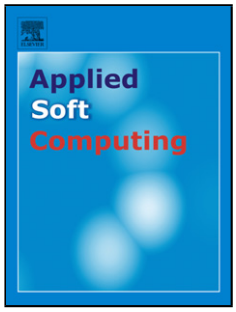


Title: Guide them through: an automatic crowd control framework using multi-objective genetic programming

Author: Nan Hu Jinghui Zhong Joey Tianyi Zhou Suiping Zhou Wentong Cai Christopher Monterola

PII: S1568-4946(18)30043-7
DOI: https://doi.org/doi:10.1016/j.asoc.2018.01.037
Reference: ASOC 4684

To appear in: *Applied Soft Computing*

Received date: 20-7-2016
Revised date: 26-12-2017
Accepted date: 23-1-2018

Please cite this article as: Nan Hu, Jinghui Zhong, Joey Tianyi Zhou, Suiping Zhou, Wentong Cai, Christopher Monterola, Guide them through: an automatic crowd control framework using multi-objective genetic programming, *<![CDATA[Applied Soft Computing Journal]]>* (2018), https://doi.org/10.1016/j.asoc.2018.01.037

Guide them through: an automatic crowd control framework using multi-objective genetic programming

Abstract

We propose an automatic crowd control framework based on multi-objective optimisation of strategy space using genetic programming. In particular, based on the sensed local crowd densities at different segments, our framework is capable of generating control strategies that guide the individuals on *when* and *where* to slow down for optimal overall crowd flow in realtime, quantitatively measured by multiple objectives such as shorter travel time and less congestion along the path. The resulting Pareto-front allows selection of resilient and efficient crowd control strategies in different situations. We first chose a benchmark scenario as used in [1] to test the proposed method. Results show that our method is capable of finding control strategies that are not only quantitatively measured better, but also well aligned with domain experts' recommendations on effective crowd control such as "slower is faster" and "asymmetric control". We further applied the proposed framework in actual event planning with approximately 400 participants navigating through a multi-story building. In comparison with the baseline crowd models that do no employ control strategies or just use some hard-coded rules, the proposed framework achieves a shorter travel time and a significantly lower (20%) congestion along critical segments of the path.

Keywords: Crowd modelling and simulation, Crowd Control, Genetic Programming, Multi-objective Optimisation

2016 MSC: 00-01, 99-00

1. Introduction

Crowd modelling and simulation has gained increasing attention from industry, academia and government due to its wide applications [2] to understand, replicate and

predict crowd dynamics in various situations. As a natural extension to and an application of crowd modelling and simulation, crowd control aims to intervene [3] the movement of crowds in a desired manner so that certain objectives are met, for instance, to prevent turbulence or stampede in events involving massive crowds, to avoid bottlenecks of crowd flow, or to minimize overall travelling time etc.

To apply appropriate crowd control strategies to intervene the crowd in a desired manner, one needs to first understand the implicit (unintervened) crowd dynamics under specific scenarios, which can be studied through crowd modeling and simulation. One promising approach is agent-based modelling (ABM), which treats individuals as agents that can perceive, decide and act independently based on some rules [4]. From the ABM perspective, crowd dynamics emerge from the motions of individuals, and the motions can be generated through a simplified two-layer movement model [5]. At the *path planning* layer (the higher layer), an agent plans/finds a path to navigate through the environment. The path segments are usually formed as a list of waypoints representing important landmarks and accessable areas. While at the *collision avoidance* layer (the lower layer), it avoids collisions with others while moving along the planned path. From the modeling perspective, there are some established methods to specify the rules for agents at both layers. For path planning, both shortest path algorithms (such as A^*) [6] and accumulative segment-based algorithms that take account of vision range [7] have been well established to guide an agent to move through a set of static spatial obsctales in an environment. For collision-avoidance, algorithms such as reciprocal velocity obstacle (RVO) and its variants [8], and social force model (SFM) [9] are proven efficient and widely adopted. With the two layers of movement behaviours, ABM can generate various crowd dynamics given the initial configurations of the agents (e.g., preferred speed, personal space factor etc.) and the environment (e.g., waypoints of paths, obstacles etc.).

Due to the complex interactions among the agents, the stochastic nature of the crowd model and the large number of parameters involved, finding a "good" crowd control strategy that explicitly intervenes movements of crowds in order to produce the "desired" crowd movements often requires a large number of simulations, which is time-consuming if performed manually. It is therefore important to automate the

search process for optimal crowd control strategies. Evolutionary algorithms (EAs) are population-based non-deterministic search algorithms, which can be used to adaptively evolve a simulation model through automating the calibration of model parameters as well as model structures (such as behavioural rules of agents) [10, 11]. Although there are scattered existing works on using EAs for automatic crowd control, they mainly focus on the optimization of parameters, which may limit the search space by the fixed number of parameters. In this paper, we apply Genetic programming (GP) to enable both parameter and structure evolving for automatic crowd control, which will be depicted in Section 4.1.

The need for multi-objective optimization [12] is also essential for crowd control, as a good control strategy often needs to achieve different aspects of crowd dynamics simultaneously. For example, increasing the speed of an escalator may improve the flow rate of one segment of a path, while it may cause congestion at other (e.g., the subsequent) segments if there are spatial bottlenecks. Thus, the overall flow rate and the congestion conditions along the path need to be considered simultaneously in searching for a good crowd control strategy in this case. In this paper, our proposed GP-based framework can automatically search for the optimized parameters and rules used in an agent-based crowd model for crowd control purposes, specifically to optimize multiple objectives from the crowd dynamics perspective.

The rest of the paper is organized as follows: Section 2 describes the existing efforts in applying EAs to calibrate crowd simulation models, and traditional crowd control approaches. The problem of automatic crowd control through optimization of an agent-based model is formally defined in Section 3. As the proposed solution to address the problem, the GP-based crowd control framework is discussed in Section 4. In Section 5, we test the framework with two scenarios, a well studied evacuation scenario in [1] and a real life event planning scenario, where approximately 400 delegates are directed to leave a multi-story building with escalators transporting between stories. Section 6 concludes the paper and gives recommendations for future work.

2. Related Work

2.1. Application of Evolutionary Algorithms in Crowd Simulation Models

Modeling and simulation has become a promising approach to study crowd dynamics in recent years. Various models [4, 5, 13, 14] have been proposed with different focuses on particular aspects of a crowd according to the requirements of an application. One common and critical objective of these models is to generate *realistic* crowd behaviors through model calibration and validation [10, 11, 15], and variations of EAs have been applied to achieve this goal.

The most common idea is to use EAs to tune parameters of a crowd model so that the *microscopic* individual behaviors (e.g., moving trajectories) can match those retrieved from image or video data. For example, Johansson et al. [16], applied an EA to calibrate the parameters of the social force model (SFM). They tried to match the microscopic motions of pedestrians such as the moving speed and direction. Similarly, Li et al. [17] used a Genetic Algorithm (GA) to find an optimal set of weighting parameters for composing virtual forces in a crowd model. On the other hand, efforts have also been made to tune crowd model to match *macroscopic* crowd features such as dominant moving directions and paths of the crowd in a specific scenario. For example, Zhong et al. [11] proposed an EA-based framework to evolve the parameters of modified SFM in order to match *macroscopic* crowd features (i.e., the crowd densities). Wolinski et al. [18] also suggected several macroscopic metrics for model calibration based on EAs.

There are two major differences of our proposed approach compared to these methods. First, most existing EA-based approaches focus on tuning parameter settings of specific models (e.g., SFM and RVO2) based on videos [18], assuming a predefined set of rules can capture different crowd dynamics well under different situations. However, this may not be the case as crowd behaviors are complex and stochastic in nature. Thus, not only the parameters but also some behavioral rules need to be evolved and applied to a specific situation to reproduce the observed crowd behaviors. Zhong et al. [10] has demonstrated GP-based approaches can be applied to find behavioral rules of crowd in order to match the simulation results with the empirical data. In this paper,

we use a GP-based apporach to allow both rule structure and parameter evolving in order to search for the most fit rules to influence (control) crowd behaviors. Second, the EA-based optimization approach has seldom been applied in crowd control, which *explicitly* intervene the crowd behaviors besides the implicit behaviors rules. In this paper, we define and differentiate two sets of rules and parameters that implicitly and explicitly influence crowd's behaviors, respectively. Implicit rules mimic intrinsic rules that agents follow to define their navigational and collision-avoidance behaviors, while explicit rules are defined for crowd control intervention. Exisiting EA-based approach has been focused on the implicit set optimization in order to achieve realistic simulation results; whereas we will focus on optimizing the explicit set that may represent some crowd control strategies which can be maneuvered by planners to better manage large crowds.

2.2. Existing Crowd Control Methods

Automatic crowd control is an emerging research topic in crowd studies. Traditionally, two main levels of controlling measures are adopted for crowd control [3, 19]. On the *aggregated* level, interventional measures are applied to regulate and improve the overal crowd flow by artificially setting up spatial and/or temporal constraints on the crowd. For example, spatial constraints such as barriers or fences are usually used to separate crowd flow in front of a bottleneck entrance and to direct the crowd in S shape queues in many crowded scenarios. Temporal intervention such as releasing a large crowd in batches with freeze time in between is another common control measure. Such interventional measures usually serve as general guidelines based on past experience and are almost fixed at the pre-planning stage of a new crowd scenario. While on the *individual* level, some regulators geared with different non-lethal weapons/tools are arranged to guide and ease hotspots within the crowd in realtime to prevent crowd incidents. This level of crowd control is subject to the dynamic change of the current crowd status, and thus it is a challenge for planners to optimize the deployment of such regulators before the actual events. In summary, crowd control is a complex task with the combination of various factors and procedures to intervene the crowd dynamics in a desired manner. Due to the dynamic interactions of crowds, crowd control solu-

tions/strategies cannot merely reply on empirical guidelines, and on-site intervention strategies need to be studied according to the dynamics of crowd. In this context, agent-based models with proper calibration can be used to test individual solutions/strategies in realtime through simulation-based *what-if* analysis.

In an agent-based crowd model, crowd control solutions/strategies can be implemented as a set of scenario-specific parameters and rules to affect agents' movement. Due to the large number of parameter/rule combinations, it is necessary to automate the process of searching for the optimal combination to generate the desired effect on crowd dynamics. EA-based methods have been applied for the automatic optimization purpose. Eldridge and Maciejewski [20] proposed to use social robots in crowded situations to improve pedestrian flow. In this method, a GA is used to find the optimal parameters for the interaction model between the robots and the people. Schubert and Suzic [21] proposed a GA with stochastic simulation to learn control strategies for riot control. A fixed set of parameters such as the number, position and strength of barriers were optimized through the GA in this study. More recently, Hu et al. [22] proposed an EA based method for crowd control in a military operation in urban terrain scenario. They developed an appraisal-based emotion model and allowed the soldier agents to intervene crowds' movement through emotional influence. Similar to other studies mentioned above, only scenario-specific parameters were evolved using an Complex Adaptive Systems Evolver (CASE) [23]. The effectiveness of a control strategy was measured through the aggregated emotion levels such as anger and fear, instead of the crowd dynamics emerged from the movement of the crowd.

