2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

PSO-Based Evacuation Simulation Framework

Pei-Chuan Tsai
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: pctsai@nclab.tw

Chih-Ming Chen
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: ccming@nclab.tw

Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: ypchen@cs.nctu.edu.tw

***Abstract*—Evacuation simulation is a critical and important research issue for people to design safer building layouts or plan more effective evacuation routes. Many studies adopted methodologies in evolutionary computation into the evacuation simulation systems for finding better solutions. To simulate human behavior or crowd motion is one key factor to the practicality of the system. Particle swarm optimization algorithm (PSO), which is originated from the inspiration of bird flocking, is commonly applied to model human behavior. Based on the PSO-based human behavior simulation, many studies have got good results on evacuation simulation. However, the configurations of describing the experiment environment in the literature are complicated and specialized for certain specific scenarios. Observing the fact, we propose a new PSO-based simulation framework in order to provide a simple and general way to configure various simulation scenarios. This work adopts our previously proposed PSO-based crowd movement controlling mechanism and introduces new mechanisms to make the simulation fitting into evacuation circumstance more real. In the proposed framework, all people, obstacles, exits, and even the evacuation guide indicators are modeled as the original component of the PSO algorithm. It is convenient to setup the simulation environment upon the framework. Therefore, taking the proposed work as a research tool will be advantageous when the issue of evacuation simulation is investigated.**

I. Introduction

Evacuation simulation is useful and critical in several aspects such as evacuation planning and building safety evaluation. Lots of evacuation simulation models have been developed to apply in various situations. Many of them utilize methodologies in evolutionary computation to find better evacuation plan or building layout. Garrett et al. used Genetic Algorithms (GA) and Estimation of Distribution Algorithms (EDA) to find the better placement of exit positions in two specified scenarios [1]. Zong et al. applied multi-ant colony to model evacuation routing with mixed traffic flow and showed that the congestion on the optimal exit caused by single ant colony modeling can be prevented by providing communications between different colonies [2]. Yusoff, Ariffin, and Mohamed proposed an improved discrete particle swarm optimization (DPSO) scheme to solve vehicle assignment problem for evacuation [3]. Although there are many kinds of evacuation simulation models designed to solve diverse problems, the simulation of human behavior is always an important part in these modeling systems. In order to appropriately simulate the behavior or the movement of human, many researchers take advantage of particle swarm optimization in evolutionary computation. Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart in 1995 which was inspired by the behavior of bird flocking [4]. Researchers set up their environments and used the particle swarm movement to simulate the human moving in the evacuation scenarios [5], [6]. Kou et al. further combined PSO and a non-dominated sorting genetic algorithm to simulate and optimize the evacuation plan [7].

Observing that although all studies which used PSO-based human behavior simulation got good experimental results, the settings of environment in each work are specialized and not compatible with another. Therefore, this paper proposes a PSO-based evacuation simulation framework to provide a general configuration for describing environmental settings and human behavior as well. The framework adopts the PSO-based crowd movement controlling mechanism proposed in [8], [9]. In the controlling mechanism, the obstacles were formulated as parts of objective functions and the collision avoidance mechanism between crowd members was also provided. By the proposed simulation framework, it would be convenient to describe various scenarios and apply to find the better evacuation plans or better building layouts for evacuation.

The remainder of the paper is organized as follows. Section II gives the details of the PSO-based crowd controlling scheme proposed in [8], [9], including the mapping from human movement to particle swarm, obstacle formulation, and collision avoidance mechanism. In section III, we propose the improvements applied on the framework which make the simulation be more realistic in evacuation scenarios. For demonstrating the proposed simulation framework, we set up several scenarios to simulate. The details of configurations and the experimental results are presented in section IV. Finally, the conclusions and our contributions are given in section V.

