Density-based evolutionary framework for crowd model calibration

Zhong, Jinghui; Hu, Nan; Cai, Wentong; Lees, Micheal; Luo, Linbo

2015

Zhong, J., Hu, N., Cai, W., Lees, M., & Luo, L. (2015). Density-based evolutionary framework for crowd model calibration. *Journal of computational science*, 6, 11-22.

<https://hdl.handle.net/10356/107221>

<https://doi.org/10.1016/j.jocs.2014.09.002>

© 2014 Elsevier B.V. This is the author created version of a work that has been peer reviewed and accepted for publication by *Journal of Computational Science*, Elsevier B.V. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [Article DOI: <http://dx.doi.org/10.1016/j.jocs.2014.09.002>].

Downloaded on 26 Apr 2023 17:50:32 SGT

Density-Based Evolutionary Framework for Crowd Model Calibration

Jinghui Zhong^a, Nan Hu^b, Wentong Cai^a, Michael Lees^c, Linbo Luo^d

^a*School of Computer Engineering, Nanyang Technological University, Singapore*

^b*Institution of High Performance Computing, Agency for Science Technology and Research, Singapore*

^c*Section Computational Science, University of Amsterdam, Netherlands*

^d*School of Computer Science and Technology, Xidian University, Xi'an, China*

Abstract

Crowd modeling and simulation is an important and active research field, with a wide range of applications such as computer games, military training and [evacuation modeling](#). One important issue in crowd modeling is model calibration through parameter tuning, so as to produce desired crowd behaviors. Common methods such as trial-and-error are time consuming and tedious. This paper proposes an evolutionary framework to automate the crowd model calibration process. In the proposed framework, a density-based matching scheme is introduced. By using the dynamic density of the crowd over time, and a weight landscape to emphasize important spatial regions, the proposed matching scheme provides a generally applicable way to evaluate the simulated crowd behaviors. Besides, a hybrid search mechanism based on differential evolution is proposed to efficiently tune parameters of crowd models. Simulation results demonstrate that the proposed framework is effective and efficient to calibrate the crowd models in order to produce desired macroscopic crowd behaviors.

Keywords: Crowd modeling and simulation, density distribution, differential evolution, evolutionary algorithm, model calibration

---

1. Introduction

Crowd modeling and simulation has become an important and active research field recently, with applications in diverse areas such as computer games [1], military training [2], [and evacuation modeling](#) [3, 4, 5, 6]. Various modeling approaches such as the particle models [7, 8], flow-based models [9, 10], and agent-based models [11, 12, 13, 14, 15] have been proposed for crowd simulation.

Email addresses: jinghuizhong@gmail.com (Jinghui Zhong), hun@ihpc.a-star.edu.sg (Nan Hu), aswtcai@ntu.edu.sg (Wentong Cai), m.h.lees@uva.nl (Michael Lees), lblood@xidian.edu.cn (Linbo Luo)

A thorough survey on crowd modeling and simulation technologies can be found in [16].

One important issue in crowd modeling and simulation is model calibration, which aims to tune the parameters of a model so that the output of the model can match with observed data in real world (e.g., crowd behaviors extracted from videos). Once a crowd model has been well calibrated, it can then be used for prediction and “what-if” scenario analysis. However, as the simulated crowd behavior is affected indirectly and unintuitively by the parameter settings of the crowd model, finding the correct parameter settings to reproduce desired crowd behaviors is a challenging task. Typically modelers will manually tune parameters until an acceptable simulated output is obtained. This trial-and-error method is time consuming, tedious, and requires an experienced person to manipulate the calibration process. Therefore, developing an effective approach to automate the crowd model calibration process is of urgent need. This paper aims to address this crowd model calibration problem (CMCP). That is, given a crowd simulation model and some objective behaviors, such as those extracted from videos, we aim to find the correct parameter setting of the crowd model through an automatic mechanism so that the simulated output closely matches the objective behaviors.

The CMCP problem has been addressed previously and these approaches focus mainly on using Evolutionary algorithms (EAs) [17, 18] such as Genetic Algorithm (GA). EAs are metaheuristic optimization methods, which have been shown very effective to find near global solutions to complex optimization problems [19, 20]. A salient feature of EAs differing from traditional optimization techniques is that they require no explicit formulation of the problem but only a fitness function to evaluate solutions. The CMCP is just such a problem for which it is hard to formulate the relationship between inputs and outputs. Besides, as EAs incorporate randomness in the search process, they are capable of avoiding stagnation in local optima. Therefore, EAs are suitable to solve the CMCP.

However, there are some drawbacks of the existing methods. Existing methods focus mainly on matching crowd behaviors at a microscopic level (e.g., trajectories of pedestrians). Nevertheless, collecting microscopic data of dense crowd is still challenging in practice [21]. Besides, in some cases, the individual’s behaviors are highly dynamic and stochastic, matching these behaviors is usually infeasible. In addition, EAs are population-based stochastic search algorithms. They require a large number of fitness evaluations during the evolution process. For crowd model calibration, each fitness evaluation requires a few simulation runs. Therefore, using EAs to solve CMCP requires a large number of simulation runs, which can be very time consuming. It is still highly desirable to develop an efficient mechanism to reduce the computational time.

To tackle the above drawbacks, this paper proposes an effective and efficient evolutionary calibration framework. [The proposed framework features a novel density-based matching scheme that is used for fitness evaluation.](#) In the proposed density-based matching scheme, the dynamics of crowds are captured by the density distribution of the crowds over a time period. A weight vector is

utilized to scale the density distribution so that more important regions, where a highly accurate match is essential (e.g., around a doorway in evacuation scenarios), can play more important roles in the similarity measurement. By calculating the distance between the scaled density distribution, the similarity between a simulated crowd behavior and an objective crowd behavior can be measured at a macroscopic level. The proposed density-based matching scheme is easy to implement and can be applied to different crowd scenarios such as the emergency evacuation scenarios. With this matching scheme, the proposed framework can automatically adjust control parameters so that certain desired collective behavior can emerge. Thus, our framework is useful for the development of crowd management strategies and can be used in conjunction with other work in the area of evacuation modeling and analysis [22, 23, 24, 25].

To improve the search efficiency, the proposed framework makes use of a hybrid search mechanism based on differential evolution (DE). DE is a popular and powerful EA which was proposed by Storn and Price [26] for global optimization. It has been shown to perform much better than several other EAs such as GA on a wide variety of problems [20]. To accelerate the search, the proposed EA framework incorporates an “Exploration Module” and an “Exploitation Module”. The “Exploration Module” focuses on global search, while the “Exploitation Module” focuses on refining the best-so-far solution to accelerate the search. In addition, the proposed framework tunes the parameter setting of a given crowd model based on multiple objective behaviors (i.e., training cases). In this way, more general solutions can be generated for prediction and the overfitting problem can be avoided. The proposed framework has been evaluated through the calibration of two modified social-force models. The simulation results demonstrate the effectiveness and efficiency of the proposed framework¹.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 describes the proposed density-based evolutionary framework for crowd model calibration. Section 4 presents the simulation studies. At last, Section 5 draws the conclusions.

2. Related Work

The commonly used methods for automatic model calibration are mainly based on the gradient-based methods [27], Nelder-Mead [28], and linear regression [29]. These deterministic search methods can provide solutions quickly, but they may get trapped into local optima easily [30].

Recently, stochastic search methods such as GA has become popular to calibrate computer models [17, 31, 32, 33]. Take Agent-Based Models (ABMs) for example, Calvez and Hutzler [34, 35] proposed to consider the parameter estimation of an ABM as an optimization problem and suggested using GAs to solve the problem. Stonedahl and Wilensky [36] incorporated a GA into a

---

¹Related simulation videos can be found at <http://crowds.sce.ntu.edu.sg/index.php/research/dbmc>.

software tool named “Behavior Search” to calibrate the parameters of two agent-based flocking models. Smith [37] applied a GA to calibrate the parameters of an agent-based cowbird model.