In summary, existing works mainly focus on the optimization of certain fixed parameter set, which may limit the search space of an EA (with the fixed number of parameters) to find the optimal crowd control solutions/strategies. Such evolvable parameters may only re-emphasize and fine-tune the known guidelines for crowd control (e.g., barriers deployments), which seems insufficient to address the crowd control requirements in dynamic situations. In this paper, we extend the existing work and target for a more flexible optimization method to find innovative and feasible crowd control solutions/strategies through both rule structure and parameter evolving.

3. Problem Definition

In this paper, crowd control is defined as a multi-objective optimization problem over an agent-based crowd simulation model. The inputs of the crowd model consist of agent-specific configurations and scenario-specific configurations as shown in Figure 1.

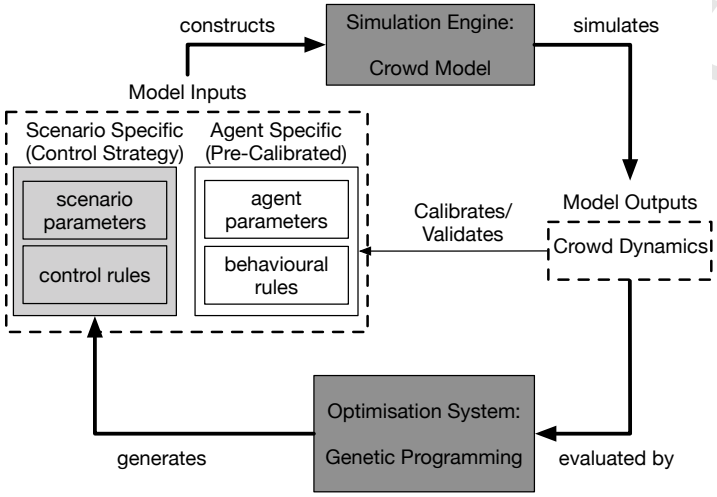


Figure 1: The proposed crowd control framework.

Agent-specific configurations include parameters describing an agent's characteristics (e.g., preferred walking speed and personal space factor etc.), as well as behavioural rules that determine its movement implicitly. In the two-layer motion model, path-finding algorithms and collision avoidance algorithms can be used to define such behavioural rules. In this study, a list of waypoints is used to represent the path for an agent to move through at the path planning layer. At the collision avoidance layer, agents avoid collisions with both the static and the dynamic obstacles (i.e., other agents) along the path. Various collision avoidance algorithms have been proposed from both crowd modeling and robotics research communities. In this paper, both social force model [24] and RVO2 [8] are adopted in the two scenarios respectively due to their effectiveness in generating realistic collision avoidance behaviors of pedestrians [18]. In both models, each agent takes into account the observed static and dynamic obstacles and selects an actual velocity for the agent to avoid collisions with deviation

from its current preferred velocity. In crowd simulation studies, the preferred walking speed of an agent is usually initialized at the beginning of the simulation according to the known crowd profile (e.g., normal mean walking speed of 1.3 m/s is reported for adults [25, 26], while the old and children can have different speed profiles).

While agent-specific configurations are determined by the crowd itself and are usually beyond the control of decision-makers/responders, some scenario-specific configurations can be manipulated explicitly to affect the movement of a crowd.

Focusing on these controllable scenario-specific configurations, we define a *control strategy* as the setting of scenario-specific parameters (denoted as **M**) and external rules (denoted as **R**) that are used by decision makers/responders to affect the movement of a crowd. For example, in an event planning scenario, combinations of different numbers of marshals to be deployed and the instructions these marshals issued to the crowd are considered as control strategies to facilitate the crowd flow in the event. In this study, we suppose the agent-specific configurations are known in advance (e.g., they can be calibrated independently based on empirical data in a specific scenario). Our objective is to optimize the scenario-specific parameters and rules (i.e., **M** and **R**) so as to minimize multiple objectives (denoted as **F**). Hence, the problem of automatic crowd control optimization now is formulated as follows:

$$minimize\ \mathbf{F}(\mathbf{M},\mathbf{R}) = (F_1(\mathbf{M},\mathbf{R}),...,F_m(\mathbf{M},\mathbf{R}))^T \tag{1}$$

where $F_i(\mathbf{M},\mathbf{R})$ is the i-th objective function and m is the total number of objective functions. The desired crowd dynamics define the objective functions (fitness function) for the optimization. In scenarios with dense crowds, *travel time* and *densities* are two important measures to evaluate the effectiveness of a crowd control strategy. In general, travel time reflects the overall effectiveness of crowd movement, for instance, in crowd evacuation and event planning scenarios; whereas density plays an important role in characterizing crowd status. For instance, in dense scenarios with a crowd density beyond the critical density, there is a fundamental reverse relationship between the average moving speed and crowd density [27]. Thus, highly dense crowd becomes not only susceptible to cascading crowd failures [28], such as stampede, but also less effective in general.

4. Proposed Method

To address the crowd control optimization problem, we propose an automatic crowd control framework as shown in Figure 1, with mainly two components: the *simulation engine* and the *optimization system*. The simulation engine and the optimization system form a feedback system: the control strategies generated by the optimization system are fed into the simulation engine to affect the crowd movement; while the output of the simulation engine (for instance, travel time and crowd density) are used by the optimization system as fitness functions to find better crowd control strategies. Note that the control strategies are not executed by the simulation engine at every simulation frame - they are triggered by certain conditions (e.g. segment densities become higher than thresholds).

Agent-specific configurations of the simulation model can be calibrated separately for a specific scenario, while only control strategies (scenario specific configurations) that are used to influence the movement of agents are evolved as shown in Figure 1. For example, speed reduction control for agents at specific segments along the path will change their original preferred speeds and thus results in different simulation results. The *optimisation system* receives the simulation results and corresponding control strategy as inputs, evaluates the results and produces a new "generation" of control strategies to feedback into the model through genetic programming. The generation of these new control strategies is driven by specific desired crowd dynamics in a given scenario (e.g., minimizing the travel time and the the densities along the path). The details of the genetic programming algorithm are depicted below.

4.1. Traditional Chromosome Representation of CGP

We propose a GP-based framework for automatic crowd control in this work. Cartesian Genetic Programming (CGP) is a famous GP variant [29], which has been widely used to solve many real-world complex optimization problems. In CGP, each chromosome represents a directed graph which can be further decoded as a computer program (e.g., a formula or a logical rule). The directed graph contains two types of nodes: function nodes and output nodes. Each function node represents a particular function

and it contains a number of genes. The first gene represents the function type (e.g., "+"), and the remaining genes describe the input sources of the function. The input sources of a function can be a previous function node or a terminal. The number of input sources of a function type is determined by the function type. For example, "+" has two input sources while "sin" has one input source. The output nodes describe which function node (or terminal) should be used to generate the final output. Based on the above descriptions, the chromosome of CGP with one output can be represented by the following vector of integers:

$$[f_1, t_{1,1}, ..., t_{1,m}, ..., f_n, t_{n,1}, ..., t_{n,m}, o] \tag{2}$$

where n is the number of function nodes in each chromosome, m is the maximum number of input sources a function may have, f_i represents the function type of the i-th function node, $t_{i,k}$ represents the k-th input source of the i-th function node, and o represents the source index of the output. A typical chromosome with 6 function nodes and 1 output node can be expressed as:

$$[0, 0, 2, 1, 1, 3, 3, 2, 2, 3, 4, 1, 2, 4, 6, 0, 5, 6, 9] \tag{3}$$

Figure 2 shows the directed graph represented by the chromosome. In this example, there are four terminals, six function nodes, and one output node. To construct the directed graph, the four terminals are used to form the first four nodes in the graph. Then, the function nodes in the chromosome are added to the graph one-by-one. Finally, the output is added to the graph. In the CGP, the function type is represented by integers. In the above example, function types are represented as follows: $+ \rightarrow 0, - \rightarrow 1, * \rightarrow 2, sin \rightarrow 3$. The maximum number of input sources among these functions (i.e., m) is 2. The first function node is $< 0, 0, 2 >$, which means that the function type is "+", the two input sources are the first node and the third node of the graph (i.e., x and z) respectively. This function node is then added to the graph as the fourth node. Similarly, in the second function node, the function type is "-" and the input sources of the function are y and the fourth node of the graph respectively. In this way, we can build a directed graph as shown in Figure 2. The final solution is then decoded as $f(x, y, z, u) = (y - u) + sin(z)$. Arbitrary rule structures can be embedded

in the chromosome representatin of CGP, which enables flexible crowd control strategy composition and optimization in this study.

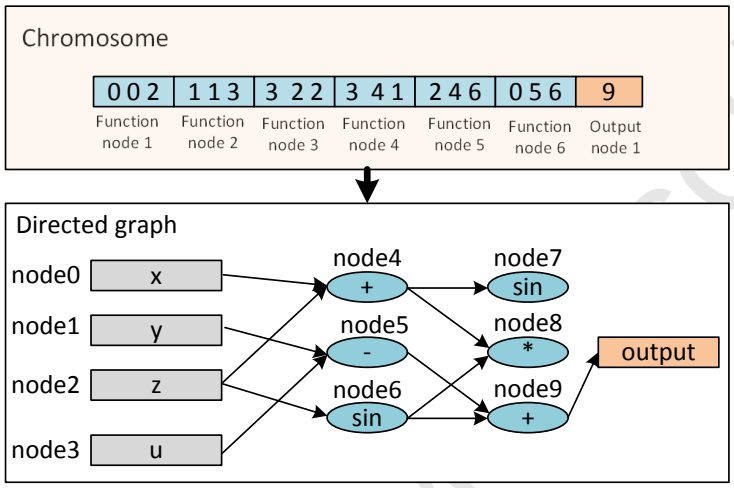


Figure 2: Traditional chromosome representation in CGP.

4.2. Proposed Hybrid Chromosome Representation

In this study, a crowd control strategy consists of 1) a set of scenario-specific *parameters* (e.g., in the second scenario of the event planning, parameters are used to control the different track usage of the escalators), and 2) a set of control *rules* to determine when and where to slow down certain portions (segments) of the crowd (the crowd is segmented and there is a control rule for each segment). In order to evolve both control parameters and rules, the traditional chromosome representation of CGP is extended as follows, which consists of three parts:

$$v = [b_1, b_2, ..., b_o], [f_1, a_{1,1}, a_{1,2}, ..., f_n, a_{n,1}, a_{n,2}], [s_1, s_2, ..., s_K]. \quad (4)$$

where o and K represent the number of parameters and rules to be optimized respectively. The first part represents scenario-specific parameters that can be tuned to affect crowd dyanmics. For example, b_i can be used to represent the escalator operation status. A value of 0 indicates single escalator is to be used at the i-th escaltor lot; and a value of 1 indicates double escalator usage with the same moving direction. In the

second part, the rule's condition with variable structure is represented in the form of a sequential symbolic combinations. f_i represents the function type of the ith function node, $a_{i,k}$ represents the k-th input source of the i-th function node. A function type can be of logical/algorithmic function operator and is used to connect several function nodes in an algebraic expression. In this paper, we use four commonly used logical function operators to link the features and construct the crowd control rules. The four logical operators are $and(\&)$, $or(|)$, $not\_and(\neg\&)$, and $not\_or(\neg|)$. A function node a in this case can be a either a terminal (i.e., the sensed environment information $\rho_{\mathbf{N}}$) or a function node. To limit the search space, we binarise the density values of segments ρ_i to be "high" or "low" density with an integer number of 1 or 0 respectively. Since the inputs values are all binarised in this study, we can limit the function types to be logical operators only. The combination of different segments' densities through the four logic operators will form a rule condition to trigger certain control strategies. The third part s_i represents the source index of output that controls the i-th segment. For example, the value of the i-th output indicates the instructed speed control (reduction) for the i-th segment in this study. If the i-th output equals to 1, then the preferred moving speed of agents in the i-th segment will be reduced (to simplify the problem, we reduce the preferred speed by half for all agents at the segment in simulation). Otherwise, the preferred moving speed of agents in the i-th segment is remained unchanged.