II. PSO-based simulation scheme

In 1995, Kennedy and Eberhart proposed PSO algorithm to solve optimization problems. The PSO method formulates each possible solution as a particle in a D-dimensional space:

$$x = [x_1, x_2, ..., x_D]^T.$$

Each particle has its position, velocity, and an objective value determined by the objective function. The new position of the particle is decided by its experience and social knowledge in order to move toward to expected and currently known best solution. Before deciding the new position, the velocity is

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

calculated first. The velocity of particle p_i at time $t+1$ is determined by the following equation

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 \left(p_{iBLS} - p_i(t)\right) + c_2 r_2 \left(p_{BGS} - p_i(t)\right),$$

where v_i and p_i are the velocity and position of the ith particle. ω is the weight for the previous velocity. p_{iBLS} is the best position according to its experience and p_{BGS} is the global best position that have achieved by the swarm. c_1, c_2 are the parameters to control the influence of p_{iBLS} and p_{BGS}. r_1, r_2 are random numbers selected from $[0, 1]$ to make the velocity more diverse. After computing the velocity $v_i(t+1)$, the new position of the ith particle can be determined by

$$p_i(t+1) = p_i(t) + v_i(t+1).$$

In each iteration, every particle updates its velocity/position and then evaluates its own objective value according to the objective function. Among all the new objective values, the values of p_{iBLS} and p_{BGS} may also be updated to take the swarm toward the optimal solution.

Since the particle is free to fly to any possible position, there has some modifications need to be considered for applying to simulate human behavior. Based on [8], [9], there are three major parts to be modified: 1) restriction of particle movement to simulate human moving path; 2) obstacle modeling; 3) collision avoidance between crowds and obstacles. The details are described in following sections.

A. Particle movement

First of all, considering that the evacuation simulation usually examined in a $2D$ space, we restrict the particle position and velocity into $2D$ space as $p_i = [p_{ix}, p_{iz}]^T$ and $v_i = [v_{ix}, v_{iz}]^T$. In order to simulate human pace and prevent the particle from jumping too far to be unreal for human, the velocity is separated into direction part $D_i = [D_{ix}, D_{iz}]^T$ and speed part S_i. The direction part is used to decide where to go and the speed part can therefore be used to control the moving distance of each step. The update equation for the direction part is similar to update the velocity

$$D_i(t+1) = \omega D_i(t) + c_1 r_1 \left(p_{iBLS} - p_i(t)\right) + c_2 r_2 \left(p_{BGS} - p_i(t)\right),$$

where $(p_{iBLS} - p_i(t))$ and $(p_{BGS} - p_i(t))$ are represented in unit vector for indicating the direction only. Other parameters are defined as the original PSO. By deciding the direction of the particle, the new position of the particle can be determined as in original PSO

$$p_i(t+1) = p_i(t) + S_i(t+1) \times D_i(t+1),$$

where $S_i(t+1)$ is the speed part within a range $[0, V_{max}]$. V_{max} can be set as a step size or determined by other user defined parameters. For making the movement of the particle more like a human being, the speed can be updated proportional to the reverse of the particle's objective value. This setting is inspired by the intuition that human would slow down their speed when approaching close to an obstacle. Since PSO is established to search the minimum around the environment, the objective value should be higher when the particle is close to an obstacle and therefore a lower velocity is rendered.

B. Objective function

In [8], [9], a cost function to evaluate the relations between the particle and any other objects in the search space was defined. The cost value of particle p with object q is computed by the following exponential function

$$Cost(p,q) = \exp\left(-\left(\frac{(p_x - q_x)^2}{(\sigma_{p_x} + \sigma_{q_x})^2} + \frac{(p_z - q_z)^2}{(\sigma_{p_z} + \sigma_{q_z})^2}\right)\right), \quad (1)$$

where (p_x, p_z) is the position of the particle p, (q_x, q_z) is the position of the object q, and $(\sigma_{p_x}, \sigma_{p_z})$, $(\sigma_{q_x}, \sigma_{q_z})$ is the scope of the particle p and the object q, respectively. The scope represents the coverage area of the object. Since the object q can be an obstacle, target, or other particles, the scope would be different. For example, the scope of an object with 20 by 40 would be set to $(\sigma_{q_x}, \sigma_{q_z}) = (10, 20)$. For representing the target, the scope may be set to the whole search area. Among these related configurable parameters, the object is easy to adjusted and controlled for satisfying user needs. By defining the cost function, the search space with a minimum position as target can be obtained by overlaying these exponential functions with identical or different parameters. Therefore, the objective function of the scheme is defined as

$$F_{obj}(p) = c_{obs} \times \max_{o \in O}\left(Cost(p,o)\right) + \frac{1}{Cost(p,g)},$$

where O is the set of all the obstacles including specified obstacles and other human in the crowd, g is the target for particle to move forward, and c_{obs} is a constant for user to control the importance between the obstacles and the target.

