Optimal Control of Pedestrian Flows by Congestion Forecasts Satisfying User Equilibrium Conditions

Yamada, Hiroaki
Fujitsu Laboratories Ltd.

Kamiyama, Naoyuki
Institute of Mathematics for Industry, Kyushu University

出版情報：International Conference on Principles and Practice of Multi-Agent Systems: PRIMA 2020 (Lecture Notes in Computer Science). 12568, pp.299-314, 2021-02-14. Springer
バージョン：
権利関係：

Optimal Control of Pedestrian Flows by Congestion Forecasts Satisfying User Equilibrium Conditions

Hiroaki Yamada[1] and Naoyuki Kamiyama[2,3]

[1]Fujitsu Laboratories Ltd., Kawasaki, Japan
yamadah@fujitsu.com
[2] Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan
[3] JST, PRESTO, Kawaguchi, Japan
kamiyama@imi.kyushu-u.ac.jp

Abstract. Reducing congestion is one of the most important issues in theme park management. Optimization algorithms for reducing congestion in theme parks using simulation optimization methods have been proposed. In many existing methods, theme park managers directly regulate the movement of visitors. However, restricting the freedom to wander considerably reduces the visitor satisfaction. Thus, when controlling congestion in theme parks, we must consider the trade-off between reducing congestion and restricting freedom. In this paper, we propose an indirect control method for pedestrian flows using congestion forecasts and information distribution. Specifically, we propose a simulation-based heuristic algorithm for the problem of finding an optimal information distribution policy for congestion forecasts satisfying user equilibrium conditions.

1 Introduction

Congestion is ubiquitous in theme parks, and is chiefly caused by popular rides and attractions. Reducing congestion can reduce the waiting time of visitors, thus improving visitor satisfaction. Furthermore, because congestion can cause accidents, it is one of the most important issues in theme park management. Of course, it is not difficult to observe that we can easily reduce congestion by regulating the movement of visitors. However, restricting the freedom to wander considerably reduces visitor satisfaction. Thus, when controlling congestion in theme parks, we must consider the trade-off between reducing congestion and restricting freedom. This is one of the characteristic problems in the optimal control of pedestrian flows.

Recently, some theme parks have introduced information distribution systems to reduce congestion without restricting the freedom to wander and reducing visitor satisfaction. For example, some theme parks disseminate the current waiting times of attractions through smartphone apps to reduce huge congestion at popular attractions. However, distributing the current congestion information causes visitors to react to the information and avoid crowded attractions. Consequently, although congestion is reduced at the places that were expected to

be crowded, new congestion occurs in the places that were not expected to be crowded. Furthermore, such a situation may repeat during a day. This situation is called the *hunting phenomenon* (see [1]). This implies that the conventional information distribution policy may not optimize the entire system.

Yamada and Kamiyama [2] proposed a framework for an information distribution method for avoiding the hunting phenomenon. This method supplies congestion forecasts, which satisfy user equilibrium conditions for a part of the visitors (see Figure 1). A *user equilibrium* is a situation where no one can improve her/his utility by changing only her/his decision (i.e., it is a Nash equilibrium). In [2], a user equilibrium is specifically defined as the situation where the future predicted by the forecast will materialize if the visitors follow the congestion information. Because the future that the forecast predicts will materialize, the visitors will have no regrets if they decide their actions following her/his utility and the given information. Therefore, if we can supply a user equilibrium as the congestion forecast, we can facilitate the materialization of the forecast. Yamada and Kamiyama [2] call this reproducible congestion forecast a *congestion forecast satisfying the user equilibrium conditions*. The algorithm proposed in [2] finds a congestion forecast that approximately satisfies the user equilibrium conditions by focusing on the state after people have visited a place infinitely times.

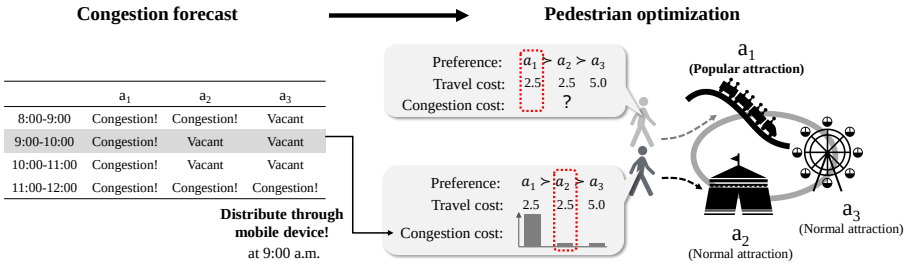

Fig. 1. Illustration of pedestrian flow control by congestion information. We attempt to optimize pedestrian flows by distributing congestion information to some of the visitors. A visitor who does not have any information about the current congestion condition selects the most preferred attraction. On the other hand, a visitor who has information about the current congestion condition considers the congestion and selects the second-best attraction.