4.3. Proposed Multi-objective Cartesian Genetic Programming Algorithm

Traditional CGP is used for single objective optimization. To address multi-objective optimization problems, non-dominated sorting genetic algorithm II (NSGA-II) [30] is a widely adopted EA due to its efficiency and good spread of solutions. NSGA-II optimizes multiple objectives simultaneously through an approximated Pareto-front by a non-dominated sorting strategy. Inspired by this, we extended CGP to form Multi-Objective Cartesian Genetic Programming (MO-CGP) by utilizing the non-dominated sorting strategy of NSGA-II. The proposed MO-CGP aims to find the best parameter values and control rules to optimize both objectives simultaneously. The output of algorithm is essentially a Pareto-front with multiple objectives. The decision makers/responders can then evaluate candidate control strategies and choose one to apply

according to the requirements of a particular application. For example, travel time (or evacuation time) is critical in air-plane evacuation [31], thus a control strategy resulting in the shortest travel time will be selected. Specifically, the proposed MO-CGP consists of three main steps as follows:

Step 1: Initialization

This step generates N random individuals as the initial population. For each initial individual, the values of the three parts in the chromosome are assigned to a random values by using different methods. For elements of the fist part, each dimension is assigned with either 0 or 1 randomly. For the second part, the ith function node (each function node contains 3 genes) is set by:

$$\begin{cases} f_i = rand_i(0, |\mathbf{F}| - 1) \\ a_{i,1} = rand_i(0, c + \lfloor \frac{i}{3} \rfloor - 1) \\ a_{i,2} = rand_i(0, c + \lfloor \frac{i}{3} \rfloor - 1) \end{cases} \tag{5}$$

where $rand_i(a, b)$ returns a random integer within $[a, b]$, $|\mathbf{F}|$ is the number of functions in the function set, and c is the number of terminals. For the third part, the i-th output is randomly set by:

$$s_i = rand_i(0, n + c) \tag{6}$$

where n is the number of function nodes. When all individuals have been generated, the non-dominated sorting strategy used in NSGA-II [30] is applied subsequently to rank all individuals. Specifically, the individuals are classified into a series of non-dominated fronts at first based on their objective fitness values. The individuals in the higher order fronts are dominated by individuals in lower order fronts, while individuals in the same front are non-dominated with each other. Here individual A dominates B if and only if A is not worse than B in any objective and A is better than B in at least one objective. If A cannot dominate B and B cannot dominate A, then A and B are non-dominated with each other. After ranking all individuals in terms of fitness values, each individual is then assigned with a crowding-distance value. The crowding-distance measures the local density of individuals. The more neighboring

individuals, the higher the crowding-distance an individual will have. Then all individuals are ranked by the following crowded-comparison operator: A is better than B if r(A) $<$ r(B) or (r(A) $=$ r(B) and d(A) $>$ d(B)) where r(A) and d(A) return the front order and the diversity distance of A respectively. In this way, individuals with lower front order and larger diversity distance would come first.

Step 2: Population Reproduction

The standard CGP adopts the $1+\lambda$ evolution strategy without crossover to generate offspring, which has been shown effective to find promising solutions. As in CGP, a mutation is used in this step to generate N new individuals. First of all, to generate an offspring, a random non-dominant parent individual is selected from the population as the base individual. What we hope is that the offspring generated based on a non-dominant individual can be a new non-dominant individual. Then, a point mutation operation is performed on the parent individual to generate an offspring. Each dimension of the chromosome has a probability of pm to be mutated to a new value, i.e.,

$$x_i' = \begin{cases} x_i, \text{if } rand_r(0,1) \geq pm \\ u_i, \text{otherwise} \end{cases} \tag{7}$$

where x_i is the corresponding value in the parent individual, $rand_r(a,b)$ returns a random floating number within (a,b), and pm is the mutation rate. The new value of the dimension u_i is set in the same way as done in the initialization step. Specifically, if u_i belongs to the fist part of the chromosome, then it is assigned with either 0 or 1 randomly. If u_i belongs to the j-th function node of the second part, then it is set by:

$$u_i = \begin{cases} rand_i(0, |\mathbf{F}|-1), x_i \text{represents the function type} \\ rand_i(0, c + \lfloor \frac{j}{3} \rfloor - 1), \text{otherwise} \end{cases} \tag{8}$$

If u_i belongs to the third part, then it is set by:

$$u_i = rand_i(0, n+c) \tag{9}$$

Once, a new individual is created, the chromosome of the new individual will be decoded to obtain a crowd control strategy. The decoded control strategy is then used in

the simulation model to generate the simulation results. The objective values are then evaluated based on the simulation results.

Step 3: Selection

In this step, some worse parent individuals are replaced with better new individuals. First of all, a pool of individuals **U** is constructed by inserting the N new individuals into the current population. The size of **U** is $2 * N$. Then, we use the non-dominated sorting strategy to rank all individuals in **U** and select the N better individuals from **U** to form the new population.

There is a repetition from step 2 to step 3 until the termination condition is met. There are various termination conditions can be used, such as reaching the maximum number of generations.

5. Experiment Studies

In theory, the proposed crowd control framework can be applied in any crowd control scenario, where scenario-specific parameters and the condition of control rules can be represented as a combination of measurable status of different components. In this paper, such components refer to individual segments' densities. To evaluate the effectiveness of the proposed framework, we test it in two scenarios: first, we use a fundamental, and well-studied evacuation scenario as reported in [1] to find optimal control strategies using our framework and compared them with the empirically proven recommendations from the subject-domain experts. Next, we apply the framework to a more complex real-life case for event planning.

In both scenarios, an agent-based crowd model is constructed to simulate the movement of individuals in the environment. In the first scenario, we configure the model parameters according to the calibrated values reported in [1]. In the second scenario, our model is calibrated and validated using data collected from video recordings of an event rehearsal. In both studies, crowd control strategies are applied to improve the crowd dynamics with two specific objectives: 1) to minimize T (the total travel time), and 2) to minimize T_d (the total dense segment-time). A smaller T value means all individuals can evacuate from the room or reach the destination faster, while a smaller T_d

indicates less dense condition over the whole travel period along certain important (for scenario 1) or all (for scenario 2) path segments. The two scenarios also demonstrate the benefit of our multi-objective CGP algorithm to allow the flexible choice of control strategies with different focus on specific objective. In the first evacuation scenario, the control strategy favoring the first objective (T) is preferred, while the second objective (T_d) may be emphasized in the second event planning scenario, which results in smooth and comfortable experience for delegates with less congestion from the crowd along the their paths.

5.1. Scenario 1: Evacuation with a Single Exit

Figure 3 shows the evacuation scenario of one exit in a 15 by 15 meters room. The width of the only exit is set to 1 m following the paper [1]. 200 agents are initiated randomly at the left half of the room and set to move towards the exit at the right hand side. In this case, the paths for all the agents are set to move from their initial position to the exit. We apply the social force model as used in the original paper [1] for the collision avoidance layer in our crowd simulation model. The radii of agents are set according to a normal distribution with mean of 0.2 m and standard deviation of 0.02 m. The preferred speeds of agents are set according to another normal distribution with a mean of 1.3 m/s and a standard deviation of 0.3 m/s. These values are set based on empirical data representing the same context as used in the second event planning scenario. Other parameters for the social force model are set exactly the same as described in [1].

In order to test the proposed framework, we divide the room into 7 segments with a radius of 3 m, 6 m and 9 m, symmetic with regard to the central horizontal line. Since agents are initiated on the left side, we only consider segment 0 to segment 5 (6 segments) for density status monitoring and crowd control rule formation.

The proposed MO-CGP algorithm is used to learn the crowd control rules only (no scenario-specific parameters in this case) by evolving a logic expression that determines whether agents remain at the original preferred speed or slow down by half at each segment based on densities of various segments. That is, the algorithm is supposed to find a speed reduction rule (R_i in Equation 10) for each of the 6 segments, based on

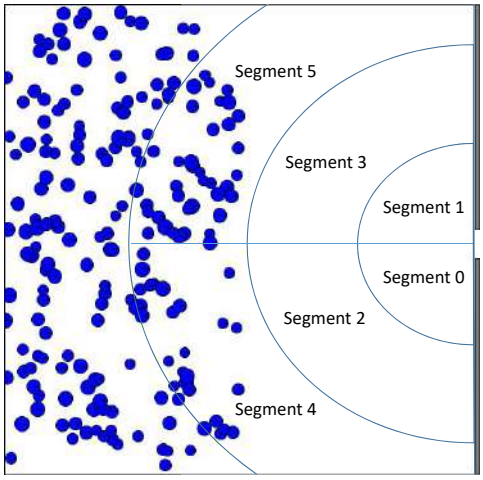


Figure 3: Layout of single evacuation scenario.

the combination of (potentially all the) segement densities. There are two objectives to be optimized, namely the total travel time (T) and the total dense segment-time (T_d). In this case, we note through pilot simulation that the total dense segment time for all segments does not vary much due to the dense scenario configuration, and the lack of variation for path choices (as agents all move towards the exit). Instead, we note different control rules may result in different dense time for the hotspot areas before the exit (i.e. segment 0 and segment 1), which need to be watched out in practise. Thus, for the second objective function T_d in this case, we calculate the over-dense period for only segment 0 and segment 1. The optimization problem is expressed as:

$$\begin{aligned} &minimize\ \mathbf{F}(R_0, ...R_6) \\ &\qquad = (F_1(R_0, ...R_6), F_2(R_0, ..., R_6))^T \\ &subject\ to\ R_i \in \Gamma; i = 1, 2, ..., 6 \end{aligned} \tag{10}$$

where F_1 and F_2 return the total travel time and total dense segment time for segment 0 and segment 1, and Γ is the solution space of control rules for speed regulation of all 6 segments.