As for automatic crowd model calibration (i.e., the CMCP), the research is still at an early stage due to the difficulties in automatic evaluation of simulated crowd behaviors. The dominant methods to evaluate simulated crowd behaviors are based on “look and feel” [38]. Johansson et al. [17] proposed an EA to calibrate the parameters of a classical crowd model: the social force model. They utilized the microscopic motions of pedestrians such as the moving speed and direction to evaluate the simulated crowd behaviors. The main difference between our work and the above work is that our aim is to match crowd behaviors at a macroscopic level (in terms of density distribution), while the above work focuses on matching crowd behaviors at a microscopic level (in terms of trajectories of pedestrians). Besides, our work utilizes a specific mechanism to reduce the computational time. Wolinski et al. [18] proposed a general framework to compare different crowd models. The authors suggested using EAs to calibrate the crowd models before comparing their performances. The authors also listed some feasible metrics for evaluation such as the path length metric. However, the effectiveness of the framework is investigated mainly based on simple cases (e.g., collision avoidance behaviors), while the discussion on using EAs to calibrate complex crowd models (e.g., those contain high-level behavior rules) still remains at a conceptual level.

Some data-driven methods have been proposed to generate realistic crowd behaviors. For example, Lee et al. [21] made use of historical video data to train state-actions of agents so as to generate realistic crowd behaviors. Lerner et al. [39] proposed to control the motions of autonomous agents by searching examples in historical data that closely match the situation that they are facing. Ju et al. [40] used small examples in historical data to construct larger scale crowd behaviors. The main difference between our work and the above work is that our work focuses on developing an effective and efficient framework to calibrate a given crowd model, while the above works focus on generating realistic crowd behaviors.

3. The Proposed Density-based Evolutionary Framework for Crowd Model Calibration

3.1. The General Framework

To apply an EA to crowd model calibration, three important issues need to be addressed. The first issue is to define a general and effective fitness function to measure the distance between a simulated crowd behavior and the objective crowd behavior. The second issue is to reduce the computational time, because EAs require a large number of fitness evaluations which can be very expensive (especially for large scale scenarios). The third issue is to obtain robust solutions for avoiding the over-fitting problem. That is, the calibrated crowd model is not only able to reproduce the specific given objective behavior, but also able

to correctly predict unknown behaviors that come from the same crowd. For example, when the initial positions of pedestrians in an objective behavior are changed, it is desired that the crowd model is able to predict the crowd behavior with the modified initial conditions. To solve these problems, we propose a density-based evolutionary framework as shown in Fig. 1.

Figure 1: The flowchart of the proposed calibration framework.

Specifically, the proposed calibration framework consists of four modules. The first module is the *Preprocessing module* which aims to obtain a certain number of training and testing cases. The training and testing cases can be extracted from video data or other data sources. The training cases are used for training the parameters of the crowd model, while the testing cases are used to evaluate whether the calibrated model is good enough.

After the training cases and testing cases are generated, an enhanced Differential Evolution with Global and Local search modules (*GL-DE training module*) is used to calibrate the model, so that the calibrated crowd model well fits to the training cases. We choose a well-known EA variants (that is,

Table 1: The basic notations used in the paper.

Name	Summary
$\rho(\chi, t)$	The local density value of location χ at time step t .
$\varrho(\chi, t)$	The desired local density value of location χ at time step t .
C	The total number of agents in the entire region.
R	A constant to determine the impacts of nearby agents on the density value.
t	The current time step.
$r_i(t)$	the position of the i -th pedestrian at time step t .
W	The number of discrete grids in X axis.
L	The number of discrete grids in Y axis.
ξ_i	The i -th representative point.
T	The maximum time step.
$\mathbf{M}$	The density matrix to represent a crowd behavior.
ω_i	The weight assigned to ξ_i .
X	A solution, i.e., a configuration of the crowd model.
O_i	The i -th training objective behavior.
θ_i	The i -th testing objective behavior.
K_{train}	The total number of training cases.
K_{test}	The total number of testing cases.
$f(X)$	The fitness value of X computed by using K_{train} training cases.
$\varphi(X)$	The fitness value of X computed by using K_{test} testing cases.
$X_i^{(g)}$	The i -th target vector at generation g .
$x_{i,j}^{(g)}$	The j -th variable of $X_i^{(g)}$.
$Y_i^{(g)}$	The i -th mutant vector at generation g .
$y_{i,j}^{(g)}$	The j -th variable of $Y_i^{(g)}$.
$U_i^{(g)}$	The i -th trial vector at generation g .
$u_{i,j}^{(g)}$	The j -th variable of $U_i^{(g)}$.
F	The scale factor of the mutation operator of DE.
CR	The crossover rate of DE.
n	The number of parameters to be optimized.
NP	The number of new solutions generated by the framework in each generation.
p	The proportion of new solutions generated by the exploration module in each generation.
LB_i	The lower bound of the i -th variable.
UB_i	The upper bound of the i -th variable.

DE) to accomplish the training task. The reasons are as follows. First, as will be described later, DE adopts a one-to-one selection strategy in the selection process. This mechanism is good for capturing multiple local optima simultaneously. Second, the mutation operation of DE uses the difference of the current population to generate offspring. As the difference vectors contain no bias towards any special search direction, the mutation operation is good for maintaining the population diversity and avoiding local stagnation. Last but not least, as the difference vectors are calculated based on the current population, they would gradually shrink as the population converges. Thus the difference vectors can automatically adapt to the objective function landscape and achieve certain local fine-tuning effects [20]. To further reduce the computational time and accelerate the search, the proposed *GL-DE training module* uses a small population size and incorporates an “Exploration Module” and an “Exploitation Module”. The “Exploration Module” focuses on global search, while the “Exploitation Module” focuses on fine-tuning the best-so-far solutions.

During the training process, the third module named *density-based fitness evaluation module* would work in cooperation with the *GL-DE training module* to accomplish the training task. The objective of the *density-based fitness evaluation module* is to evaluate the fitness values of solutions given by the *GL-DE training module*. As will be described later, the *density-based fitness evaluation module* makes use of the density distribution of crowds to measure the similarity between simulation crowd behaviors and objective behaviors. This module would also work in cooperation with the *Validation module* to accomplish the calibration process.

The fourth module uses testing cases to validate the calibrated model. Its objective is to test whether the simulated outputs of the calibrated model fit well to the testing cases. If the testing results are not satisfactory, then the proposed framework would return to the *PreProcessing Module* to revise the training and testing cases. Otherwise, the framework outputs the best solution.

In the following parts, we describe the implementation of the *density-based fitness evaluation module* and the *GL-DE training module*. The basic notations of the algorithm framework are listed in Table 1.

3.2. Density-based Fitness Evaluation Module

3.2.1. Density Distribution

The density of a crowd is an intuitive and important feature of crowd behaviors [41]. The dynamic density distribution over time reflects macroscopic crowd dynamics. Therefore, we propose to use the density distribution to characterize the macroscopic crowd behaviors. That is, for each time step, the entire crowd behavior is represented by the corresponding density distribution. Fig. 2 shows an example of representing the crowd behavior using a series of density distributions.

Specifically, the local density value of the crowd at location χ at time step t

Figure 2: An example of crowd behavior representation using a series of density distributions.

can be computed as:

$$\rho(\chi, t) = \sum_{i=1}^C \frac{1}{\pi R^2} \exp[-\|r_i(t) - \chi\|^2 / R^2] \quad (1)$$

where C is the number of pedestrians in the observed region; $r_i(t)$ is the position of the i -th pedestrian and R is the kernel radius (e.g., $R = 2\text{m}$). The greater R , the smaller variance of the local density values. If $R \rightarrow \infty$, all local density values are equal to the overall number of pedestrians divided by the area they are distributed in [42]. To estimate the density distribution of the crowd, we evenly divide the observed region into $W \times L$ discrete grids. Obviously the resolution of the grid can be adapted to the problem, with higher resolution providing a larger search space for the EA. The center points of the discrete grids are used as representative points for estimating the density distribution of the crowd at time step t . Denote the set of representative points as $\{\xi_1, \xi_2, \dots, \xi_{W \times L}\}$. The dynamic crowd behaviors can then be represented by:

$$\mathbf{M} = \left[\begin{bmatrix} \rho(\xi_1, 1) \\ \rho(\xi_2, 1) \\ \dots \\ \rho(\xi_{W \times L}, 1) \end{bmatrix}, \dots, \begin{bmatrix} \rho(\xi_1, T) \\ \rho(\xi_2, T) \\ \dots \\ \rho(\xi_{W \times L}, T) \end{bmatrix} \right] \quad (2)$$

where T is the number of time steps or frames to be used in the calibration.