Observing the definition of the objective function, it can be found that the particle would have lower objective value when approaching to the targets. On the other hand, when the particle moves close to an obstacle, no matter it is a wall in real world or other human in the crowd, the objective value would rise. Under the objective function, the PSO optimization ability could be applied in such an evacuation simulation model. The evacuation exits or safety zones can be modeled as targets, and the walls, holes, or furniture can be modeled as obstacles. Another advantage of the simulation scheme is that no extra model needs to be added for including new objects. All objects or other kinds of particles, such as moving cars or elders, can be modeled by controlling the introduced parameters. With appropriate parameter settings, the PSO-based simulation scheme would achieve its design goal and the particles would converge to the desired target by a reasonable path with a high probability.

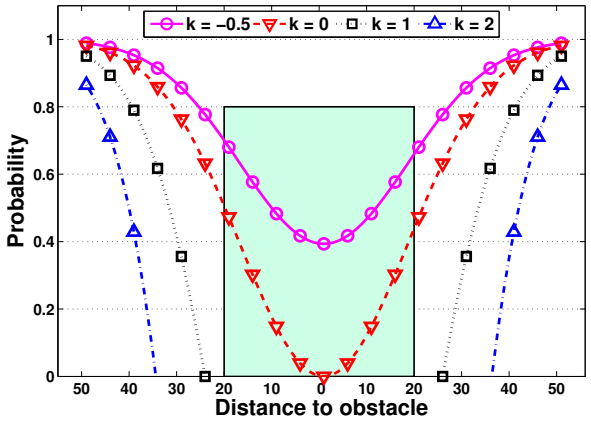


Fig. 1. The figure shows the acceptance probability of a new position for different parameter sizes, in which the radius of particle and the obstacle are respectively 5 and 20.

C. Collision avoidance

Although the objective function was designed to make the particle to walk around the obstacles, there is still a possibility that the particle would go through or collide with the obstacles. To avoid this possibility, the new position of the particle should be checked whether it can be accepted or not. The check mechanism is also based on exponential functions. Depending on users' decision or preference, other function families can be adopted in both the objective function and check mechanism. The exponential function is chosen in this study because it is easy to compute and manipulate. The check mechanism uses the probability as following to decide the acceptance of a newly computed position

$$Prob(f) = 1 - \frac{f}{e^{-k}}, \tag{2}$$

where f is the cost of the newly computed position and k is a constant to control the behavior when a particle comes close to the obstacle. The constant k lets the user conveniently specify the acceptance around the obstacles. According to the acceptance probability function, there exists a hard boundary when setting $k = 1$ because the probability would be 0 if the cost is $f = e^{-1}$, which means the particle and the obstacle stay close together by the definition of the cost function (Equation (1)). Therefore, the collisions can be prevented by the check mechanism. Assume that the radius of the particle and the obstacle are respectively 5 and 20. Figure 1 illustrates the acceptance probability while the particle is close to the obstacle. If the new position is not accepted, another direction, such as adding a random degree between $\pm 20^\circ$ to original direction, would be tested for making the path smoothly. The position of the lower cost would be chosen as the new position of the particle.

III. Improvements applied on the PSO-based simulation framework

Although the proposed simulation scheme have dealt with almost all the differences between crowd in real and swarm in free space, some details still need to be attended, such as the solution to a trapped particle to get out and the non-line-of-sight problem. We propose two improvements in our simulation framework to solve these situations and explain in the following sections.