The aim of this study is to formalize and evaluate the information distribution method proposed in [2] based on simulation optimization [3]. Simulation optimization is a methodology for optimizing complex systems by mapping the target system to a simulation and optimizing the simulation by methods such as black-box optimization algorithms [4]. Following a review of the related work, we explain our agent-based simulation model of a theme park and define our problem in Section 2. In Section 3, we explain our algorithm. In Section 4, we discuss

the results of the computational experiments. Finally, Section 5 concludes this paper.

We summarize the highlights of our algorithm. First, our algorithm is inspired by the algorithms for computing an equilibrium traffic assignment in road networks [5]. Second, the algorithm can solve a large-scale instance by decomposing the original problem into sub-problems and sequentially solving the sub-problems. Especially, we focus on the time structure of the simulation. Because the actions of agents at the later time steps do not affect their decisions at the earlier time step, we can consider the policy from the earlier time steps to the later time steps sequentially. We believe that our algorithm is a good example of decomposing a large simulation optimization problem by using the structure of the agent model.

1.1 Related Work

One of the most popular simulation optimization approaches is to enumerate a small number of alternatives and evaluate all policies by using simulation. Then, we select the best policy based on the results of the simulation (see [6–8]). Because we have to select a small number of policies in advance, we cannot analyze all possibilities with this approach. In order to deal with this issue, black-box optimization approaches have been recently studied (see, e.g., [4]). Black-box optimization approaches can be used without knowledge and analysis of the simulation model. However, in agent-based simulation (ABS), because agents react to policies and change their behavior or interact with each other, the direct use of black-box optimization cannot provide a solution or may require extensive calculation time to obtain a solution. Another popular approach is the analytical approach. Masuda and Tsuji [9] studied the effect of priority tickets (i.e., fast passes) on a congestion equilibrium. By using models based on game theory, they obtained optimal control methods in the situation where people behave strategically. This kind of method can treat well-formalized policies such as priority tickets and pricing of attractions. However, this kind of approach is not flexible because it needs strict assumptions. To the best of our knowledge, no one has studied an algorithm for finding an optimal information distribution policy in a pedestrian (agent) situation using an analytical approach.

In the field of simulation excluding ABS, many researchers have developed algorithms by focusing on various properties/problems of simulations (see [10]). However, excepting those using black box optimization directly in ABS, few optimization algorithms are available. The main contribution of this paper is that it develops a new optimization algorithm by focusing on the properties/problems of ABS, especially the time structures of the agents' decisions and the collective property of user equilibrium.

2 Simulation Model and Problem Formulation

In this section, we first explain the simulation model and the parameters used in this model. Then, we formulate the problem.

For each positive integer x, we define $[[x]] := \{1, 2, \ldots, x\}$, i.e., $[[x]]$ denotes the set $\{1, 2, \ldots, x\}$. We denote by $[0, 1]$ the set of real numbers from 0 to 1. Furthermore, we denote by $\mathbb{N}$ the set of positive integers.

2.1 Simulation Model

In this study, we consider the following simulation model of a theme park. Each attraction has a capacity, which is the maximum number of visitors who can use this attraction simultaneously. Notice that the visitors who cannot use an attraction have to wait in line for this attraction. Furthermore, each attraction has a fixed processing time (i.e., the time required to ride this attraction once). In this paper, *congestion* is defined as queues of visitors waiting to enjoy an attraction (e.g., rides), and *reducing congestion* means reducing the maximum length of queues for an attraction. A fixed number of visitors randomly arrives at and leaves the theme park. The arrival time and departure time of each visitor obey a probability distribution (e.g., the normal distribution). In the theme park, each visitor i repeats the following sequence of actions. First, the visitor i selects an attraction a that she/he wishes to visit next. Then, the visitor i moves to the attraction a. If no one is (or a sufficiently small number of visitors are) waiting in the queue for the attraction a, the visitor i can immediately ride the attraction a. Otherwise, the visitor i waits at the end of the line. After riding the attraction a, the visitor i selects the next attraction again. This sequence of actions is repeated until the visitor leaves the theme park.

2.2 Parameters

The parameters in the model are described as follows.