There are several advantages of using density distribution to describe the dynamic crowd behaviors. First, the density distribution is easy to compute and there are various techniques available in the literature for crowd density estimation [43]. Hence, using density distribution is convenient for practical applications. Second, the density distribution is general to describe crowd behaviors, as it does not depend on specific crowd models or scenarios.

3.2.2. Distance Measure

Let $\rho(\xi_i, t)$ be the density value at ξ_i and time step t of a simulated crowd behavior and $\varrho(\xi_i, t)$ be that of the objective crowd behavior, the distance between these two behaviors can be computed by

$$distance = \sum_{t=1}^T \left(\sum_{i=1}^{W \times L} (|\rho(\xi_i, t) - \varrho(\xi_i, t)|) \right) \quad (3)$$

where T is the total number of time steps; $W \times L$ is the total number of representative points.

In some applications, the crowd behaviors at certain spatial regions are more important (e.g., at a doorway or stairway during an evacuation). To allow specific spatial emphasis for different applications, we use a predefined weight vector to scale the density value. The weight vector can be expressed as:

$$[\omega_1, \omega_2, \dots, \omega_{W \times L}] \quad (4)$$

where ω_i is the weight assigned to ξ_i . Important representative points will have larger weight values. Fig. 3 shows an example landscape of the weight values where the crowd behaviors around the gate will play a more important role in the distance measurement.

Figure 3: An example landscape of weight values

Given an objective crowd behavior O and a simulated crowd behavior S , their distance can be computed by:

$$D(S, O) = \sum_{t=1}^T \sum_{i=1}^{W \times L} (\omega_i \cdot |\rho(\xi_i, t) - \varrho(\xi_i, t)|) \quad (5)$$

where $\rho(\xi_i, t)$ and $\varrho(\xi_i, t)$ are the density values of S and O respectively.

To avoid the over-fitting problem, we use K_{train} training cases to evaluate the fitness of each solution X . Each training case is a unique objective behavior

that has the same features as the other training cases, but with different initial conditions (e.g., the number of pedestrians and the initial positions of pedestrians). For example, to calibrate a crowd model that simulates people entering a building, the K_{train} training cases can be extracted from K_{train} videos, each of which records people entering the same building in different time periods. Based on the K_{train} training cases, the fitness value of X is evaluated by:

$$f(X) = \frac{\sum_{i=1}^{K_{train}} D(S_i, O_i)}{K_{train}} \quad (6)$$

where O_i are the i -th training cases, and S_i is the simulated behavior where the crowd model is configured by X and the initial conditions are set the same as O_i .

At the last state of the algorithm, the qualities of the solutions found by the algorithm need to be validated using K_{test} testing cases. The testing cases are generated in the same way as generating the training cases. The quality of solution X is calculated by:

$$\varphi(X) = \frac{\sum_{i=1}^{K_{test}} D(S_i, \theta_i)}{K_{test}} \quad (7)$$

where θ_i is the i -th testing case, and S_i is the simulated behavior where the crowd model is configured by X and the initial conditions are set the same as θ_i . To facilitate description, we called $f(X)$ and $\varphi(X)$ as training fitness value and testing fitness value respectively.

3.3. The GL-DE Training Module

This subsection describes the implementation of the *GL-DE training module*. Generally, the *GL-DE training module* generates NP new solutions in each generation to maintain a population of $NP * p$ individuals. Here $p \in (0, 1]$ is a controlling parameter and $NP * p$ is the size of the population. In each generation, the “Exploration Module” first generates $NP * p$ new solutions to update the $NP * p$ individuals, by using DE operations that are suitable for global search. Then, the “Exploitation Module” generates $NP * (1 - p)$ new solutions to refine the best individual found in the “Exploration Module”, by using DE operations that are suitable for local search. The implementation details of the *GL-DE training module* are as follows.

3.3.1. Initialization

The first step is to generate NP random solutions. Each solution is represented by a vector of floating-point numbers (called target vector):

$$X_i^{(g)} = [x_{i,1}^{(g)}, x_{i,2}^{(g)}, \dots, x_{i,n}^{(g)}] \quad (8)$$

where g is the generation and n is the dimension of the problem (i.e., the number of parameters to be evolved). The NP random solutions are generated by

$$x_{i,j}^{(1)} = rand(LB_j, UB_j) \quad (9)$$

ALGORITHM 1: THE GL-DE TRAINING MODULE.

Output: The best-so-far solution X_{best}

$g = 1$

for $i = 1$ **to** NP **do**

for $j = 1$ **to** n **do**

$x_{i,j}^{(g)} = rand(LB_j, UB_j);$

Evaluate all initial solutions;

X_{best} = the best of the NP solutions;

Update the best of the first $NP * p$ solutions by X_{best} ;

while $g \leq MAXGENS$ **do**

for $i = 1$ **to** $NP * p$ **do**

$F = rand(0, 1); CR = rand(0, 1);$

        Generate three random indices r_1, r_2, r_3 , with $i \neq r_1 \neq r_2 \neq r_3$

        and $1 \leq r_1, r_2, r_3 \leq NP * p$;

$k = rand\_int(1, n);$

for $j = 1$ **to** n **do**

if $rand(0, 1) < CR$ **or** $j = k$ **then**

$u_{i,j}^{(g)} = x_{r_1,j}^{(g)} + F \cdot (x_{r_2,j}^{(g)} - x_{r_3,j}^{(g)});$

else

$u_{i,j}^{(g)} = x_{i,j}^{(g)};$

if $f(U_i^{(g)}) \leq f(X_i^{(g)})$ **then**

$X_i^{(g+1)} = U_i^{(g)};$

if $f(U_i^{(g)}) \leq f(X_{best})$ **then**

$X_{best} = U_i^{(g)};$

else

$X_i^{(g+1)} = X_i^{(g)};$

for $i = NP * p + 1$ **to** NP **do**

$F = gaussian(0, 1); CR = rand(0, 1);$

        Generate two random indices r_1, r_2 with $r_1 \neq r_2$ and

$1 \leq r_1, r_2 \leq NP * p$;

$k = rand\_int(1, n)$

for $j = 1$ **to** n **do**

if $rand(0, 1) < CR$ **or** $j = k$ **then**

$x_{i,j}^{(g+1)} = x_{best,j} + F \cdot (x_{r_1,j}^{(g+1)} - x_{r_2,j}^{(g+1)});$

else

$x_{i,j}^{(g+1)} = x_{best,j};$

if $f(X_i^{(g+1)}) \leq f(X_{best})$ **then**

$X_{best} = X_i^{(g+1)};$

Update the best individual (i.e., the best of the first $NP * p$ solutions)

by X_{best} ;

$g = g + 1;$

return $\{X_{best}\}$

---

where LB_j and UB_j are the lower and upper bounds of the j -th parameter, $rand(a, b)$ returns a real value uniformly distributed within a and b .

After generating the variable values of the NP random solutions, their fitness values are evaluated by (6). The best of the NP solutions is copied as the best-so-far solution denoted as X_{best} . Then the first $NP * p$ solutions are regarded as the initial population, and the best individual will be updated by X_{best} if its fitness is worse than that of X_{best} .