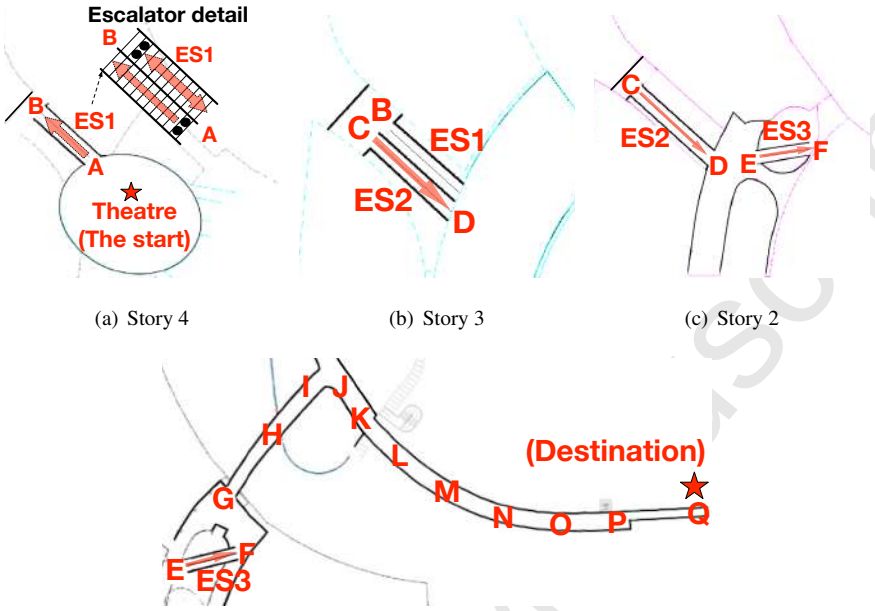
To evaluate the results, we compare our methods with three other benchmarking methods: 1) without any control, and 2) with a hard-coded control rule: *reduce the*

preferred speed of agents for a segment if the density of the same segment is dense (i.e., with equal or more than 1.5 agents per square meters), and 3) a speculative control rule: *reduce the preferred speed of agents for segment 0 or segment 1 if the density of the same segment is dense*. The threshold density value of 1.5 is chosen based on pedestrians' level of service as described in [25]. We will also compare the found optimal control rules with the empirical rules from domain experts such as "slower is faster" and "asymmetric control leads to better flow" for dense crowd as discussed in [24].

5.2. Scenario 2: Event Planning

We then further apply the proposed framework for crowd control during an event planning in a building scenario. In this scenario, around 400 VIP delegates are guided to move from a starting location (i.e., a theatre) to another gathering point inside a multi-story building. The delegates need to move down from story 4 to story 1 through escalators. Along the path, the delegates will be guided by some marshals, who may ask the delegates to slow down at certain points. Marshals are to be deployed along the planned path. The path can be divided into 17 segments represented by a list of way points (i.e., A, B, ..., Q) as shown in Figure 4. There are three escalation lots ES_1, ES_2 and ES_3 connecting different stories. As an example, the details of ES_1 are demonstrated in Figure 4(a). At each escalation lot, two escalators are installed to transport delegates in two directions (i.e., up and down) respectively under normal situation. During special events, both escalators can be set to transport the crowd towards the same direction simultaneously. The width of an escalator is 1.3 m, which can accomodate two persons side by side comfortably.

In this scenario, the crowd dynamics can be complex due to the spatial constraints of the environment: the transporting speed of escalators is fixed and slower than the normal walking speed of delegates, while some path segments (e.g., segment BC) have more limited space than others. With different combinations of escalator direction configurations, the crowd dynamics can vary significantly. Besides, marshals may regulate (intervene) the crowd flow. To prevent over-intervention of the natural movement of delegates, 17 marshals are deployed in the 17 segments to guide direction and



(a) Story 4 (b) Story 3 (c) Story 2

(d) Story 1

Figure 4: Layouts of the virtual environment in a four-story building.

to possiblely slow down the delegates only in the corresponding segment. It requires specific rules to guide the marshals on when they should slow down the delegates at the current segments under their charge. Thus, there are three binary scenario-specific parameters to control the escalators (i.e., b_1, b_2, and b_3) and 17 rules to regulate the speed of delegates in the 17 segments (i.e., $R_1, R_2, ..., R_{17}$). There are two objectives to be optimized, namely the total travel time and total dense segment time. Thus, the optimization problem is expressed as:

$$\begin{aligned} &minimize\ \mathbf{F}(b_1, b_2, b_3, R_1, ..., R_{17}) \\ &\quad = (F_1(b_1, b_2, b_3, R_1, ..., R_{17}), F_2(b_1, b_2, b_3, R_1, ..., R_{17}))^T \\ &subject\ to\ b_i \in \{0, 1\}; i = 1, 2, 3 \\ &\quad R_i \in \Gamma; i = 1, 2, ..., 17 \end{aligned} \tag{11}$$

where F_1 and F_2 return the travel time and total dense segment time of the simulation with the given configurations respectively, and Γ is the solution space of control rules for speed regulation.

5.3. Crowd Model Calibration and Validation

To calibrate the agent specific configurations of the model for the second applied scenario, we use the empirical data such as the known delegates' profile and pilot simulations from rehearsal events. A bounded normal distribution of preferred speed (with mean of 1.3 m/s, standard deviation of 0.4 m/s, minimum of 0.8 m/s and maximum of 1.6 m/s) is assigned to initialize the agents according to pedestrian characteristics in Singapore as reported in [32]. The average pedestrian moving speed under normal situation [26] is used as the mean preferred speed of the agents, while the standard deviation, the minimum and maximum speeds are set to capture the speed variances of the delegates with different genders and ages. The personal space factor, $\phi = 1.0 \times$ agent radius, is assigned in the scenario. It represents the preferred distance between two delegates. The time horizon parameter τ of the RVO2 algorithm is set to 2 to mimic the visual perception time of 2 seconds [8]. A radius is set for each waypoint to cover the width of the path segment. An agent is considered reaching the waypoint once it is within the circle. When an agent reaches a new waypoint, its preferred velocity is set towards a random position inside the circle of the next waypoint along the planned path. At the entrance of each escalator lot, there are maximum four access points with two escalators going the same direction simultaneouly and each escaltor can accomodate two persons at a time. Agents are assumed to stand still on the escalator, where their speeds are set to 0.433 m/s based on the escalator configuration. Obstacle lines and polygons are drawn according to the buildings layout for collision avoidance. Some model details and the calibrated values are shown in Figure 5.

Through calibration, the crowd simulation model is supposed to generate similar crowd dynamics as observed in the scenario. Model validation is conducted to verify this hypothesis. For this purpose, we applied the face validity and the statistical validity of the crowd model by comparing the simulation results with data collected from video recordings of an rehearsal event. During the rehearsal, approximately 400 participants were asked to leave the theatre after a group meeting and walked to the destination along the path as shown in Figure 4. Marshals were deployed along the path to guide them about the moving direction (between waypoints), but with no control on their movement. All participants were supposed to walk naturally at their comfort-

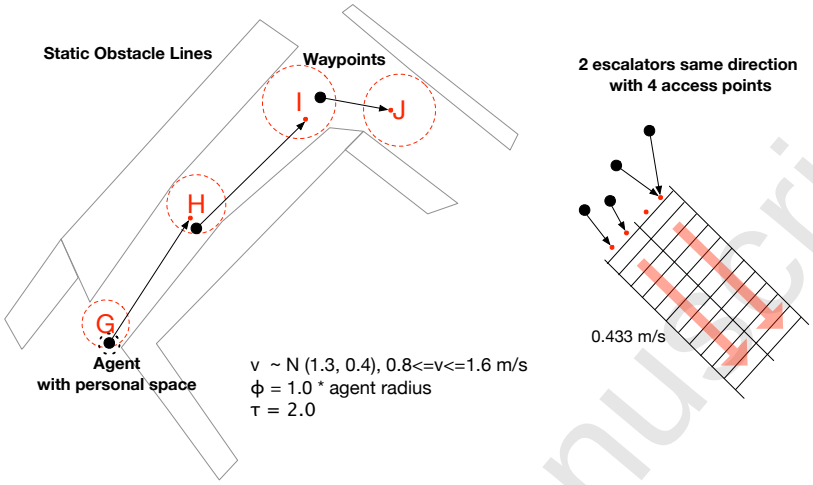


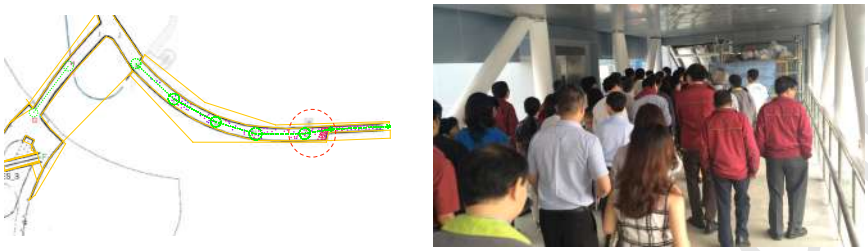
Figure 5: Crowd model details and calibrated parameter values for event planning scenario study.

able speeds. During the rehearsal, only one escalator for each moving direction was used to transport participants between stories as usual. Videos were taken at most path segments, where data is retrieved to validate the simulation results of the crowd model.

As demonstrated in Figure 6, some of the well known crowd phenomena such as the uneven density distribution, vortex and stop and go wave [14, 33] are observed from the simulation results. In addition, it is observed that congestions occur at similar path segments in the simulation and the recorded videos (e.g., segment OP before the staircase as circiled in Figure 6(a) and the actual area in Figure 6(b)). Face validity of the crowd model is thus achieved through producing visually similar crowd dynamics as observed in the recorded videos [1].

Quantitative measures on the generated crowd dynamics are also used to provide statistical validity to the model. We focus on the comparison of the simulation results with the recorded videos through measures of the travel time and densities at each path segment. We count the agents/participants at each segment along time, which is divided by the corresponding area to get the segment density. Due to the spatial constraints, the

[1]The simulation results can be accessed at `https://drive.google.com/open?id=0B8t255d65nTGVS1ST1Vvem15VTg`



(a) Simulation frame (b) Video frame

Figure 6: Face validation of the crowd model through crowd phenomena and congestion similarity

videos were recorded through hand-held cameras along the path with different oblique angles. Not all the segments were fully covered by the videos, we thus merge some path segments (e.g., GH and HI into GHI) according to the video recordings for comparison purposes. We then manually count the participants with an interval of 30 seconds, although some advanced video analytics method such as [34] can be used to automatically retrieve density information from the videos. In the simulation study, the average count of agents at each segment along time are generated from 10 simulation runs (the black line in Figure 7 indicates the sample mean with standard error calculated on 95% confidence interval (C.I)), both density and the travel time of each segment are found consistent with video-retrieved data (the red line with circle marks in Figure 7). It is observed that the total travel time is around 520 seconds and the most dense segments are BC, GHI, NOP and PQ (as highlighted) excluding the escalator segments (as grey-shaded).

5.4. Crowd Control Strategies Generated by MO-CGP Algorithm

For both scenarios, the mutation rate of the MO-CGP algorithm is set to 0.05 across generations. The maximum chromosome length is 200 to limit the search space.