- $T \in \mathbb{N}$, which represents the number of time slots.
- $\ell \in \mathbb{N}$, which represents the number of time steps in one time slot.
- $m \in \mathbb{N}$, which represents the number of agents.
- A finite set $A = \{a_1, a_2, \ldots, a_n\}$ of n attractions.
- A finite set P of locations where the agents can exist. We assume that $A \subseteq P$.
- A travel cost function $d\colon P^2 \to [0, 1]$. For each pair $(p_1, p_2) \in P^2$, $d(p_1, p_2)$ represents the travel cost between p_1 and p_2. Here, $d(p_1, p_2)$ is a normalized travel time.
- $q \in \{0, 0.1, \ldots, 1\}$, which represents the proportion of visitors who receive the congestion information.
- Q, which represents the set of visitors who receive the congestion information. We assume that $|Q| = qm$ and Q is chosen uniformly at random.

For example, if we consider the simulation from 8:00 a.m. to 6:00 p.m. and each slot is one hour, then $T = 10$. Furthermore, if one slot is one hour and one time step is one minute, then $\ell = 60$. For each integer $t \in [[T]]$, we define

$$I_t := \{(t-1)\ell + 1, (t-1)\ell + 2, \ldots, (t-1)\ell + \ell\}.$$

That is, I_t represents the set of time steps in the tth slot.

Furthermore, for each agent i, we are given the following parameters.

– The arrival time at_i and the departure time dt_i of i. We assume that $\mathsf{at}_i, \mathsf{dt}_i \in [[T\ell]]$ and $\mathsf{at}_i \leq \mathsf{dt}_i$. We assume that the set of arrival times $\{\mathsf{at}_i \mid i \in A\}$ and the set of departure times $\{\mathsf{dt}_i \mid i \in A\}$ are randomly decided according to a fixed probability distribution.
– The preference function $\alpha_i \colon A \to [0,1]$ of i. For each attraction $a \in A$, $\alpha_i(a)$ represents the preference of the visitor i for the attraction a.

The current status of an agent $i \in N$ is described using a pair (p, c_i) of the current position $p \in A$ of i and the current congestion information $c_i \in [0,1]^A$ of i such that $\max_{a \in A} c_i(a) = 1$. For each attraction $a \in A$, $c_i(a)$ represents the current congestion information about a that i has. If a visitor does not receive the congestion forecast, then $c_i(a)$ is same value for every attraction $a \in A$ (e.g., $c_i(a) = 1$). This represents the visitor expects all of congestion levels are same due to no information.

In theme parks, visitors select the next attraction to visit based on the distance from the current position to that attraction. Furthermore, if a visitor has some information about congestion, then she/he selects an attraction with short waiting time based on the information. That is, she/he tries to avoid congested attractions. In this study, we use the agent model proposed in [6], as it can consider information gathering and decision making. In [6], the multinomial logit model is used to model people's behavior and reproduce the congestion in theme parks. More specifically, the utility $U_i(a,p)$ of a visitor i at a position p for each attraction a is defined by

$$U_i(a,p) := \alpha_i(a) + \beta_1 d(a,p) + \beta_2 c_i(a), \tag{1}$$

where β_1, β_2 are the parameters for balancing the travel cost and congestion cost. In this model, once a visitor rides on an attraction, she/he will remove it from a list her/his want to ride. Notice that the congestion cost may change when the visitor i obtains a new congestion forecast. As mentioned above, if a visitor i does not receive a congestion forecast, then $c_i(a) = 1$ for every attraction $a \in A$. In the simulation, visitors sequentially decide the attraction to visit by maximizing their utility function (1). The visitors always decide sequentially and do not plan route in advance.

2.3 Problem Formulation

Here, we define our problem. The simulation has two elements of randomness. The first is the set of visitors who receive the distributed congestion information (i.e., the set Q), and the second is the arrival time and departure time of each visitor. We denote by ξ the realization of this randomness of the system. A *congestion forecast* is a function $F \colon [[T]] \times A \to [0,1]$ such that for every integer $t \in [[T]]$, $\max_{a \in A} F(t,a) = 1$.

Assume that we are given a congestion forecast F. For each integer $\theta \in [[T\ell]]$ and each attraction $a \in A$, we define $w_F(\theta, a; \xi)$ as the length of the waiting queue for the attraction a at the time step θ when we use F as the input of the

simulation with a realization ξ of the randomness of the system. Furthermore, for each integer $t \in [[T]]$ and each attraction $a \in A$, we define $\overline{w}_F(t, a; \xi)$ by

$$\overline{w}_F(t, a; \xi) := \frac{1}{\ell} \sum_{\theta \in I_t} w_F(\theta, a; \xi).$$