3.3.2. Exploration Module

In the second step, the $NP * p$ individuals are updated using the DE search mechanism. The DE operators used in this step should be good at maintaining the population diversity, so as to guarantee the global search ability of the algorithm.

First of all, a mutation operation is performed on each target vector to create a mutant vector. In the DE community, there are various mutation schemes, each of which is suitable for solving specific kinds of problems. Here the commonly used mutation scheme named “DE/rand/1” (as expressed in Eq.(10)) is utilized to create a mutant vector:

$$Y_i^{(g)} = X_{r_1}^{(g)} + F \cdot (X_{r_2}^{(g)} - X_{r_3}^{(g)}) \quad (10)$$

where F is the scaling factor, r_1, r_2, r_3 and i are four distinct solution indices (i.e., $i \neq r_1 \neq r_2 \neq r_3$ and $1 \leq i, r_1, r_2, r_3 \leq NP * p$). We use the “DE/rand/1” mutation scheme in the *Exploration Module*, because it has been shown to have strong global search ability [44].

After that, each target vector $X_i^{(g)}$ is crossed with its mutant vector $Y_i^{(g)}$ to create a trial vector $U_i^{(g)}$:

$$u_{i,j}^{(g)} = \begin{cases} y_{i,j}^{(g)}, & \text{if } rand(0, 1) < CR \quad \text{or} \quad j = k \\ x_{i,j}^{(g)}, & \text{otherwise} \end{cases} \quad (11)$$

where CR is the crossover rate, k is a random integer between 1 and n , $u_{i,j}^{(g)}, y_{i,j}^{(g)}$ and $x_{i,j}^{(g)}$ are the j -th variables of $U_i^{(g)}, Y_i^{(g)}$ and $X_i^{(g)}$ respectively. Finally, the better one between each pair of the target and trial vectors is chosen as a new target vector:

$$X_i^{(g+1)} = \begin{cases} U_i^{(g)}, & \text{if } f(U_i^{(g)}) \leq f(X_i^{(g)}) \\ X_i^{(g)}, & \text{otherwise} \end{cases} \quad (12)$$

Here $f()$ is the fitness function defined in (6). If the new target vector $X_i^{(g+1)}$ is better than X_{best} , then X_{best} will be updated by $X_i^{(g+1)}$ (i.e., X_{best} always equals to the best individual).

To improve the robustness of the algorithm, the values of F and CR are randomly set at the beginning of each generation, i.e.,

$$F = rand(0, 1) \quad (13)$$

$$CR = rand(0, 1) \quad (14)$$

3.3.3. Exploitation Module

In the *Exploitation Module*, $NP \cdot (1-p)$ new solutions are generated using DE operators which are good at local fine tuning. The goal of the exploitation step is to refine the best individual found in the *Exploration Module*. Specifically, the exploitation module generates $NP \cdot (1-p)$ solutions surrounding X_{best} using the traditional “DE/best/1” mutation scheme and the crossover operation. The “DE/best/1” mutation scheme is adopted due to its good local search ability [44].

First of all, the “DE/best/1” mutation scheme and the crossover operation are performed on the X_{best} to generate a neighboring solution:

$$x_{i,j}^{(g+1)} = \begin{cases} x_{best,j} + F \cdot (x_{r1,j}^{(g+1)} - x_{r2,j}^{(g+1)}), & \text{if } rand(0, 1) < CR \text{ or } j = k \\ x_{best,j}, & \text{otherwise} \end{cases} \quad (15)$$

where r_1 and r_2 are two solution indices with $r_1 \neq r_2$ and $1 \leq r_1, r_2 \leq NP \cdot p$. In the exploitation module, the scaling factor F is set as

$$F = gaussian(0, 1) \quad (16)$$

where $gaussian(0, 1)$ returns a random value with standard normal distribution. The crossover rate CR is set by (14).

Once a neighboring solution is generated, X_{best} will be updated by it if its fitness value is better than that of X_{best} :

$$X_{best} = \begin{cases} X_i^{(g+1)}, & \text{if } f(X_i^{(g+1)}) \leq f(X_{best}) \\ X_{best}, & \text{otherwise} \end{cases} \quad (17)$$

After refining X_{best} based on the $NP \cdot (1-p)$ neighboring solutions, the best individual found in the *Exploration Module* is updated by X_{best} .

The exploration module and the exploitation module are executed repetitively until the maximum number of generations is reached. The pseudocode of the *GL-DE Training Module* is described in Algorithm 1.

4. Simulation Studies

In this section, the proposed density-based evolutionary framework is used to calibrate two crowd simulation models. First, for completeness, we present the formal description of the social force model used. Then, two modified social force models are designed to simulate a crowd riot scenario and a street crossing scenario respectively, and the performances of the proposed framework on calibrating the two crowd models are discussed.

4.1. Social Force Model

The social force model proposed by Helbing et al. [45] is a crowd simulation model which uses virtual forces to guide pedestrian motions. Specifically, the force imposed on an entity is expressed as:

$$f_i = m_i \frac{dv_i}{dt} = f_{i0} + \sum_{j(j \neq i)} f_{ij} + \sum_w f_{iw} \quad (18)$$

where m_i and $\frac{dv_i}{dt}$ are the mass and acceleration rate of the pedestrian; f_{i0} , f_{ij} and f_{iw} are attractive force towards the goal, repulsive force from other pedestrians and repulsive force from the static obstacles (e.g., wall) respectively.

The attractive force is computed by

$$f_{i0} = m_i \cdot \frac{v_i^0(t)e_i^0(t) - v_i(t)}{\tau_i} \quad (19)$$

where $v_i^0(t)$ is a certain desired speed and $e_i^0(t)$ is the direction towards the goal; τ_i is a characteristic time for the pedestrian to adapt the actual velocity. The repulsive force from other pedestrians is computed by

$$f_{ij} = \left[A \cdot \exp\left(\frac{r_{ij} - d_{ij}}{B}\right) + k \cdot g(r_{ij} - d_{ij}) \right] \cdot \mathbf{n}_{ij} + \kappa \cdot g(r_{ij} - d_{ij}) \Delta v_{ji}^t \cdot \mathbf{t}_{ij} \quad (20)$$

where d_{ij} is the distance between two pedestrians $\mathbf{r}_i$ and $\mathbf{r}_j$, and $r_{ij} = r_i + r_j$ is the sum of their radiuses; A, B, k and κ are four predefined constants. Function $g(x)$ returns zero if x is smaller than zero, otherwise it is equal to x ; $\mathbf{n}_{ij} = (n_{ij}^1, n_{ij}^2) = (\mathbf{r}_i - \mathbf{r}_j)/d_{ij}$ is the normalized vector pointing from $\mathbf{r}_j$ to $\mathbf{r}_i$. $\mathbf{t}_{ij} = (-n_{ij}^2, n_{ij}^1)$ is the tangential direction and $\Delta v_{ji}^t = (v_j - v_i) \cdot \mathbf{t}_{ij}$ is the tangential velocity difference. Similarly, the repulsive force from the wall can be computed by

$$f_{iw} = \left[A \cdot \exp\left(\frac{r_i - d_{iw}}{B}\right) + k \cdot g(r_i - d_{iw}) \right] \cdot \mathbf{n}_{iw} + \kappa \cdot g(r_i - d_{iw}) (\mathbf{v}_i \cdot \mathbf{t}_{iw}) \mathbf{t}_{iw} \quad (21)$$

where d_{iw} is the distance to a static obstacle such as a wall, n_{iw} and t_{iw} are the direction perpendicular and tangential to it respectively.

The social force model contains four parameters, which are A, B, k , and κ . In the following parts, two modified social force models are designed for testing. In both models, all four parameters of the social force model need to be optimized. The lower bounds (LB), the upper bounds (UB), and the default values (dv) of the four parameters are listed in Table II. The default values are set to the suggested values in [45], while the lower and upper bounds are set empirically.