5.4.1. Scenario 1

In this scenario, we set the termination condition as 40 generations. For each generation, 100 individual control strategies are generated as the best non-dominant solutions

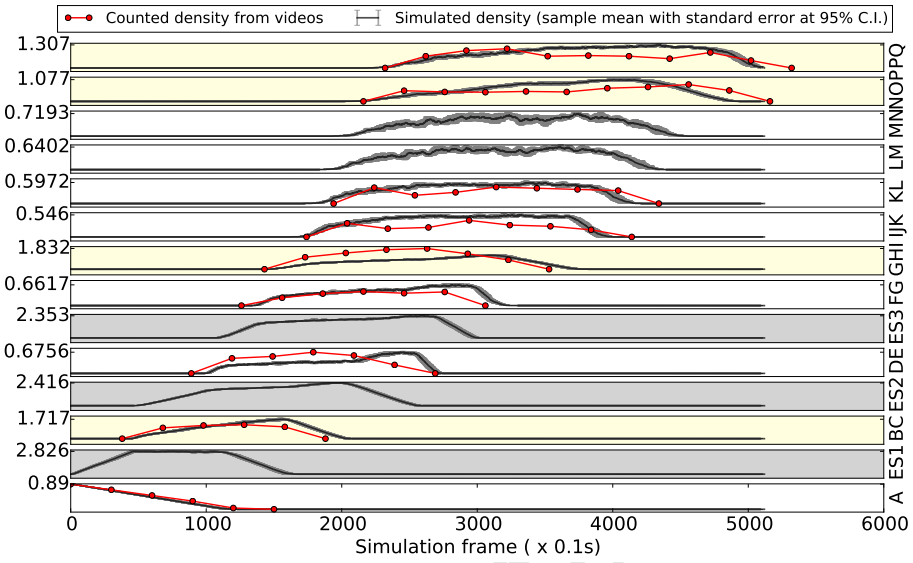


Figure 7: The density distribution at different path segments (or grouped segments) is consistent with the counted participant densities from recorded videos.

based on the mean results of objective values T and T_d from 40 simulation runs with random seeds. The 100 candidate solutions along different generations are plot in Figure 8. The Pareto-front can be identified (as the red line) at generation 40. Figure 8 also shows the result generated without applying crowd control. Note that the hard-coded rule and speculative rule generate results that are much worse than the GP generated candidates, thus they cannot fit in the same scale as shown in the figure.

Our method has outperformed the benchmarking methods in the two objective measures. To further examine the found "optimal" rules, we choose the control strategy resulting in the shortest evacuation time as highlighted by the dashed circle in Figure 8. We generate the truth table of this rule as shown in Figure 9. In the figure, I_0, ..., I_5 indicate the status of local segment density for segment 0, ..., segment 5 respectively. A value of 1 indicates the current segment is overdense, and a value of 0 indicates otherwise. A combination of these variables through logic operators form the condition of the rule. J_0, ..., J_5 indicate the speed reduction action. A value of 0 indicates the speed reduction on the preferred speeds of agents is applied to the agents currently in

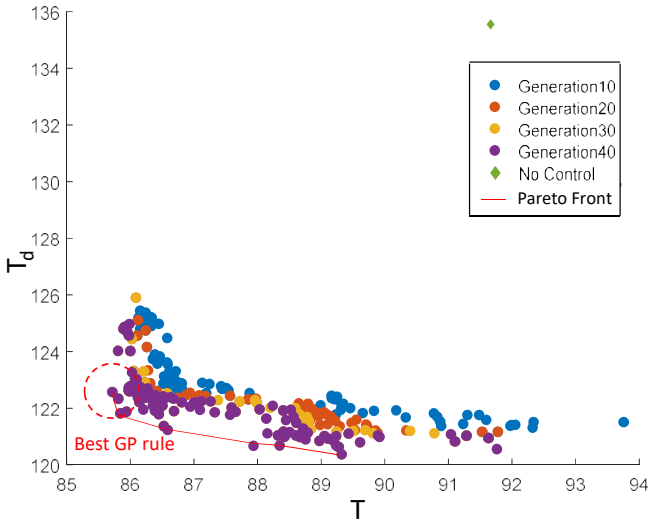


Figure 8: Candidate control strategies along different generations.

the segment, and a value of 1 indicates otherwise. From the rule's truth table, we can identify that a clear "asymmetric" speed reduction control (J) has been adopted by the rule to segments on the left and right sides of the exit as highlight in the truth table. Figure 10 further confirms such a finding through simulation (particularly during the earlier stage of the simulation as shown in figure 10.2), where agents whose preferred speed are reduced are highlighted with a yellow circle.

Through analysis, we find the automatically evolved rule is well aligned with experts' speculation on good crowd control strategies to enhance crowd flow and to reduce the evacuation time as described in [1]. First, Helbing et al. highlighted the "faster is slower" phenomenum caused by the clogging at the exit (hotspot areas corresponding to segment 0 and 1 in our study). Our results (by only reducing the speed of agents at different locations and time) has shown that a corresponding control strategy indeed results in the "slower is faster" effect on the crowd. Secondly, they mentioned that asymmetric crowd split by pillars could result in better crowd flow at the exit to break the clogs. In our case, the evolved rule reduced the preferred speeds of agents in an asymmetric manner for segments on the left/right hand-side of the exit as shown in

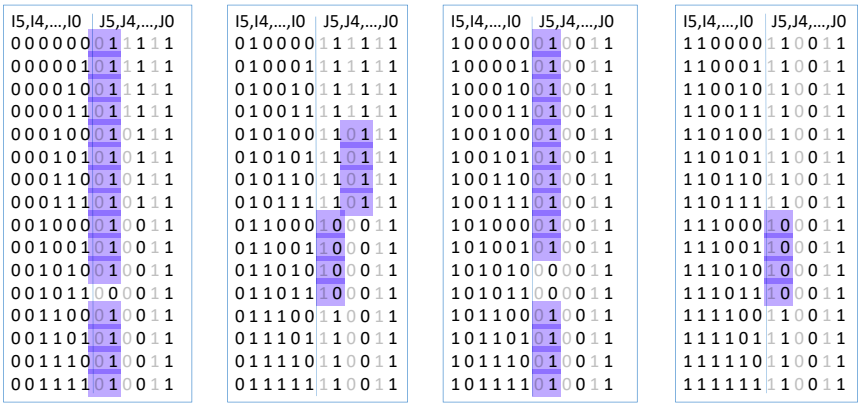


Figure 9: Truth table of the learned rule for the single exit evacuation scenario.

Figure 10. Agents with yellow circle surrounded indicate that their preferred speeds are reduced at the corresponding segments as specified by the optimal rule. By breaking the tie in competing for exit in terms of speeds, the overall flow has been enhanced, achieving a shorter evacuation time. Such results prove that the proposed automatic crowd control framework and the MO-CGP algorithm is effective to find optimal crowd control rules, which are at least on par with years of experience accumulated by domain experts.

5.4.2. Scenario 2

In this scenario, we set the termination condition as 100 generations of evolution, assuming it is a more complex scenario than the first one. For each generation, 30 individual control strategies are generated as the best non-dominant solutions at the current generation. We repeat the evolution process with random seeds for 10 times and found the best 300 solutions at generation 100 as shown in Figure 11. The Pareto-front can be identified as shown in the figure. Individual solutions on the edge of the

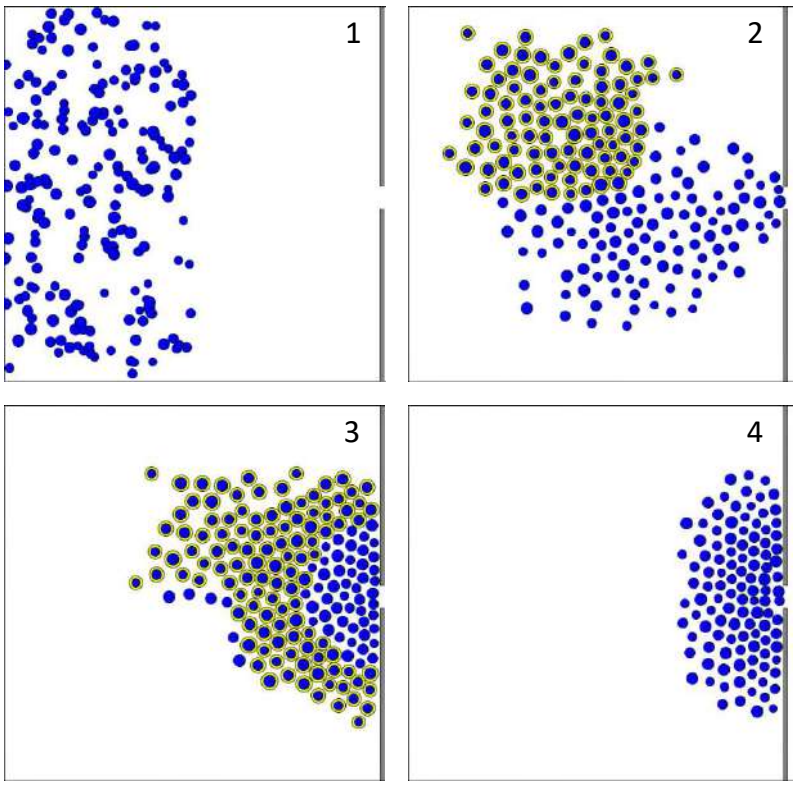


Figure 10: Asymmetric speed reduction is observed from the automatically evolved rule through different simulation steps.

Pareto-front trade off between the two objectives. It provides multiple options for the users to choose crowd control strategies with preferences to a specific objective when there is no specific objective that is obviously more important than the other as in the case of Scenario 1.

For illustration purpose, we choose two optimized candidate control strategies from the Pareto-front as highlighted in Figure 11. CS_1 focuses more on achieving shorter total travel time T and CS_2 results in smaller times of dense segments T_d.

To further examine the inter-dependency of different segment densities on the control rules for each segment, we examine the generated control strategies CS_1 and CS_2

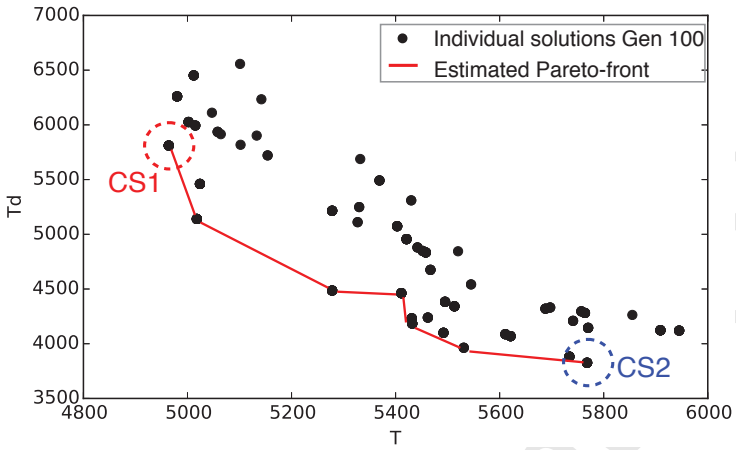


Figure 11: The approximate Pareto-front of the 300 individual solutions at generation 100 of the evolution. Note that some solutions achieve the same results with regard to the two measures.

from the proposed MO-CGP and decode them in terms of scenario-specific parameters (e.g., track usage of the three escalators b_1, b_2, b_3) and control rules (e.g., speed reduction rules on a specific segment based on observed local densities of all segments) as shown in Appendix A.

5.5. Crowd Control Results and Discussion

5.5.1. Scenario 1

To emphasize on the total evacuation time T, we choose the optimal control strategy from the Pareto-front that results in the shortest T to further analyze as highlighted by a dashed circle in the figure. The evacuation time and the total dense segment time in segment 0 and segment 1 of this control strategy are 85.7281 and 122.5500 seconds respectively. The results, together with the results of other three benchmarking methods are listed in Table 1. It is noted that the results obtained by the GP method is better than all the other three benchmarking methods.

