 14.1, we show an example of the following behaviour (B2) for the crowd with a small inter-robot distance in a space cluttered with obstacles (E3). In Fig. 14.2, we show an example of group movement (B1) where the desired inter-robot distance is set to a higher value, and the crowd needs to pass the narrow passage (E2) in order to reach the goal.

Table 14.1. 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

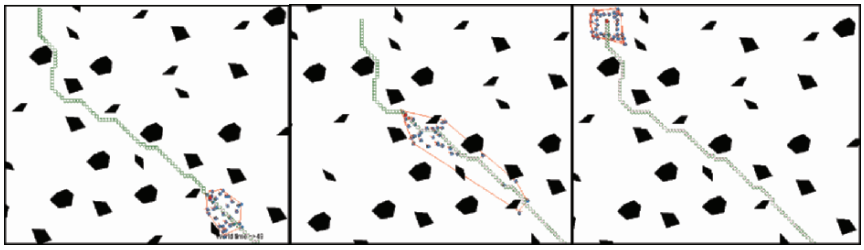

Fig. 14.1. Example of simulation results on the following behaviour (B2) with a small inter-robot distance in a space cluttered with small obstacles (E3)

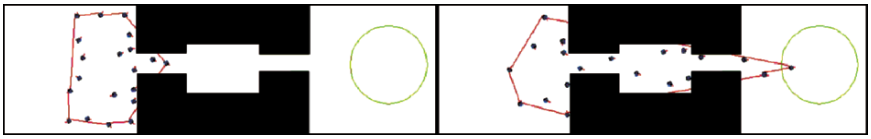

Fig. 14.2. Example of simulation results for the group moving behaviour (B1) with a large inter-robot distance passing a narrow passage (E2)

14.5 Conclusions

In this chapter, we have proposed to formulate the problem of how to compose virtual forces to drive the simulation as an optimization problem on the weighing parameters for virtual forces by the use of a genetic algorithm. The fitness functions used in the genetic algorithm are composed according to the desired behaviours from the elementary ones designed for evaluating a specific aspect of crowd motion. 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 a given scene and a desired behaviour of movement.

References

1. T. Balch and R.C. Arkin, "Behaviour-based formation control for multirobot teams," *IEEE Trans. on Robotics and Automation*, 14(6), pp 926–939 (1998).
2. E. Bouvier, E. Cohen and L. Najman, "From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation," *J. of Electronic Imaging*, 6(1), pp 94–107 (1997).
3. D.C. Brogan and J. Hodgins, "Group behaviors for systems with significant dynamics," *Autonomous Robots*, 4, pp 137–153 (1997).
4. J. Funge, X. Tu and D. Terzopoulos, "Cognitive model: knowledge, reasoning, and planning for intelligent characters," *Proc. of ACM SIGGRAPH*, pp 29–38, Los Angels (1999).

5. T.Y. Li, Y.J. Jeng and S.I. Chang, "Simulating virtual human crowds with a leader-follower model," *Proc. of 2001 Computer Animation Conf.*, Seoul, Korea (2001).
6. C.C. Wang, T.Y. Li. "Evolving Crowd Motion Simulation with Genetic Algorithm," *Proc. 3rd Intl. Conf. on Autonomous Robots and Agents - ICARA'2006*, Palmerston North, New Zealand, pp 443–448 (2006).
7. G. Mitsuo and C. Runwei, *Genetic Algorithms & Engineering Design*, John Wiley & Sons. Inc., New York (1997).
8. 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).
9. S.R. Musse and D. Thalmann, "Hierarchical model for real time simulation of virtual human crowds," *IEEE Trans. on Visualization and Computer Graphics*, 7(2), pp 152–164 (2001).
10. C.W. Reynolds, "Flocks, herds, and schools: A distributed behavioural model," *Computer Graphics*, pp 25–34 (1987).
11. C.W. Reynolds, "Steering behaviours for autonomous characters," *Proc. of Game Developers Conf.*, San Jose (1999).