4.2. Case 1: Crowd Riot Scenario

4.2.1. Crowd model definition

In this case study, we consider a crowd riot scenario as shown in Fig. 4. A group of pedestrians are scattering in a 15m $\times$ 12m rectangle region. An

Table 2: Four Parameters of The Social Force Model.

Name	Description	LB	UB	dv
A	Reflects the strength of interaction	1000	5000	2000
B	Determines the interaction range	0.01	1	0.08
k	Determines the obstruction effects of physical interactions	100000	300000	120000
κ	Determines the obstruction effects of physical interactions	10000	300000	240000

Figure 4: The crowd riot scenario for simulation study

instigator is located at the left while a gate is located at the right. We design a modified social-force model to simulate the crowd movements. The model contains 12 parameters. Our objective is to apply the proposed framework to tune these 12 parameters so as to produce desired objective behaviors. The meanings of these parameters and related assumptions of the model are described as follows:

- (1) The motions of pedestrians are determined by the social force model and the initial goals of pedestrians are to reach the instigator.
- (2) When a pedestrian is within a range of l to the instigator, its anger level will increase by an amount of $\Delta\alpha$.
- (3) The initial anger level of all pedestrians is set to be zero. When a pedestrian's anger level reaches a threshold M' , it will change its goal towards the gate and its desired speed to be its maximum speed.
- (4) When there are at least n' neighbors whose anger levels are larger than M' , the pedestrian's anger level will also increase by an amount of $\Delta\alpha$.
- (5) The anger level of each pedestrian will decay with time and we denote the decay coefficient as δ .
- (6) There is a compound attractive-repulsive force (f_{TRi}) from the gate assigned to each pedestrian, which is related to the anger level (α_i) and the distance to

the gate (d_{ig}). Based on this, the total social force calculated for each agent in Eq. (18) thus becomes:

$$f_i = m_i \frac{dv_i}{dt} = f_{i0} + \sum_{j(j \neq i)} f_{ij} + \sum_w f_{iw} + f_{TRi} \quad (22)$$

where f_{TRi} is computed by

$$f_{TRi} = \left[A_2 \cdot \exp\left(\frac{d_{max} - d_{ig}}{B_2}\right) - \omega \cdot \alpha_i \right] \cdot \mathbf{g}_i \quad (23)$$

where d_{max} is a constant distance and $\mathbf{g}_i$ is a unit vector, pointing from the gate to the pedestrian. If d_{ig} is small and the anger level is small, f_{TRi} is positive, which simulates a repulsive force from the gate. Otherwise, the f_{TRi} is negative and simulates an attractive force from the gate.

(7) When a pedestrian reaches close enough to the gate (i.e., $d_{ig} < 1m$), the attractive-repulsive force from gate will disappear (i.e., $f_{TRi} = 0$).

Table 3: Scenario Specific Parameters of The Crowd Riot Scenario

Name	Description	LB	UB	dv
l	The sensible range of each pedestrian	1	5	3
$\Delta\alpha$	The amount of anger added at each step	0.1	5	2
M'	The maximum anger beyond which a pedestrian will change its goal to the gate and run at maximum speed	0	2	0.5
δ	The decay rate of anger	0.1	0.99	0.95
n'	The number of anger neighbors beyond which the pedestrian's anger level would increase by $\Delta\alpha$	1	10	5
A_2	Determines the strength of repulsive force in (23)	100	10000	200
B_2	Determines the interaction range of repulsive force in (23)	1	30	15
ω	Reflects the strength of attractive force in (23)	1	30	20

4.2.2. Parameter settings

The proposed framework is used to calibrate the four social force parameters and the eight scenario specific parameters (as listed in Table 3). We generate five training objective behaviors and ten testing objective behaviors. Each of the fifteen objective behaviors is generated by running the same simulation model with the same parameter values (i.e., the default values in Table 2 and Table 3), but the initial positions of individuals in each objective behavior are different. We try to calibrate the twelve parameters based on the five training

Table 4: Parameter Settings of GA, DE and GL-DE.

Algorithm	Parameter settings
GA	$NP = 20$; $mutationrate = 0.1$; $crossoverrate = 0.7$; maximum generation = 100.
DE	$NP = 20$; $F = 0.5$; $CR = 0.9$; maximum generation = 100.
GL-DE	$NP = 20$; $p = 0.75$; maximum generation = 100.

objective behaviors. Then the ten testing objective behaviors are used to test the robustness of the solutions.

In each simulation, the number of pedestrians is set to be 200, the maximum time step is set to be 1000. The value of d_{max} , the maximum speed V_{max} and the original desired speed V_0 are set to be: 16m, 2.5m/s, and 1m/s, respectively. The entire region is evenly divided into 18×12 grids, and the center of each grid is used as a representative point for estimating the density distribution. The model is implemented using Repast HPC [46]. We are more interested in the crowd behaviors around the gate. Hence we assign larger weights to those regions. Here, the weight values in (5) is computed by

$$w_i = \exp(-d_i/5) \quad (24)$$

where d_i is the distance from ξ_i to the gate. The landscape of the weight vector is shown in Fig. 2.

To investigate the effectiveness and efficiency of the *GL-DE training module*, we replace it with two other EAs to form two new calibration methods. The first EA is the genetic algorithm (GA) which is one of the most popular algorithm for model calibration. The second EA is the standard DE. We compare *GL-DE training module* with DE to validate the effectiveness of the hybrid search mechanism. The parameter settings of GA, DE and GL-DE are listed in Table 4. All algorithms terminate when the maximum generation (100) is reached (i.e., the maximum number of fitness evaluations (FES) is $100 * NP = 2000$).

4.2.3. Simulation Results

GA, DE and GL-DE are carried out for 30 independent runs with different random seeds and the results of the 30 runs are used for analysis. In each run, we record the best-so-far training fitness value found by the algorithm at each fitness evaluation. Fig. 5 shows the average of the best-so-far training fitness values versus the number of fitness evaluations. It can be observed that DE generally performs better than GA, because it converges faster and the final training fitness values are better than those found by the GA in average. Meanwhile, the final training fitness values found by GL-DE and DE are very similar, but GL-DE converges much faster than DE.

Table 5 shows the results of the final solutions found by the algorithms in the 30 independent runs, where “Training” represents the average of the training fitness values of the final solutions, and “Testing” represents the average of the testing fitness values of the final solutions. We also use two-sample t-test to

Figure 5: Average of the best fitness values versus the number of fitness evaluations.

Table 5: Simulation results of the riot scenario with FES = 2000.

Algorithm	Training	P value	Testing	P value
GA	360.9 ± 31.3	$4.4\text{E-}10\ddagger$	463.5 ± 36.7	$2.1\text{E-}4\ddagger$
DE	317.4 ± 10.9	$0.93\S$	416.6 ± 21.9	$0.1\S$
GL-DE	317.2 ± 9.79	N/A	428.2 ± 32.2	N/A

Note: FES is the maximum number of fitness evaluation; $\ddagger$ and $\S$ mean the results are significant worse than (i.e., P value < 0.05), similar to those of GL-DE respectively, according to two-sample t-test at 0.05 level.

check whether the 30 results found by GA (or DE) are significantly different from those found by the proposed GL-DE. The results show that the results of GA are significantly worse than those of GL-DE, in both the training cases and testing cases. Meanwhile, the results of DE and GL-DE are not significantly different.

The results in Fig. 5 show that GL-DE generally converges faster than DE. This feature is useful when the calibration process only allows a small number of fitness evaluations since the crowd simulation is very expensive. To validate this, we record the 30 best-so-far solutions, one for each run, found by the three algorithms respectively when the number of fitness evaluations is 100 per run. The results in Table 6 show that the results found by GL-DE are significantly

Table 6: Simulation results of the riot scenario with FES = 100.