As results shown in Table 1, our framework has found better rules with the strict constraints and very limited action (speed reduction) in this scenario. We note that the improvement of the total evacuation time is approximately 6%, 50% and 38% for the GP evolved rule compared to the benchmarking methods without control, with hard-

Table 1: The simulation statistics of the proposed MO-CGP algorithm, comparing with three benchmarking methods.

	Total evacuation time, T	Total dense segment time, T_d
Optimal rule found by GP	85.7281s	122.5500s
Model without control	91.6631s	135.5531s
Hard-coded rule	169.2781s	229.7553s
Speculative rule	137.5725s	219.4922s

coded control rule and the speculative control rule respectively. We think the improve of 6% as compared to the no-control case may attribute to the fact that the area is quite dense overall, and there is no alternative path for the agents except for moving towards the exit. Consequently, without any control, segments 0 and segment 1 will become overly congested as time goes by. The poor results from the two hard-coded rules imply that naive control based on only the local density of the current segments fails to consider the density propagation across segments caused by the movement of agents; whereas this has been included in the GP-evolved rules.

5.5.2. Scenario 2

To test the effectiveness of the proposed framework, we use two baseline models to compare with the generated strategies by MO-CGP: 1) the simulation model without any crowd control; 2) the simulation model with a hard-coded control strategy: "if a segment is over-dense (using the threshold of 1.2 pax/sqm because the space we study in this scenario is much larger than scenario 1, and the threshold is identified through calibration of the model as discussed in Section 5.3), then speed reduction control on the current segment is applied". For each model, 10 simulation runs with different random seeds were conducted and the results (sample mean and standard error with 95% confidence interval) for the most dense segments are shown in Figure 12.

Compared to the baseline model without any external control on agents' movement, the MO-CGP generated control strategies CS_1 and CS_2 have improved the overall crowd dynamics: control strategy CS_1 has resulted in a shorter travel time, while peak density of the most dense segments (i.e., BC, GHI, and NOP) has been reduced

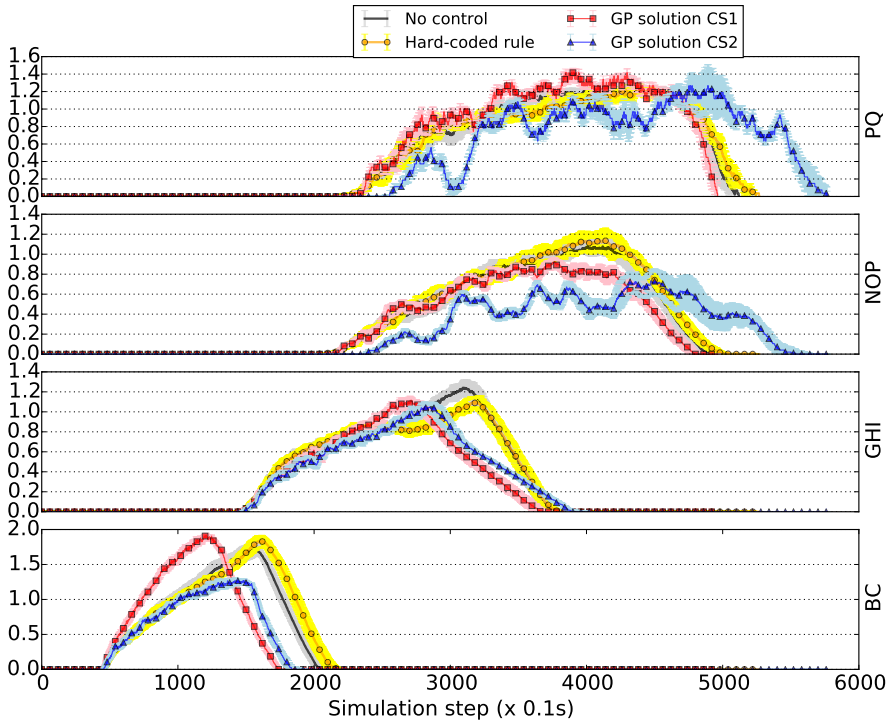


Figure 12: Simulation results and comparison of different crowd control strategies at the most dense path segments (excluding the escalator segments).

by approximately 20% with control strategy CS_2. On the contrary, the hard-coded strategy does not have obvious improvement from the baseline model without control strategy. This is counter-intuitive as marshals in real life situations usually perform crowd control based on the current segment's density. However, the results show ineffectiveness of this simple strategy. We suspect the failure of such hard-coded strategies is due to their lack of considerations on the inter-dependencies of the crowd dynamics across different path segments. Such inter-dependency is related to the spatial constraints of the environment as well as the interactions among pedestrians with different moving speeds. For example, the inflow and outflow of an escalator will certainly affect the densities of crowds at the subsequent segments that are even further away from the escalator. Hard-coded rules can hardly exploit such inter-dependencies; whereas GP-based optimization methods overcome this shortage through efficient search over

the whole solution space.

In summary, the results indicate that the proposed GP based automatic framework is capable of improving the crowd dynamics through evolving both parameter and rule structure despite very strict constraints (e.g., allowing only to slow down agents in order to make them pass through the whole path faster).

5.5.3. Application of the Proposed Framework

The proposed framework can be deployed in real applications through: 1) deploying a number of sensors to retrieve segment densities in real-time; 2) deploying a marshal at each segment along the path; 3) training the GP based on either rehearsal event or simulation data; and 4) generating guidance to regulate the crowd flow for each marshal according to the GP and sensed segment densities in real-time.

One general limitation of a GP-based framework is that the evolved rules can be very complex with many levels of combinations as shown in Appendix A. Further investigation of the evolved rules can provide insights on more important segments over others. However, it is a very challenging task to simplify such formula even for combinations of only logic operations with a large number (e.g., larger than 4) of features. In this paper, we propose to address this challenge through a machine learning approach, where the rule analysis is converted to a feature selection problem. Our objective is to find the most important segments (features) out of the complex combinations that fire the speed reduction rule for a particular segment. To achieve this, we first generate all possible value combinations of a control rule as shown in Appendix A, and use the logistic combination to generate the action results accordingly. That is, we generate sample pairs for each control rule as $< x_i, y_i >_{i=1}^{n}$, where x_i is generated from $x_{1,2,...17}$. x_i = 0 or 1 representing whether a path segment has over-densed crowd at the moment, and y_i = 0 or 1 representing whether the current control strategy needs to reduce the preferred speed of agents at segment i (corresponding to $rule_i$). We then transform all the records with y_i=0 to y_i=-1 to facilitate calculation in the feature selection process. Then we use feature generation machine [35] as shown in Equation 12 to extract the feature importance of x_i that results in speed reduction (y_i=1):

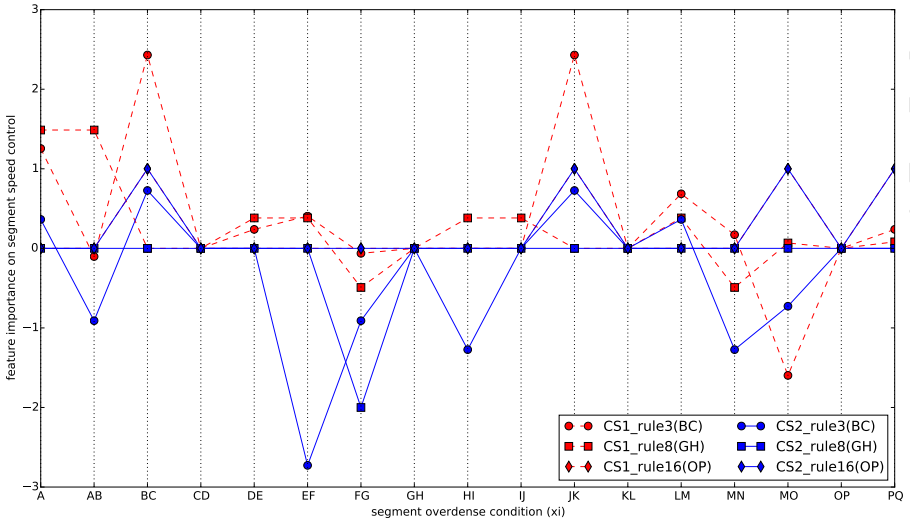


Figure 13: Feature importance of the generated control rules 3, 8, 16 for both control strategies CS_1 and CS_2. The higher positive value indicates higher importance of the feature with positive correlation to the result (Y=1).

$$\min_{\mathbf{d}} \min_{\mathbf{w},\rho,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 - \rho \qquad (12)$$
$$\text{s.t.} \qquad y_i \mathbf{w}^\top (\mathbf{x_i} \odot \mathbf{d}) \geq \rho - \xi_i$$

where $\mathbf{w}$ is the feature weights and $\mathbf{d} \in \{0,1\}^d$ is the feature selection indicator (i.e., 1 means *select* and 0 means *discard*) , ξ_i is slack error for data $\mathbf{x_i}$, ρ is the linear model bias.

For illustration purpose, we plot the feature importance for $rule_3$, $rule_8$ and $rule_{16}$ corresponding to the control rules over the most dense segments BC, GH and OP as found from the previous simulation. In general, local densities at segment BC and JK are more important indicators that lead to the speed reduction control at the most dense segments as shown in Figure 13, which can be used to improve the cost-efficiency of the actual system by reducing the number of necessary sensors. For example, there are 11 features in the original rule construction of the generated $rule_3$. Sensors should be allocated at all these 11 segments to retrieve density information. Through the fea-

ture importance analysis, we have identified that densities at segment BC and JK are significantly more important that others. Thus, sensors can be allocated mainly to the limited but important segments in practise. We also note that segments (e.g., JK) that are further away can be important to determine when to fire a speed reduction rule for a specific segment (e.g., BC). The feature importance analysis has affirmed our hypothesis that the inter-relation of different features can be complex to form an optimal control rule. The analysis suggests that the proposed GP-based automatic control system is much more feasible to achieve better crowd control than hard-coded rule systems, which are generally based on past experience.

6. Conclusions and Future Work

In this paper, we have proposed an automatic crowd control framework based on genetic programming techniques. The framework consists of a simulation engine to generate crowd dynamics and an optimization system based on genetic programming. The framework aims to provide the "optimal" crowd control strategies in order to generate crowd dynamics with multiple objectives through simultaneous evolution of scenario-specific parameters and control rules that affect the agent's movement explicitly in the simulation model.

From the base scenaio of crowd control in a room evacuation case, the proposed framework has found rules that result in shorter evacuation time with less density at the hotspot areas in front of the exit (to avoid pushing). Although the improvement is limited due to the strict constraints and very limited action choice to construct the rule, we demonstrated that the proposed framework can automatically identify crowd control strategies that are well aligned with the speculative recommendation from experts based on years of experience.

The scenario study of an event planning shows that this framework is promising to provide a set of better control strategies to influence crowd dynamics more effectively in a more complex and practical scenario. We further provide a method to analyze and identify the most important components in the optimized rules based on the feature selection machine, which overcomes the simplification challenge of the usually

over-complex rules found by genetic programming. With proper sensor resources, the proposed framework can be applied in different practical crowd control applications.