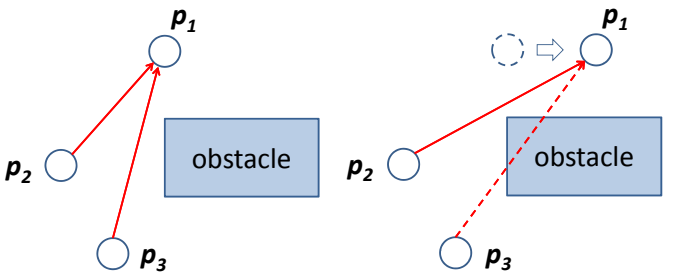
A. Local search

Observing the collision avoidance mechanism adopted in [8], [9], we find that a particle still has the probability to step into a position too close to an obstacle or facing a non-round obstacles such that it can not find an accepted position in the next iteration, even though it has searched all directions in the range between $\pm 20^\circ$. The particle is finally trapped on that position and would not move to any where till the end of the simulation. To get the particle out of this situation, we improve the local search for finding possible new positions. Local search is a common mechanism used with PSO and also applied in [8], [9] for collision avoidance. However, the search area is restricted within $\pm 20^\circ$ to the original direction which may be sufficient for round obstacles. For the obstacles with sharp edges or rectangles, particles can still not find an accepted position in such a restricted area. For getting the particle out, we extend the local search area incrementally until find an accepted direction to move. For an unaccepted direction, D, computed from the original PSO, we randomly sample a new direction from $[D - \frac{\theta}{2}, D + \frac{\theta}{2}]$ and check the acceptance. If the direction is still unaccepted, we repeatedly sample another one to check for at most a configurable times, e.g., 20 times. If all these sampled directions are unacceptable, we then extend the search range with θ each time, i.e., the search range would be extended from θ to $2\theta, 3\theta, ...,$ etc.

The search mechanism is inspired from the behavior that when a human faces the wall and wants to find a way to keep moving, he may walk along the wall to seek a door or the end of it. The movement of the trapped particle based on this incrementally search would finally act as moving along the edge of the obstacle. The extended local search area and the random sample mechanism provide the particle the opportunity to move on and also avoid the oscillation around the area.

B. Global best value updating mechanism

The original PSO has no obstacles in the search space and the particles always update the global best information no matter where the position is. In real world, human would not know the information if there are obstacles between itself and the position with the global best objective value. In such a circumstance, the global best information should not be updated to the particles with such positions unless they "see" someone who has the best position. It is usually called as "non-line-of-sight" effect. As a consequence, the update mechanism should be modified to fit into the real world properly. For



(a) Before particle p_1 moves to new position (b) After particle p_1 moves to new position

Fig. 2. Illustration of global best information p_{iBGS} update mechanism.

modeling non-line-of-sight effect, we make each particle keep itself global best information, p_{iBGS}, instead of sharing an universal global best information, p_{BGS}, with all particles. The global best information of each particle would keep the best position that ever appears in its sight. The information would be updated only if there is a particle with better objective value coming into its sight, i.e., no obstacles between them. Figure 2 illustrates an example of the improved update mechanism. Assume the objective value of particle p_1 is the best of all the other particles p_2, p_3. In Figure 2(a), since both p_2 and p_3 can see the particle p_1, they will keep the position of p_1 as their global best information. As shown in Figure 2(b), after particle p_1 moves to the new position with better objective value, only particle p_2 will get the updated global best information because particle p_1 still in its sight. The particle p_3 would not have the new information of p_1 in this situation. We use this mechanism to model the situation that people would not ever know what the direction someone who is covered behind an obstacle moves to. Therefore, the modified computation of each particle would be the following steps in each iteration:

1) Find an accepted new position,
2) Calculate the objective value,
3) Update its local best information and global best information if needed
4) Update the global best information of all particles in its sight

By applying aforementioned improvements, the simulation framework would much well model the actual human behavior. We provide simulations in several different scenarios in the following section to show the capability and flexibility of our proposed simulation framework.

IV. Experimental Results

In this section, we set up four room layouts to demonstrate our simulation framework. The layouts are shown in Figure 3. All the rooms are in the same size of 600×400 and the coordinates are specified in Figure 3. Each particle is set as a circle with radius 5, i.e., $\sigma_{p_x} = \sigma_{p_z} = 5$, and the maximal speed is $V_{max} = 5$. The parameters used in PSO are given in Table I. We assume that particles would refer to the position with the global best objective value with a higher priority than the local best position ever seen. The target of each

TABLE I
The settings of parameters used in PSO.

Parameter	ω	c_1	c_2	k	C_{obs}	V_{max}
Value	1.0	0.2	1.0	1.0	0.1	5.0

TABLE II
The simulation results of each layout.

Layout	3(a)	3(b)	3(c)	3(d)
Average iterations	134.783	104.504	119.427	117.785
Worst iterations	334.977	291.120	250.203	244.260
Standard deviation	12.486	9.043	9.008	8.508

TABLE III
URLs of video clips demonstrating simulation of each layout.