Algorithm	Training	P value	Testing	P value
GA	2166.4 ± 1455.9	$0.00246\ddagger$	2143.0 ± 1427.9	$0.00178\ddagger$
DE	2102.5 ± 1242.6	$0.00186\ddagger$	2052.6 ± 1221.1	$0.00171\ddagger$
GL-DE	1232.2 ± 970.8	N/A	1200.1 ± 920.6	N/A

Note: $\ddagger$ means the testing results are significant worse than those of GL-DE, according to two-sample t-test at 0.05 level (i.e., P value < 0.05).

better than those found by GA and DE. Therefore, when the maximum number of fitness evaluations is small, the proposed GL-DE is able to provide better performance than GA and DE.

Figure 6: Examples of simulation results with the solutions found by the proposed GL-DE in different generations.

Fig. 6 shows the performances of the solutions found by the proposed GL-DE in different generations. Our objective is to investigate whether the solutions found by the proposed framework is effective and whether the solutions found are becoming better and better. Specifically, we record five groups of solutions during the 30 independent runs. The 1st, 2nd, 3rd and 4th group of solutions are the 30 best-so-far solutions found by the GL-DE in the 2, 5, 10, and 100 generation respectively. The last group consists of 30 solutions which are randomly generated in the search space. Then for each solution (i.e., a configuration of the crowd model), we perform one simulation run where the initial positions of pedestrians are set the same as in the first training case. To verify the calibrated results, we use the accumulated number of pedestrians entered the gate (denoted as τ) at each time step. The average of τ for each group of solutions are plotted in Fig. 6, where the objective curve is the result of the first training case. It can be observed that the τ values of the group of random solutions are the most different from the objective curve. The τ values of the first group of solutions are better than the group of random solutions, but are still much different from the objective curve. As the generation increases, the τ values become more and more similar to the objective curve. The τ values of the 4th group of solutions are very close to the objective curve. These results demonstrate that the proposed framework is effective to calibrate the model to approximate the desired objective behaviors.

In the above studies, we use five training cases to train the parameter setting of the crowd model. In general, using more training cases would result in larger computational cost, because more simulation runs are required to evaluate the

Figure 7: Average of the best fitness values versus the number of training cases

Table 7: Simulation results of the riot scenario with different numbers of training cases.

Study case	Num	Training	P value	Testing	P value
first	1	244.2 ± 16.3	$3.9E-29^\dagger$	451.8 ± 47.0	$0.014^\ddagger$
second	5	317.2 ± 9.8	N/A	428.2 ± 32.2	N/A
third	10	362.0 ± 19.3	$2.4E-16^\ddagger$	413.9 ± 26.7	$0.065^\S$

Note: Num represents the number of training cases; ‡ , § , † mean the results are significant worse than, similar to, and significant better than those of the second study respectively, according to two-sample t-test at 0.05 level.

fitness value of each solution, as expressed in (5). However, using too few training cases may lead to overfitting, that is, the calibrated model fit well with training data but not be able to predict unseen data. In the following parts, we investigate the impacts of the number of training case on the performance of the proposed framework. We design three studies, which use one, five, and ten training cases to train the parameter settings of the crowd model respectively. Fig. 7 shows the evolution curves of the training fitness values in the three studies. It can be observed that during the training process, the algorithm generally converges faster and is able to find better training fitness value, when using fewer number of training cases. Since different training cases contain different patterns of crowd movements, a solution may perform well on one training case but perform badly on another training case. Therefore, it becomes more difficult to find a general good solution (i.e., solution with small training fitness value) as the number of training cases increases.

As shown in Table 7, the testing fitness values (i.e., the prediction performance) of the best-so-far solutions in the first test study are significantly worse than those of the second test study. These results indicate that using a single training case can cause the overfitting problem. Note that the calibrated models are often used for prediction and “what-if” analysis. Thus the testing performance is more important than training performance. From this point of view,

the solutions found in the second test study are better than those found in the first test study. These results suggest that using multiple training cases can obtain better solutions than only using a single training case. However, using too many training cases may not have any more added benefit. As shown in Table VII, the testing fitness values of the best-so-far solutions in the third study are similar to those of the second study. Considering the large computational cost of using too many training cases, adopting an appropriate number of training cases is preferable.

4.3. Case 2: Street Crossing Scenario

Figure 8: Real world pedestrians crossing the street.

In the previous section, the simulation results with a pre-specified set of parameter values of the model were used as the desired behaviors (i.e., benchmark for fitness evaluation). The results are promising as our algorithm could calibrate the model to generate results consistent with the desired ones. In this section, we apply the proposed algorithm to a real world scenario, where a crowd of pedestrians cross a street, as shown in Fig. 8. The scenario data was obtained from a video recorded in Japan, as described in [47]. The video lasts for 60s and involves about 150 pedestrians. The footway region for pedestrian crossing is a $11\text{m} \times 31\text{m}$ rectangle and the positions of pedestrians are recorded every 0.5s. Here we use recorded step to describe the time for convenience. Our objective is to use our proposed algorithm to calibrate the following modified social force model, so as to reproduce the crowd behavior in the video.

For each simulated agent, its initial position and the destination are set to be the first and the last position of its corresponding pedestrian's appearance in the video, respectively. The desired speed of each pedestrian is set to be the average speed of its first three recorded steps. From the video, we observe that pedestrians will rush at the last several seconds when the traffic light starts to blink. To consider such behavioral variance in our model, we introduce two extra parameters on top of the social force model, denoted as T' and V' . We consider that if a pedestrian has not yet entered the footway region but the current time step is larger than T' , then the pedestrian would start to rush, and its desired speed is set to be V' in such a case. Including the four parameters of the social force model, there are totally six parameters to be optimized in this scenario.

The lower and upper bounds of the two scenario specific parameters are listed in Table 8.

Table 8: Scenario Specific Parameters of The Street Crossing Scenario

Name	Description	LB	UB
V'	The speed of running pedestrians	1	6
T'	The recorded step when pedestrians start to run	10	90

Figure 9: Weight values for the street crossing scenario.

In this case study, we generate five training cases to train the parameters and ten testing cases to test the final solutions (Note that five more training cases are generated to study the impacts of the number of training cases in later subsection). For each training and testing case, we randomly choose 50 pedestrians ($\approx 1/3$ of the total pedestrians) whose motions are determined by the crowd simulation model, while the motions of other pedestrians are set the same as in the video data. For example, suppose G_1 is the group of 50 chosen pedestrians in the first training case. To evaluate the fitness of a solution X , we first run a simulation S_1 using X as the parameter setting of the crowd model. In the simulation, the behaviors of pedestrians in G_1 at every time step are determined by the crowd model, while those of others are set the same as in the video data. Then we can calculate $D(S_1, O)$ by (5), where O is the objective behaviors in the video. In this way, we can calculate $D(S_i, O)$ for each G_i , and the training fitness value of X can be computed by (6). Similarly, the testing fitness value of X using the ten testing cases can also be computed by (7).

When computing fitness values, we assign larger weight to the left and right sides of the footway region. This is because the density of the crowd becomes large in the middle, and the subtle differences in the collision avoidance behaviors do not influence much the overall crowd densities in this area much. Specifically, the weight values used in the fitness evaluation process are shown

Table 9: Fitness values of the final solutions on the street crossing scenario with FES = 2000.

Algorithm	Training	P value	Testing	P value
GA	230.2 $\pm$ 3.9	1.8E-9 ‡	244.5 $\pm$ 3.8	0.001‡
DE	227.1 $\pm$ 3.0	2.7E-4 ‡	241.9 $\pm$ 4.0	0.23§
GL-DE	223.8 $\pm$ 3.8	N/A	241.1 $\pm$ 4.3	N/A

Note:‡ and § mean the results are significantly worse than and similar to those of GL-DE respectively, according to two-sample t-test at 0.05 level.

in Fig. 9. The weight values are computed by

$$w_i = \exp(-\frac{\xi_i}{10}) + \exp(\frac{-(31 - \xi_i)}{10}) \quad (25)$$

where ξ_i is the x position of the representative point, the two constants 31 and 10 are set empirically based on the size of the footway region.

Similar to the first case study, we perform GA, DE, and GL-DE for 30 independent runs on this scenario respectively. The parameter p of GL-DE is set to be 0.5, while other parameters of the compared algorithms are set the same as in Table 4.