That is, $\overline{w}_F(t, a; \xi)$ is the average length of the waiting queue for the attraction a during the tth slot. For each integer $t \in [[T]]$, we define

$$m_F(t; \xi) := \max_{a \in A} \overline{w}_F(t, a; \xi).$$

Thus, $m_F(t; \xi)$ is the maximum average length of the waiting queue among all the attractions during the tth slot. $\mathsf{max\_len}(F; \xi)$ is defined by

$$\mathsf{max\_len}(F; \xi) := \max\{m_F(t; \xi) \mid t \in [[T]]\}.$$

For each pair $(t, a) \in [[T]] \times A$, we define the real number $\mathbf{Sim}_F(t, a; \xi)$ by

$$\mathbf{Sim}_F(t, a; \xi) := \frac{\overline{w}_F(t, a; \xi)}{m_F(t; \xi)}.$$

Notice that $\mathbf{Sim}_F(t, a; \xi)$ is the normalized congestion by the simulation for the congestion forecast F. For each integer $t \in [[T]]$, we define

$$\mathbf{Err}_t(F; \xi) := \frac{1}{n} \sum_{a \in A} \Big| \mathbf{Sim}_F(t, a; \xi) - F(t, a) \Big|.$$

In other words, $\mathbf{Err}_t(F; \xi)$ represents the average of the absolute errors between the forecast and the result of the simulation at the tth slot. Lastly, we define

$$\mathbf{Err}(F; \xi) := \sum_{t \in [[T]]} \mathbf{Err}_t(F; \xi).$$

That is, $\mathbf{Err}(F; \xi)$ is the sum of the absolute errors between the forecast and the result of the simulation.

We are now ready to formulate the problem of finding a congestion forecast F. In the problem, the congestion forecast F has to approximately satisfy user equilibrium conditions. The problem can be formulated as follows. In (2), ε is a small real number given in advance.

$$\begin{array}{ll} \text{Minimize} & \mathsf{max\_len}(F; \xi) \\ \text{subject to} & \mathbf{Err}(F; \xi) \leq \varepsilon \\ & F \colon [[T]] \times A \to [0, 1]. \end{array} \tag{2}$$

Notice that the constraint of the problem of (2) means that F approximately satisfies the user equilibrium conditions.

3 Algorithm

In this section, we explain our algorithm for finding an appropriate congestion forecast. The algorithm comprises two parts. In the first part, we try to find a congestion forecast satisfying the user equilibrium conditions (see Algorithm 1). In the second part, we optimize the proportion of visitors who receive the congestion information.

3.1 Finding Congestion Forecast

In Algorithm 1, we are given a positive real number ε' and a positive integer Δ, where ε' is the stopping accuracy of the algorithm and Δ is the upper bound of the number of iterations for determining the forecast of each slot. Recall that ξ is a realization of the randomness of the system.

Algorithm 1: Algorithm for finding congestion forecast

Define an initial forecast $F_0 \colon [[T]] \times A \to [0,1]$ by $F_0(t,a) := 1$ for each pair $(t,a) \in [[T]] \times A$.
for $t = 1, 2, \ldots, T$ **do**
  Define $G_{t,0} := F_{t-1}$ and $\gamma_{t,0} := 1$. Set $\delta := 0$.
  while $\mathbf{Err}_t(G_{t,\delta}; \xi) > \varepsilon'$, $\gamma_{t,\delta} \neq 0$, *and* $\delta \leq \Delta$ **do**
    Define the function $h_{t,\delta} \colon [[T]] \times A \to [0,1]$ by

$$h_{t,\delta}(t',a) := \begin{cases} \mathbf{Sim}_{G_{t,\delta}}(t,a;\xi) & \text{if } t' = t \\ G_{t,\delta}(t',a) & \text{if } t' \neq t. \end{cases}$$

    For each real number $x \in \{0, 0.1, \ldots, 1\}$ and each pair $(t',a) \in [[T]] \times A$, we define

$$H^x_{t,\delta}(t',a) := (1-x) \cdot G_{t,\delta}(t',a) + x \cdot h_{t,\delta}(t',a).$$

    For each real number $x \in \{0, 0.1, \ldots, 1\}$, we define the function $E^x_{t,\delta} \colon [[T]] \times A \to [0,1]$ by

$$E^x_{t,\delta}(t',a) := \frac{H^x_{t,\delta}(t',a)}{\max_{a' \in A} H^x_{t,\delta}(t',a')}.$$

    Define $\gamma_{i,\delta+1}$ as a minimizer of

$$\min_{x \in \{0,0.1,\ldots,1\}} \mathbf{Err}_t(E^x_{t,\delta}; \xi).$$

    Define $G_{t,\delta+1} := E^{\gamma_{t,\delta+1}}_{t,\delta}$. Set $\delta := \delta + 1$.
  end
  Define $F_t := G_{t,\delta}$.
end
Output F_T, and halt.