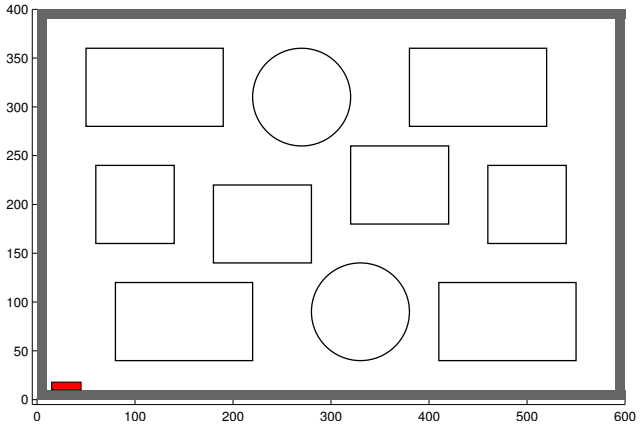
Layout	URLs of simulation video clips
3(a)	https://nclab.github.io/pso.evac/Scene1.avi
3(b)	https://nclab.github.io/pso.evac/Scene2.avi
3(c)	https://nclab.github.io/pso.evac/Scene3.avi
3(d)	https://nclab.github.io/pso.evac/Scene4.avi

layout is also shown in Figure 3. We put 50 particles in each round of simulation and record the number of iterations for each particle to reach the goal. For every layout, we simulate 1000 rounds and the average results are listed in Table II. The standard deviation represents the deviation of the average iterations among all simulation rounds. We also show the illustrations of the evacuation path generated by each particle in our simulation framework in Figure 4. The illustrations are interesting and may help user to understand the weakness of the layout and then to improve it. For dynamic simulation demonstration, one can find the video clips at the urls listed in Table III.

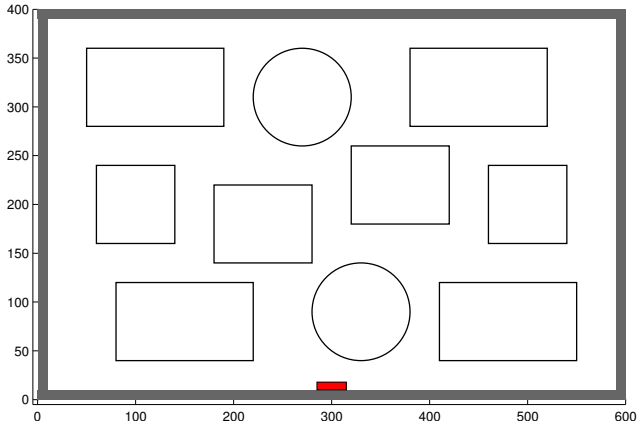
A. Comparison between different positions of exits

Many evacuation simulations are applied to evaluate and search for a better position to setup an exit. In our simulation framework, it can be achieved by changing the position of the target. The experimental results of 3(a) and 3(b) demonstrate this feature of our framework. In the two layouts, the position of exit in layout 3(b) should be better than layout 3(a), since the average distance from every position to the exit, regardless of obstacles, of layout 3(a) is larger than that of layout 3(b). The experimental results of our simulation framework exactly confirm this assumption. It shows that in layouts 3(a) and 3(b), the average iterations needed for evacuation are 133.6144 and 104.4127, respectively. Even in the worst case, layout 3(a) needs more iterations than layout 3(b).

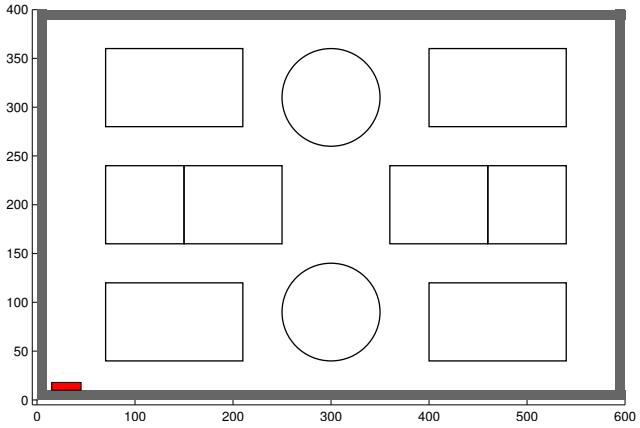
Looking into the simulation by observing the evacuation path generated by each particle in Figure 4(a) and Figure 4(b), we find that several particles would congest along with the edge of the obstacle near the exit in layout 3(a). In layout 3(b), fewer particles congest along the nearest round obstacle to the exit. It means that a rectangular obstacle would block the sight and the evacuation path more easily than a round obstacle