Some limitations of the proposed framework and the MO-CGP algorithm have also been identified in this study. First, we feel the proposed framework may be more suitable to handle crowd control in a scenario with flexible intervention on both path planning and moving speeds. We will further verify this assumption using more scenario studies. Secondly, there are trade-off between optimization convergency and the population size. In a more complex scenario for practical applications, a larger population size may be needed, which will request for much more computational resources. To conquer this challenge, we will explore hierarchical approaches for modeling, and thus simplify some model components to improve the efficiency.

Acknowledgement

Nan Hu and Wentong Cai would like to acknowledge the support from IHPC-NTU Joint R&D Project on "Symbiotic Simulation and Video Analysis of Crowds"; Jinghui Zhong is supported by the National Natural Science Foundation of China (Grant No.61602181) and Fundamental Research Funds for the Central Universities (Grant No. 2017ZD053). The authors would like to thank Dr. Tan Singkuang and Dr. Yi Wenchao for their support and contribution to the paper.

References

[1] D. Helbing, I. Farkas, T. Vicsek, Simulating dynamical features of escape panic, Nature 407 (6803) (2000) 487–490.

[2] S. J. Guy, M. C. Lin, D. Manocha, Modeling collision avoidance behavior for virtual humans, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 2-Volume 2, 2010, pp. 575–582.

[3] D. Grieger, An overview of crowd control theory and considerations for the employment of non-lethal weapons, Tech. Rep. DSTO-GD-0373, Land Operations

Division System Sciences Laboratory, Australian Government, DSTO, Australia (2004).

[4] W. Shao, D. Terzopoulos, Autonomous pedestrians, in: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2005, pp. 19–28.

[5] N. Hu, M. H. Lees, S. Zhou, A pattern-based modeling framework for simulating human-like pedestrian steering behaviors, in: Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology, ACM, 2013, pp. 179–188. doi:10.1145/2503713.2503723.
URL http://doi.acm.org/10.1145/2503713.2503723

[6] M. Ghallab, D. G. Allard, $A_{epsilon}$ - an efficient near admissible heuristic search algorithm, in: Proceedings of the 8th International Joint Conference on Artificial Intelligence. Karlsruhe, FRG, August 1983, 1983, pp. 789–791.

[7] S. J. Rymill, N. A. Dodgson, Psychologically-based vision and attention for the simulation of human behaviour, in: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, ACM, 2005, pp. 229–236.

[8] J. Van den Berg, M. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: Proceedings of the 2008 IEEE International Conference on Robotics and Automation, IEEE, 2008, pp. 1928–1935.

[9] D. Helbing, P. Molnar, Social force model for pedestrian dynamics, Physical review E 51 (5) (1995) 4282.

[10] J. Zhong, L. Luo, W. Cai, M. Lees, Automatic rule identification for agent-based crowd models through gene expression programming, in: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, 2014, pp. 1125–1132.

[11] J. Zhong, N. Hu, W. Cai, M. Lees, L. Luo, Density-based evolutionary framework for crowd model calibration, Journal of Computational Science 6 (2015) 11–22.

[12] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, 1st Edition, Wiley, 2001.

[13] M. Moussaid, D. Helbing, G. Theraulaz, How simple rules determine pedestrian behavior and crowd disasters, Proceedings of National Academic Sciebce, U.S.A 108 (17) (2011) 6884–8.

[14] S. J. Guy, S. Curtis, M. C. Lin, D. Manocha, Least-effort trajectories lead to emergent crowd behaviors, Physical Review E: Statistical, Nonlinear, and Soft Matter Physics 85 (1 Pt 2) (2012) 016110.

[15] A. Lerner, Y. Chrysanthou, D. Lischinski, Crowds by example, in: Computer Graphics Forum, Vol. 26, Wiley Online Library, 2007, pp. 655–664.

[16] A. Johansson, D. Helbing, P. K. Shukla, Specification of the social force pedestrian model by evolutionary adjustment to video tracking data, Advances in complex systems 10 (supp02) (2007) 271–288.

[17] T.-Y. Li, C.-C. Wang, An Evolutionary Approach to Crowd Simulation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, Ch. Autonomous Robots and Agents, pp. 119–126. doi:10.1007/978-3-540-73424-6_14.
URL http://dx.doi.org/10.1007/978-3-540-73424-6_14

[18] D. Wolinski, S. J Guy, A.-H. Olivier, M. Lin, D. Manocha, J. Pettré, Parameter estimation and comparative evaluation of crowd simulations, in: Computer Graphics Forum, Vol. 33, Wiley Online Library, 2014, pp. 303–312.

[19] J. M. Kenny, C. Mcphail, P. Waddington, S. Heal, S. Ijames, D. N. Farrer, J. Taylor, D. Odenthal, Crowd behavior, crowd control, and the use of non-lethal weapons, Tech. Rep. OMB No. 0704-0188, Institue for non-lethal defense technologies, The Pennsylvania State University, PA 16804-3000 (January 2001).

[20] B. D. Eldridge, A. A. Maciejewski, Using genetic algorithms to optimize social robot behavior for improved pedestrian flow, in: 2005 IEEE International Conference on Systems, Man and Cybernetics, Vol. 1, 2005, pp. 524–529 Vol. 1. doi:10.1109/ICSMC.2005.1571199.

[21] J. Schubert, R. Suzic, Decision support for crowd control: Using genetic algorithms with simulation to learn control strategies, in: MILCOM 2007 - IEEE Military Communications Conference, 2007, pp. 1–7. doi:10.1109/MILCOM.2007.4455059.

[22] N. Hu, J. Decraene, W. Cai, Effective crowd control through adaptive evolution of agent-based simulation models, in: Simulation Conference (WSC), Proceedings of the 2012 Winter, 2012, pp. 1–12. doi:10.1109/WSC.2012.6465040.

[23] J. Decraene, Y. T. Lee, F. Zeng, M. Chandramohan, Y. Y. Cheng, M. Y. H. Low, Evolutionary design of agent-based simulation experiments, in: The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3, AAMAS '11, 2011, pp. 1321–1322.

[24] D. Helbing, A. Johansson, H. Z. Al-Abideen, Dynamics of crowd disasters: An empirical study, Physical review E 75 (4) (2007) 046109.

[25] J. Fruin, Pedestrian Planning and Design, Metropolitan Association of Urban Designers and Environmental Planners, Inc, New York, 1971b.

[26] G. K. Still, Crowd dynamics, Ph.D. thesis, University of Warwick, Department of Mathematics (2000).
URL http://webcat.warwick.ac.uk/record=b1371042~S1

[27] A. Seyfried, A. Schadschneider, Fundamental diagram and validation of crowd models, in: Proceedings of the 8th International Conference on Cellular Aotomata for Reseach and Industry, Vol. 5191 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 563–566. doi:10.1007/978-3-540-79992-4_77.
URL http://dx.doi.org/10.1007/978-3-540-79992-4_77

[28] D. Helbing, P. Mukerji, Crowd disasters as systemic failures: analysis of the love parade disaster, EPJ Data Science 1 (1) (2012) 1–40. doi:10.1140/epjds7.
URL http://dx.doi.org/10.1140/epjds7

[29] J. Miller, P. Thomson, Cartesian genetic programming, in: Genetic Programming, Vol. 1802 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2000, pp. 121–132. doi:10.1007/978-3-540-46239-2_9.
URL http://dx.doi.org/10.1007/978-3-540-46239-2_9

[30] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii, in: Parallel Problem Solving from Nature PPSN VI, Vol. 1917 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2000, pp. 849–858.

[31] A. Schadschneider, W. Klingsch, H. Klpfel, T. Kretz, C. Rogsch, A. Seyfried, Evacuation dynamics: Empirical results, modeling and applications, in: Encyclopedia of Complexity and Systems Science, Springer New York, 2009, pp. 3142–3176. doi:10.1007/978-0-387-30440-3_187.
URL http://dx.doi.org/10.1007/978-0-387-30440-3_187

[32] Y. Tanaboriboon, S. S. Hwa, C. H. Chor, Pedestrian characteristics study in singapore, Journal of Transportation Engineering 112 (3) (1986) 229–235.

[33] J. Berrou, J. Beecham, P. Quaglia, M. Kagarlis, A. Gerodimos, Calibration and validation of the legion simulation model using empirical data, in: Pedestrian and Evacuation Dynamics, Springer Berlin Heidelberg, 2007, pp. 167–181.

[34] W. Ge, R. Collins, R. Ruback, Vision-based analysis of small groups in pedestrian crowds, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (5) (2012) 1003–1016. doi:10.1109/TPAMI.2011.176.

[35] M. Tan, I. W. Tsang, L. Wang, Towards ultrahigh dimensional feature selection for big data, Journal of Machine Learning Research 15 (2014) 1371–1429.
URL http://jmlr.org/papers/v15/tan14a.html

Appendix A. Decoded control strategies CS_1 and CS_2 through MO-CGP in Scenario 2.

Control Strategy	CS_1	CS_2
b_1, b_2, b_3	$1, 1, 1$	$0, 1, 0$
$rule_1(Y_1)$	$(x_{12}) \vee (\neg((x_2) \vee (x_7)) \wedge (\neg(x_{13}) \wedge (x_1)))$	$((\neg(x_{13}) \wedge (\neg(x_5) \vee (x_5))) \vee (\neg(x_1) \wedge ((x_2) \vee (x_7)))) \wedge ((x_{15}) \vee (x_{14}))$
$rule_2(Y_2)$	$\neg((((\neg(x_{13}) \wedge (x_1)) \vee (x_{14})) \wedge (\neg(x_{15}) \vee (x_{17}))) \vee (\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))))) \vee (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))$	$(\neg(((\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)))) \vee (\neg(x_6) \wedge ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))))) \wedge ((\neg(x_{10}) \wedge ((\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))))) \vee (\neg(((\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)))) \vee (\neg(x_6) \wedge ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))))) \wedge ((\neg(x_{10}) \wedge ((\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))) \wedge ((x_9) \vee ((x_2) \vee (x_7))))) \vee (\neg(x_{13}) \wedge (\neg(x_5) \vee (x_5))))) \vee ((\neg(x_{16}) \vee (\neg(x_{10}) \vee (\neg(\neg(x_2) \vee (x_{12})) \wedge (x_{17})))) \wedge ((x_{12}) \vee (\neg((x_2) \vee (x_7)) \wedge (x_6))))))$
$rule_3(Y_3)$	$((\neg(\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)) \wedge ((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1)))) \wedge ((((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))) \wedge (x_6)) \vee (\neg((x_2) \vee (x_7)) \wedge (\neg(x_{13}) \wedge (x_1))))) \vee (\neg((\neg(x_{15}) \vee (x_{17})) \vee (\neg(x_{11}) \vee (x_3))) \wedge (x_{15}))$	$\neg((\neg(((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))) \wedge ((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1)))) \wedge ((((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))) \vee (\neg((x_2) \vee (x_7)) \wedge (x_6)))) \wedge (\neg((\neg(x_{15}) \vee (x_{17})) \vee (\neg(x_{11}) \vee (x_3))) \wedge (x_{15}))$
$rule_4(Y_4)$	x_8	$(((x_9) \vee ((x_2) \vee (x_7))) \wedge (\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))))) \wedge (\neg(x_8) \wedge (\neg(x_{11}) \vee (x_3)))$
$rule_5(Y_5)$	$(\neg(\neg(x_2) \vee ((x_9) \vee ((x_2) \vee (x_7)))) \wedge (\neg(x_5) \vee (x_5))) \vee ((\neg(x_6) \wedge ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5)))) \wedge (\neg(x_{13}) \wedge (x_1)))$	$\neg(\neg(\neg(((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (\neg(\neg((x_2) \vee (x_7)) \wedge (x_6)) \vee ((\neg(\neg(x_{13}) \wedge (x_1)) \vee (x_{14})) \wedge ((x_2) \vee (x_7))))) \vee (\neg(((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))) \wedge ((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))))) \wedge ((\neg(x_1) \wedge ((x_2) \vee (x_7))) \wedge ((\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)))) \vee (\neg(x_6) \wedge ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5)))))))) \vee ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5)))$