Figure 10: Average of the best fitness values versus the number of generations on the real world scenario.

Fig. 10 shows the average of the best-so-far training fitness values found by the EAs versus the number of fitness evaluations. Similar to the results in the first case study, DE converges faster than GA and our proposed GL-DE converges the fastest. The converging fitness value could indicate that the simulated crowd behaviors become more and more similar to the video data, in terms of the density distribution of the crowd over time.

Table 9 shows the fitness values of the best-so-far solutions found at FES = 2000. It can be observed that the training fitness values of GL-DE are signifi-

Table 10: Fitness values of the final solutions on the street crossing scenario with FES = 100.

Algorithm	Training	P value	Testing	P value
GA	249.4 $\pm$ 8.7	1.9E-4‡	264.2 $\pm$ 10.1	0.002‡
DE	250.7 $\pm$ 7.0	2.2E-6‡	263.2 $\pm$ 10.0	0.005‡
GL-DE	242.0 $\pm$ 6.3	N/A	257.1 $\pm$ 7.5	N/A

Note:‡ mean the results are significantly worse than those of GL-DE according to two-sample t-test at 0.05 level.

cantly better than those of GA and DE. The testing fitness values show that the solutions found by GL-DE are significantly better than those found by GA. The solutions found by GL-DE are slightly (not significantly) better than those of DE. However, the results in Table 10 demonstrate that the best-so-far solutions found by GL-DE at FES = 100 are significantly better than those found by the GA and DE, indicating that GL-DE is more efficient than GA and DE.

Figure 11: Examples of simulation results with the solutions found by the proposed GL-DE in different generations.

To investigate the performances of the solutions found by the proposed framework during the evolution, we record four groups of solutions during the 30 independent runs. The 1st, 2nd, and 3rd group of solutions are the 30 best-so-far solutions found by the GL-DE in the 2, 10, and 100 generation respectively. The last group of solutions consist of 30 solutions which are randomly generated in the search space. For each solution, we perform 20 simulations based on the 10 training cases and 10 test cases respectively. During each simulation run, we record the speeds of the 50 chosen pedestrians at each time step. If the pedestrian walk out of the footway region at a time step, this pedestrian would not be considered at the time step. The average speed of the chosen pedestrians in the simulated behaviors at each time step are plotted in Fig. 11. The objective curve in Fig. 11 is the average speeds of the chosen pedestrians in video data.

Table 11: Simulation results of the street crossing scenario using different numbers of training cases.

Study case	Num	Training	P value	Testing	P value
first	1	193.9 ± 3.3	$2.7\text{E-}39\ddagger$	242.8 ± 4.8	$0.075\S$
second	5	223.8 ± 3.8	N/A	241.1 ± 4.3	N/A
third	10	230.7 ± 2.9	$9.1\text{E-}11\ddagger$	240.2 ± 3.0	$0.17\S$

Note: Num represents the number of training cases; $\ddagger$, $\S$, $\ddagger$ mean the results are significantly worse than, similar to, and significant better than those of the second study respectively, according to two sample t-test at 0.05 level.

Note that the original speed of pedestrians are set to be the average speed of their first three recorded steps. Parameters which influence the speed profile most include the time to rush (i.e., T') and the rush speed (i.e., V'). It can be observed that the speed profile of the random solutions is rather different from the objective curve. During the time period of 10s and 40s, the average speed of the “random setting” is around 1.7 m/s which is larger than the objective speed (around 1.3 m/s). This is because that the T' values of some random solutions are too small, so that the pedestrians would start to run at too early time steps. As the density of pedestrians in the footway region during 20s and 45s is relatively large, the speed of the “random setting” slows down during this period. As the generation increases, the speed profile becomes more and more similar to the objective ones. Taking the results of “generation = 100” for example, during the time period of 10s and 40s, the average speed is around 1.3. Then the average speed starts to increase after 40s. The above features are consist with the objective curve. Generally speaking, the proposed framework shows good performance for the applications studied in the paper.

Figure 12: Average of the best fitness values versus the number of fitness evaluations on the real world scenario.

In the following parts, we investigate the impacts of the number of training cases in the street crossing scenario. Similar to the first case study, we design

three studies, which use one, five, and ten training cases respectively. Each training case associates to a group of 50 random pedestrians, whose motions are determined by the crowd model. Fig. 12 shows the evolution curves of the fitness values of the three studies. It can be observed that the training fitness values generally converge slower when the number of training cases increases.

The fitness values of the best-so-far solutions found in the three studies are listed in Table 11. The results show that the testing fitness values found in the second study are better than those of the first study (The P value = 0.075 is very near the significant level 0.05). The testing fitness values found in the third study are slightly better than those of the second study. Similar to the first case study, these results suggest that selecting an appropriate number of training cases is important to obtain a good prediction performance.

5. Conclusions

In this paper, we have proposed a density-based evolutionary framework for crowd model calibration. In the proposed framework, a general and effective density-based matching scheme is introduced for automatic crowd model evaluation. Moreover, an enhanced differential evolution with a hybrid search mechanism is proposed to efficiently search for correct parameter settings. The proposed framework is used to calibrate two modified social-force models. Simulation results demonstrate that the proposed framework can calibrate crowd models effectively and efficiently to reproduce desired macroscopic crowd behaviors.

The key issue in applying the proposed framework is to obtain the behavior features of objective crowd behaviors, i.e., the density distribution of agents over time. If these features are available (e.g., extracted from videos), then our framework can be used to tune the parameters of a given crowd model to approximate the desired macro-scale dynamics.

There are several interesting future research directions. One direction is to extend the proposed framework to evolve both behavior rules and parameters simultaneously. In some applications where the given model is incomplete or partially not well-defined, tuning both parameters and behavior rules can achieve better performance [48]. Another promising research topic is to extend the proposed density-based matching scheme by incorporating other metrics such as moving direction and the entropy metric [49].

6. Acknowledgment

Nan Hu is supported by the Science and Engineering Research Council of the Agency for Science, Technology and Research (A*STAR) of Singapore (Complex Systems Programme grant number 1224504056). Michael Lees is also affiliated with ITMO University, St. Petersburg, Russian Federation and his work is partially supported by Russian Scientific Foundation, Project #14-21-00137. Linbo Luo is supported by “the Fundamental Research Funds for the Central Universities”, Project XJS14013.

References

- [1] D. Thalmann, Crowd simulation, Wiley Online Library, 2007.
- [2] B. Ulicny, D. Thalmann, Crowd simulation for interactive virtual environments and vr training systems, in: Computer Animation and Simulation 2001, Springer, 2001, pp. 163–170.
- [3] D. Helbing, I. J. Farkas, P. Molnar, T. Vicsek, Simulation of pedestrian crowds in normal and evacuation situations, *Pedestrian and evacuation dynamics* 21 (2002) 21–58.
- [4] K. Christensen, Y. Sasaki, Agent-based emergency evacuation simulation with individuals with disabilities in the population, *Journal of Artificial Societies and Social Simulation* 11 (3) (2008) 9.
- [5] A. Schadschneider, W. Klingsch, H. Klüpfel, T. Kretz, C. Rogsch, A. Seyfried, Evacuation dynamics: Empirical results, modeling and applications, in: *Encyclopedia of complexity and systems science*, Springer, 2009, pp. 3142–3176.
- [6] A. Zoumpoulaki, N. Avradinis, S. Vosinakis, A multi-agent simulation framework for emergency evacuations incorporating personality and emotions, in: *Artificial Intelligence: Theories, Models and Applications*, Springer, 2010, pp. 423–428.
- [7] D. Helbing, P. Molnar, Social force model for pedestrian dynamics, *Physical review E* 51 (5) (1995) 4282.
- [8] A. Braun, B. E. Bodmann, S. R. Musse, Simulating virtual crowds in emergency situations, in: *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, 2005, pp. 244–252.
- [9] R. L. Hughes, The flow of human crowds, *Annual review of fluid mechanics* 35 (1) (2003) 169–182.
- [10] A. Treuille, S. Cooper, Z. Popović, Continuum crowds, in: *ACM Transactions on Graphics*, Vol. 25, ACM, 2006, pp. 1160–1168.
- [11] S. R. Musse, D. Thalmann, Hierarchical model for real time simulation of virtual human crowds, *IEEE Transactions on Visualization and Computer Graphics* 7 (2) (2001) 152–164.
- [12] W. Shao, D. Terzopoulos, Autonomous pedestrians, in: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 2005, pp. 19–28.
- [13] L. Luo, S. Zhou, W. Cai, M. Y. H. Low, F. Tian, Y. Wang, X. Xiao, D. Chen, Agent-based human behavior modeling for crowd simulation, *Computer Animation and Virtual Worlds* 19 (3-4) (2008) 271–281.