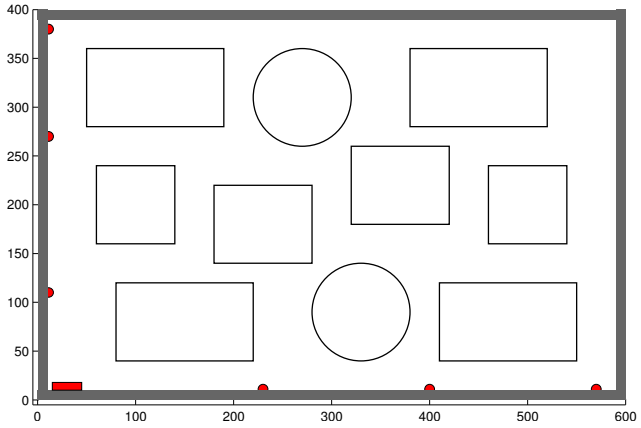
(a) The original room layout with the exit set at $(30, 10)$.



(b) The room layout with only the exit changed to $(300, 10)$.



(c) The room layout with obstacles rearranged. The sizes of obstacles and the location of exit are kept identical.



(d) The room layout with evacuation guide indicators located on the left and bottom walls.

Fig. 3. Experimental room layouts.

would. Also, comparing the space near exits, it can be found that exit of layout 3(b) has larger space which would help evacuate, too. Therefore, we can conclude that the placement of layout 3(b) is better based on intuitive comparison and simulation. The comparison demonstrates that our simulation can be used in such a problem to evaluate the appropriateness of the exit placement. It also verifies that our simulation framework can well agree with real evacuation events.

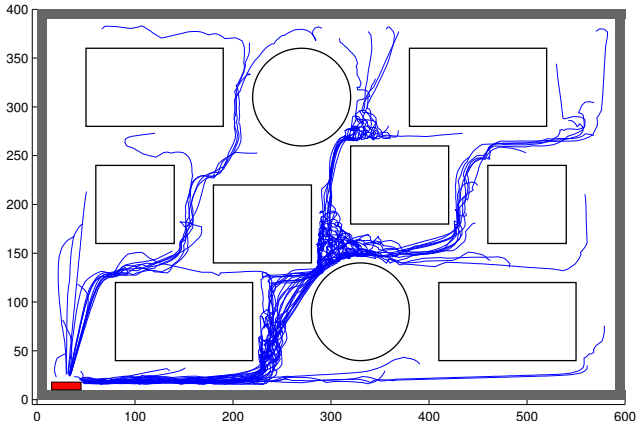
B. Comparison between different positions of obstacles

One of the evacuation simulation applications is to find a better placements of the furniture. For example, the department store sometimes rearranges their layout for allocating space for new brands or adjusting some brands to a different floor. In this situation, the position of exit is fixed and the manager would not want to rearrange into a placement which would block the evacuation path of customers. The experiment of comparison between layout 3(a) and layout 3(c) is presented to show that our simulation also can deal with this kind of problem. In layout 3(c), we place the obstacles more closely to each other than layout 3(a) to make the aisle wider. It would be helpful for evacuation if the aisle is wide enough to let many people go through. On the other hand, the wide aisle would have less possibility to block the sight of human. The simulation results confirm this rearrangement is better. The average and worst iterations needed to escape are 133.6144 and 330.423 in layout 3(a), which are 119.1571 and 250.513 in layout 3(c).

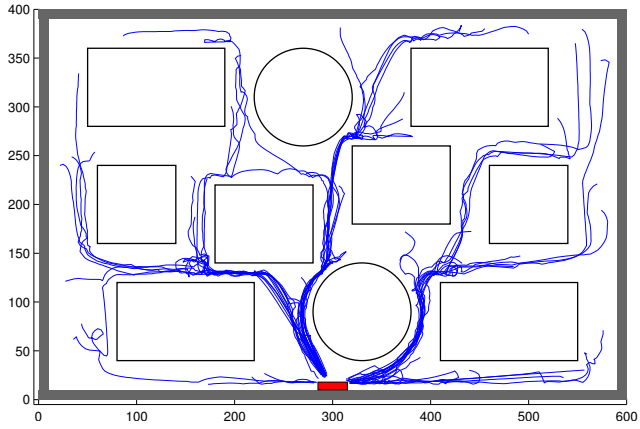
Investigating the evacuation paths in these layouts, it is obvious that the congestion caused by the obstacles in layout 3(a) is released. The wider aisle and more regular placement of obstacles enable people to pass more smoothly without idling around. It shows that the regularity also reduces the number of turns needed to reach an exit. Summing up these reasons, the placement of layout 3(c) is better and the simulation validates this assumption. It shows that our simulation framework can also be used to evaluate the placement of the obstacles.