$rule_6(Y_6)$	$lnot(\neg(x_{14}) \vee (x_9)) \wedge (\neg((((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9)) \vee (\neg(x_{14}) \vee (x_9))) \vee ((\neg(x_1) \wedge ((x_2) \vee (x_7))) \wedge ((\neg(x_{10}) \wedge ((\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))) \wedge ((x_9) \vee ((x_2) \vee (x_7))))) \vee (\neg(\neg(x_2) \vee ((x_9) \vee ((x_2) \vee (x_7)))) \wedge (\neg(x_5) \vee (x_5))))))$	$\neg(\neg(\neg(x_1) \wedge ((x_2) \vee (x_7))) \wedge (x_7)) \wedge (\neg((((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9)) \vee (\neg(x_{14}) \vee (x_9))) \vee (((\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)))) \vee (\neg(x_6) \wedge ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))))) \wedge ((\neg(x_{10}) \wedge ((\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))) \wedge ((x_9) \vee ((x_2) \vee (x_7))))) \vee (\neg(x_{13}) \wedge (\neg(x_5) \vee (x_5))))))$
$rule_7(Y_7)$	$(((\neg(x_5) \vee (x_5)) \vee (\neg(x_{12}) \vee (x_{12}))) \vee (\neg(\neg(\neg(x_{14}) \vee (x_9)) \wedge (\neg((x_9) \vee (x_{11})) \wedge (x_5))) \wedge (\neg(((x_{15}) \vee (x_{14})) \vee (\neg((x_{15}) \vee (x_{14})) \vee (x_8))) \vee ((\neg(\neg(x_{13}) \wedge (x_1)) \wedge (x_{17})) \wedge ((x_{15}) \vee (x_{14}))))) \wedge (\neg(x_2) \vee ((x_9) \vee ((x_2) \vee (x_7))))$	$\neg(x_{12}) \wedge ((x_3) \vee (\neg(\neg(\neg(x_6) \wedge ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5)))) \wedge (\neg(x_{13}) \wedge (x_1))) \vee ((x_9) \vee (x_{11}))))$
$rule_8(Y_8)$	$(\neg(\neg(\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \vee (\neg(\neg((x_2) \vee (x_7)) \wedge (\neg(x_{13}) \wedge (x_1))) \vee (((\neg(x_{13}) \wedge (x_1)) \vee (x_{14})) \wedge (\neg(x_{15}) \vee (x_{17}))))) \vee (\neg(\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)) \wedge ((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))))) \wedge (\neg(x_2) \wedge ((\neg(x_{10}) \wedge ((\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))) \wedge ((x_9) \vee ((x_2) \vee (x_7))))) \vee (\neg(\neg(x_2) \vee ((x_9) \vee ((x_2) \vee (x_7)))) \wedge (\neg(x_5) \vee (x_5)))))) \vee ((((\neg(x_{13}) \wedge (x_1)) \vee (x_{14})) \wedge (\neg(x_{15}) \vee (x_{17}))) \vee (\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)))))$	$((\neg(((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (\neg(\neg((x_2) \vee (x_7)) \wedge (x_6)) \vee ((\neg(\neg(x_{13}) \wedge (x_1)) \vee (x_{14})) \wedge ((x_2) \vee (x_7))))) \vee (\neg(((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))) \wedge ((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))))) \wedge (\neg(x_2) \wedge ((\neg(x_{10}) \wedge ((\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1))) \wedge ((x_9) \vee ((x_2) \vee (x_7))))) \vee (\neg(x_{13}) \wedge (\neg(x_5) \vee (x_5))))) \vee (((\neg(\neg(x_{13}) \wedge (x_1)) \vee (x_{14})) \wedge ((x_2) \vee (x_7))) \vee (\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)))))$
$rule_9(Y_9)$	$\neg(x_7) \vee (\neg(x_6) \vee (\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)))$	$(\neg(x_5) \vee (x_5)) \wedge (x_{16})$
$rule_{10}(Y_{10})$	$(\neg(\neg((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))) \wedge (x_1)) \wedge ((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1)))) \wedge ((((x_{14}) \wedge (\neg(x_{13}) \wedge (x_1))) \wedge (x_6)) \vee (\neg((x_2) \vee (x_7)) \wedge (\neg(x_{13}) \wedge (x_1))))$	$((\neg(x_2) \vee (x_{12})) \wedge (\neg(\neg(x_{11}) \vee (x_3)) \vee (\neg((x_{16}) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))) \vee ((\neg(x_{13}) \wedge (x_1)) \vee (\neg(x_5) \vee (x_5))))) \wedge ((x_{10}) \wedge (\neg(x_4) \vee (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))))$
$rule_{11}(Y_{11})$	$\neg(\neg(x_{13}) \wedge (x_1)) \wedge ((\neg((x_2) \vee (x_7)) \wedge (\neg(x_{13}) \wedge (x_1))) \wedge ((x_{16}) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))))$	$\neg(\neg(x_{13}) \wedge (x_1)) \wedge ((\neg((x_2) \vee (x_7)) \wedge (x_6)) \wedge ((x_{16}) \wedge (((x_9) \vee ((x_2) \vee (x_7))) \wedge (x_9))))$
$rule_{12}(Y_{12})$	$(\neg(x_5) \vee (x_5)) \vee (\neg(x_{12}) \vee (x_{12}))$	$(\neg(x_5) \vee (x_5)) \vee (\neg(x_2) \vee (x_{12}))$

$rule_{13}(Y_{13})$	$\neg((x_{10})\wedge(\neg((((\neg(x_{13})\wedge(x_1))\vee(x_{14}))\wedge(\neg(x_{15})\vee(x_{17})))\vee(\neg(x_7)\vee(\neg(x_6)\vee(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1)))))\vee(((x_9)\vee((x_2)\vee(x_7)))\wedge(x_9))))\wedge(\neg(\neg(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1))\wedge((x_{14})\wedge(\neg(x_{13})\wedge(x_1))))\wedge(x_{14}))$	$\neg((\neg(\neg(x_6)\wedge((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5))))\wedge(\neg(x_{13})\wedge(x_1)))\wedge((x_2)\vee(x_7)))\wedge(((x_{14})\wedge(\neg(x_{13})\wedge(x_1)))\wedge(((x_9)\vee((x_2)\vee(x_7)))\wedge(x_9)))$
$rule_{14}(Y_{14})$	$(\neg(\neg(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\vee(\neg(\neg((x_2)\vee(x_7))\wedge(\neg(x_{13})\wedge(x_1)))\vee(((\neg(x_{13})\wedge(x_1))\vee(x_{14}))\wedge(\neg(x_{15})\vee(x_{17})))))\vee(\neg(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1))\wedge((x_{14})\wedge(\neg(x_{13})\wedge(x_1)))))\wedge(\neg(x_2)\wedge((\neg(x_{10})\wedge((\neg(x_6)\vee(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1)))\wedge((x_9)\vee((x_2)\vee(x_7)))))\vee(\neg(\neg(x_2)\vee((x_9)\vee((x_2)\vee(x_7))))\wedge(\neg(x_5)\vee(x_5))))))\vee((((\neg(x_{13})\wedge(x_1))\vee(x_{14}))\wedge(\neg(x_{15})\vee(x_{17})))\vee(\neg(x_7)\vee(\neg(x_6)\vee(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1)))))$	$((\neg(((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(\neg(\neg((x_2)\vee(x_7))\wedge(x_6))\vee((\neg(\neg(x_{13})\wedge(x_1))\vee(x_{14}))\wedge((x_2)\vee(x_7)))))\vee(\neg(((x_{14})\wedge(\neg(x_{13})\wedge(x_1)))\wedge(((x_9)\vee((x_2)\vee(x_7)))\wedge(x_9)))\wedge((x_{14})\wedge(\neg(x_{13})\wedge(x_1)))))\wedge(\neg(x_2)\wedge((\neg(x_{10})\wedge((\neg(x_6)\vee(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1)))\wedge((x_9)\vee((x_2)\vee(x_7)))))\vee(\neg(x_{13})\wedge(\neg(x_5)\vee(x_5))))))\vee(((\neg(\neg(x_{13})\wedge(x_1))\vee(x_{14}))\wedge((x_2)\vee(x_7)))\vee(\neg(x_7)\vee(\neg(x_6)\vee(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1)))))$
$rule_{15}(Y_{15})$	$(((x_9)\vee((x_2)\vee(x_7)))\wedge(x_9))\vee(\neg(x_{14})\vee(x_9))$	$(((x_9)\vee((x_2)\vee(x_7)))\wedge(x_9))\vee(\neg(x_{14})\vee(x_9))$
$rule_{16}(Y_{16})$	$(\neg(x_{15})\vee(x_{17}))\vee(\neg(x_{11})\vee(x_3))$	$(\neg(x_{15})\vee(x_{17}))\vee(\neg(x_{11})\vee(x_3))$
$rule_{17}(Y_{17})$	$(\neg(\neg((\neg(x_{13})\wedge(x_1))\vee(\neg(x_5)\vee(x_5)))\wedge(x_1))\wedge((x_{14})\wedge(\neg(x_{13})\wedge(x_1))))\wedge((((x_{14})\wedge(\neg(x_{13})\wedge(x_1)))\wedge(x_6))\vee(\neg((x_2)\vee(x_7))\wedge(\neg(x_{13})\wedge(x_1))))$	$(\neg(((x_{14})\wedge(\neg(x_{13})\wedge(x_1)))\wedge(((x_9)\vee((x_2)\vee(x_7)))\wedge(x_9)))\wedge((x_{14})\wedge(\neg(x_{13})\wedge(x_1))))\wedge((((x_{14})\wedge(\neg(x_{13})\wedge(x_1)))\wedge(((x_9)\vee((x_2)\vee(x_7)))\wedge(x_9)))\vee(\neg((x_2)\vee(x_7))\wedge(x_6)))$

1. We propose an automatic framework based on genetic programming to generate crowd control strategies.

2. We propose a Multi-Objective Cartesian Genetic Programming (MO-CGP) algorithm capable of evolving not only parameters but also rule structures so as to find optimal control strategies for multi-objective optimization.

3. We demonstrate the effectiveness of the framework in a real life crowd control scenario to guide approximately 400 participants to pass through a multi-story building with shorter travel time and reduced congestion at hotspot segments, comparing to several common baseline crowd models.

4. We propose a rule analysis method through feature selection machine to identify a limited number of most important features in the evolved rules from GP algorithm effectively.