- [14] N. Pelechano, J. M. Allbeck, N. I. Badler, Virtual crowds: Methods, simulation, and control, *Synthesis Lectures on Computer Graphics and Animation* 3 (1) (2008) 1–176.
- [15] L. Luo, S. Zhou, W. Cai, M. Lees, M. Low, K. Sornum, Humdpm: a decision process model for modeling human-like behaviors in time-critical and uncertain situations, *Transactions on computational science XII* (2011) 206–230.
- [16] S. Zhou, D. Chen, W. Cai, L. Luo, M. Y. H. Low, F. Tian, V. S.-H. Tay, D. W. S. Ong, B. D. Hamilton, Crowd modeling and simulation technologies, *ACM Trans. Model. Comput. Simul* 20 (4) (2010) 20.
- [17] A. Johansson, D. Helbing, K. S. PRADYUMN, Specification of the social force pedestrian model by evolutionary adjustment to video tracking data, *Advances in Complex Systems* 10 (supp02) (2007) 271–288.
- [18] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, J. Pettre, Parameter estimation and comparative evaluation of crowd simulations, *EUROGRAPHICS 2014* 33 (2).
- [19] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments—a survey, *IEEE Trans. Evol. Comput.* 9 (3) (2005) 303–317.
- [20] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Trans. Evol. Comput.* 15 (1) (2011) 4–31.
- [21] K. H. Lee, M. G. Choi, Q. Hong, J. Lee, Group behavior from video: a data-driven approach to crowd simulation, in: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 2007, pp. 109–118.
- [22] J. Tubbs, B. Meacham, *Egress design solutions: A guide to evacuation and crowd management planning*, John Wiley & Sons, 2007.
- [23] A. Ferscha, K. Zia, On the efficiency of lifebelt based crowd evacuation, in: *Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, IEEE Computer Society, 2009, pp. 13–20.
- [24] A. Shende, M. P. Singh, P. Kachroo, Optimization-based feedback control for pedestrian evacuation from an exit corridor, *IEEE Transactions on Intelligent Transportation Systems* 12 (4) (2011) 1167–1176.
- [25] A. Shende, M. P. Singh, P. Kachroo, Optimal feedback flow rates for pedestrian evacuation in a network of corridors, *IEEE Transactions on Intelligent Transportation Systems* 14 (3) (2013) 1053–1066.
- [26] K. Price, R. Storn, Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous space, Tech. rep., Technical Report, International Computer Science Institute, Berkeley (1995).

- [27] M. C. Hill, C. R. Tiedeman, *Effective groundwater model calibration: with analysis of data, sensitivities, predictions, and uncertainty*, John Wiley & Sons, 2006.
- [28] D. C. Taylor, V. Pawar, D. Kruzikas, K. E. Gilmore, A. Pandya, R. Iskandar, M. C. Weinstein, *Methods of model calibration*, *Pharmacoeconomics* 28 (11) (2010) 995–1000.
- [29] M. C. Hill, *Methods and guidelines for effective model calibration*, US Geological Survey Denver, CO, USA, 1998.
- [30] J. Yuan, S. H. Ng, K. L. Tsui, *Calibration of stochastic computer models using stochastic approximation methods*, *IEEE Transactions on Automation Science and Engineering* 10 (1) (2013) 171–186.
- [31] S. Liu, D. Butler, R. Brazier, L. Heathwaite, S.-T. Khu, *Using genetic algorithms to calibrate a water quality model*, *Science of the total environment* 374 (2) (2007) 260–272.
- [32] R. Guzmán-Cruz, R. Castañeda-Miranda, J. García-Escalante, I. López-Cruz, A. Lara-Herrera, J. De la Rosa, *Calibration of a greenhouse climate model using evolutionary algorithms*, *Biosystems engineering* 104 (1) (2009) 135–142.
- [33] J. Jemai, P. C. Eggers, G. F. Pedersen, T. Kurner, *Calibration of a uwb sub-band channel model using simulated annealing*, *IEEE Transactions on Antennas and Propagation* 57 (10) (2009) 3439–3443.
- [34] B. Calvez, G. Hutzler, *Parameter space exploration of agent-based models*, in: *Knowledge-Based Intelligent Information and Engineering Systems*, Springer, 2005, pp. 633–639.
- [35] B. Calvez, G. Hutzler, *Automatic tuning of agent-based models using genetic algorithms*, in: *Multi-Agent-Based Simulation VI*, Springer, 2006, pp. 41–57.
- [36] F. Stonedahl, U. Wilensky, *Finding forms of flocking: Evolutionary search in abm parameter-spaces*, in: *Multi-Agent-Based Simulation XI*, Springer, 2011, pp. 61–75.
- [37] V. Smith, *Evolving an agent-based model to probe behavioral rules in flocks of cowbirds*, *Artificial Life* 11 (2008) 561.
- [38] N. Pelechano, C. Stocker, J. Allbeck, N. Badler, *Being a part of the crowd: towards validating vr crowds using presence*, in: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 136–142.

- [39] A. Lerner, Y. Chrysanthou, D. Lischinski, Crowds by example, in: Computer Graphics Forum, Vol. 26, Wiley Online Library, 2007, pp. 655–664.
- [40] E. Ju, M. G. Choi, M. Park, J. Lee, K. H. Lee, S. Takahashi, Morphable crowds, ACM Transactions on Graphics (TOG) 29 (6) (2010) 140.
- [41] M. Wirz, T. Franke, D. Roggen, E. Mitleton-Kelly, P. Lukowicz, G. Tröster, Probing crowd density through smartphones in city-scale mass gatherings, EPJ Data Science 2 (1) (2013) 5.
- [42] D. Helbing, A. Johansson, H. Z. Al-Abideen, Dynamics of crowd disasters: an empirical study, Physical review E 75 (4) (2007) 046109.
- [43] S.-F. Lin, J.-Y. Chen, H.-X. Chao, Estimation of number of people in crowded scenes using perspective transformation, IEEE Trans. Syst., Man, Cybern. A, Syst., Humans 31 (6) (2001) 645–654.
- [44] E. Mezura-Montes, J. Velázquez-Reyes, C. A. Coello Coello, A comparative study of differential evolution variants for global optimization, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM, 2006, pp. 485–492.
- [45] D. Helbing, I. Farkas, T. Vicsek, Simulating dynamical features of escape panic, Nature 407 (6803) (2000) 487–490.
- [46] N. Collier, Repast HPC manual, Technical report (Nov 2010), 2010.
- [47] K. Teknomo, Application of microscopic pedestrian simulation model, Transportation Research Part F: Traffic Psychology and Behaviour 9 (1) (2006) 15–27.
- [48] J. Zhong, L. Luo, W. Cai, M. Lees, Automatic rule identification for agent-based crowd models through gene expression programming, in: AAMAS 2014, 2014.
- [49] S. J. Guy, J. van den Berg, W. Liu, R. Lau, M. C. Lin, D. Manocha, A statistical similarity measure for aggregate crowd dynamics, ACM Transactions on Graphics (TOG) 31 (6) (2012) 190.