C. Comparison for the presence evacuation guide indicators

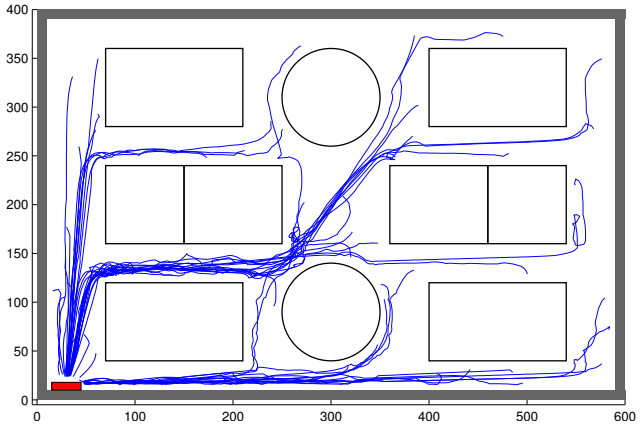
In addition to the features mentioned in previous sections, our simulation framework has the ability to simulate the evacuation environment with guide indicators. The guide indicators in evacuation are the useful information about the location of exit to lead people to get out of danger zones. It is commonly used in each building and is definitely helpful. However, the



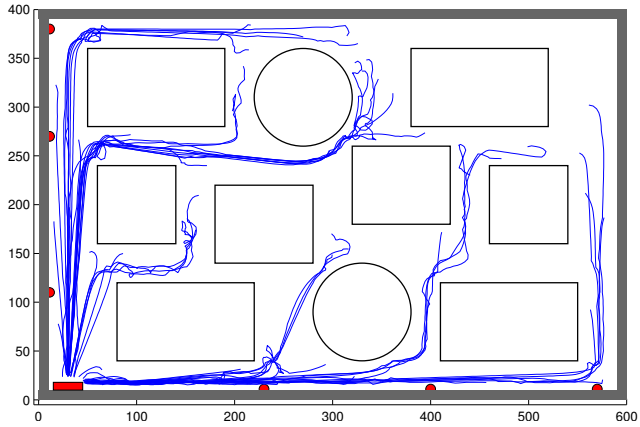
(a) The original room layout. The population needs 148.76 iterations in average and 327 iterations in the worst case to evacuate.



(b) Room layout with a different position of exit. The population needs 98.12 iterations in average and 194 iterations in the worst case to evacuate.



(c) Room layout with different positions of obstacles. The population needs 102.62 iterations in average and 218 iterations in the worst case to evacuate.



(d) Room layout with evacuation guide indicators. The population needs 112.38 iterations in average and 200 iterations in the worst case to evacuate.

Fig. 4. Evacuation paths generated by particles in each room layout are selected from one of the 1000 rounds of simulation.

position of the guide indicator is important. If a guide indicator is placed at a position invisible to people, it would be useless. In our simulation framework, we can model the guide indicator as a static particle with a better objective value. Any other particles that come into the sight of the guide indicator would therefore be updated a better global best information and then move toward it. By the help of the guide indicators, the particle would to be able find the way out more easily. The simulations of layout 3(a) and layout 3(d) show the benefit of the presence of guide indicators. The average and worst iterations used are 118.4122 and 247.086 in layout 3(d), whereas it needs 133.6144 iterations in average and 330.423 iterations in the worst case to evacuate in layout 3(a).

Observing the evacuation path generated in the simulation of layout 3(d) (Figure 4(d)), it can be found that the particles really are attracted by the indicators and make them be on the way out more quickly. By the evacuation guide indicators, particles would move to the nearest indicator and hence would not detour between the obstacles in the middle. This experiment shows that the proposed framework can also model the evacuation guide indicator without any additional modeling and the simulation confirms that the indicator indeed work under this modeling.

V. Conclusions

Since it is hard to verify the evacuation plan in real, simulation systems become important and indispensable for evaluating and testing the plan. Many studies have been developed for applying in various scenarios. Evolutionary algorithms are often used for finding the optimum solution to exit placement, evacuation route, or building layouts. For simulating human behavior, which is critical in any simulation systems, lot of studies apply particle swarm optimization (PSO) algorithm because PSO was designed with the inspiration of the movement of birds. The simulation models adopted in the literature are usually complicated, suitable for only specific scenes, and incompatible with each other, we proposed a PSO-based simulation framework in this paper which models all the factors needed in an evacuation scenario, such as human beings, obstacles, exits, and even guide indicators, by the

compositions of PSO. The users can easily set up their scenario and simulate by configuring parameters of PSO. The work is based on the proposal of controlling crowd movement by PSO in [8], [9]. In order to apply in an evacuation simulation and model the more realistic human behavior, we design some mechanisms such as local search and global optimum update to enhance the framework. For demonstrating the abilities of the simulation framework, we provide several scenarios and make the comparisons. The scenarios are designed such that we usually can judge which one is better to verify our simulation framework. By these scenarios, we can confirm the proposed framework simulates well since the simulation results match our assumption and intuition. Furthermore, the simulation framework also provides the ability to display the evacuation path which would be useful for improving the evacuation plan or the rearrangement.

In this paper, we proposed the PSO-based evacuation simulation framework that can be adopted in evaluating the placements of exits, obstacles, and even evacuation guide indicators. The modeling of human behavior in our framework is simple and useful. However, human behavior is complicated and easily affected by emotional and physical factors. For example, some researchers discussed the differences of moving pattern between normal human and elder or injured ones. They used two different groups of particle swarms with different maximum speeds for the modeling. This can also be adopted in our framework. It is a challenge to properly model complicated human behavior and we will continually pursue this line of research. On the other side, the modeling of diverse scenarios is also a promising research direction, such as modeling of multiple exits and spread of disaster area. The implementation of our proposed simulation framework is open source [10] for researchers interested in adopting in their study and practitioners interested in employing for evaluation.

ACKNOWLEDGMENTS

The authors are grateful to the National Center for High-performance Computing for computer time and facilities. The work was supported in part by the National Science Council of Taiwan under Grant NSC 101-2628-E-009-024-MY3.

REFERENCES

[1] A. Garrett, B. Carnahan, R. Muhdi, J. Davis, and G. Dozier, "Evacuation planning via evolutionary computation," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006, pp. 157–164.

[2] X. Zong, S. Xiong, Z. Fang, and Q. Li, "Multi-ant colony system for evacuation routing problem with mixed traffic flow," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2010, pp. 1–6.

[3] M. Yusoff, J. Ariffin, and A. Mohamed, "An improved discrete particle swarm optimization in evacuation planning," in *Proceedings of the IEEE International Conference of Soft Computing and Pattern Recognition*, 2009, pp. 49–53.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[5] C. Wang, B. Yang, L. Li, and H. Huang, "A modified particle swarm optimization-based human behavior modeling for emergency evacuation simulation system," in *Proceedings of the 2008 IEEE International Conference in Information and Automation*, 2008, pp. 23–28.

[6] H. Junaedi, M. Hariadi, and I. K. E. Purnama, "Multi agent with multi behavior based on particle swarm optimization (pso) for crowd movement in fire evacuation," in *Proceedings of the IEEE International Conference on Intelligent and Information Processing (ICICIP 2013)*, 2013, pp. 366–372.

[7] J. Kou, S. Xiong, H. Liu, and X. Zong, "Particle swarm and nsga-ii based evacuation simulation and multi-objective optimization," in *Proceedings of the IEEE International Conference on Natural Computation*, 2011, pp. 1265–1269.

[8] Y.-y. Lin and Y.-p. Chen, "Crowd control with swarm intelligence," in *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 3321–3328.

[9] Y.-p. Chen and Y.-y. Lin, "Controlling the movement of crowds in computer graphics by using the mechanism of particle swarm optimization," *Applied Soft Computing*, vol. 9, no. 3, pp. 1170–1176, 2009.

[10] P.-C. Tsai, C.-M. Chen, and Y.-p. Chen, "PSO Evacuation Framework Source Code," https://github.com/nclab/pso.evac, March 2014.