In Algorithm 1, we first set the initial forecast as F_0. Then, we sequentially determine the forecast of each slot from the first slot to the Tth slot. In the tth iteration of the main loop (i.e., Steps 2–12), we determine the forecast of the tth slot. If the error of the current forecast $G_{t,\delta}$ is sufficiently small (i.e., $\mathbf{Err}_t(G_{t,\delta};\xi) \leq \varepsilon'$), then we stop this iteration. Furthermore, if the current forecast $G_{t,\delta}$ is not improved or the number of iterations for this slot is sufficient, then we stop this iteration. Otherwise, we try to improve the current forecast $G_{t,\delta}$. First, we construct the function $h_{t,\delta}$ from $G_{t,\delta}$ by replacing the forecast of the tth slot with the result of the simulation for $G_{t,\delta}$. Then, we update $G_{t,\delta}$ by using $h_{t,\delta}$. We also optimize the degree of the update (i.e., $\gamma_{t,\delta}$). We repeat this from the first slot to the Tth slot.

3.2 Optimizing the Proportion of Visitors Receiving Congestion Information

Here, we consider the problem of optimizing the proportion of visitors who receive the congestion information. If q is fixed, then we can use Algorithm 1. Therefore, in order to optimize the proportion of visitors who receive the congestion information, we set the candidate sets of the proportion of visitors receiving the congestion information as $\{0, 0.1, \ldots, 1\}$, and evaluate each candidate. More precisely, for each real number $q \in \{0, 0.1, \ldots, 1\}$, we compute the output F^q of Algorithm 1 under the assumption that q is given as an input. Then, we find a minimizer q^* of

$$\min_{q \in \{0, 0.1, \ldots, 1\}} \mathsf{max\_len}(F^q; \xi).$$

Finally, we output F^{q^*}, and halt the processing.

4 Computational Experiments

In this section, we explain the experimental settings and compare our algorithm with black-box optimization methods. Then, we apply our algorithm to a large-scale instance, and analyze the scalability of our algorithm. Finally, we consider the convergence of our algorithm.

4.1 Experimental Settings

The simulation used in the computational experiments represents a theme park, which is modeled using a queueing network. The geography of the theme park is represented by using a directed graph (see Figure 2). The nodes represent the attractions, and edges represent the roads, which have travel times. An attraction consists of one queue lane and several service units. The service units represent, for example, rides of the roller coaster. If visitors are waiting in the queue lane for an attraction, a new visitor lines up at the end of the line. Otherwise, the new visitor can ride on a vacant service unit. Each unit has a capacity and time identified as parameters.

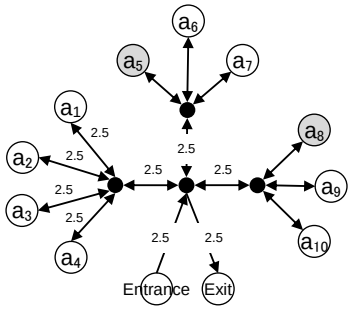

Fig. 2. An example of the geography of a large-scale instance. Gray nodes represent popular attractions.

In the computational experiments, we prepare two parameter sets called the large-scale instance and small-scale instance. First, the large-scale instance is a real scale setting, which models a single day in the entire theme park. Second, the small-scale instance is a downsized setting for the comparison experiments. In both the instances, there exist a few popular attractions, and the others are normal attractions. The goal of optimization is leveling the usage between the popular attractions and the normal attractions.

Large-scale instance In the large-scale instance, there are $50,000$ visitors and 10 attractions $A = \{a_1, a_2, \ldots, a_{10}\}$. The attractions a_5, a_8 are popular attractions. The others a_1, a_2, a_3, a_4, a_6, a_7, a_9, and a_{10} are normal attractions. The simulation is performed from 8:00 a.m. to 6:00 p.m. (i.e., $T = 10$), and each time step corresponds to a minute (i.e., $\ell = 60$). The preferences $(\alpha_i(a_1), \alpha_i(a_2), \ldots, \alpha_i(a_{10}))$ of the visitors can be grouped into two types based on the attractions. The first type prefers the popular attractions, i.e., $\alpha_i(a_5)$ and $\alpha_i(a_8)$ are 1. In the second type, the visitor does not prefer the popular attractions, i.e., $\alpha_i(a_1)$, $\ldots$, $\alpha_i(a_4)$, $\alpha_i(a_6)$, $\alpha_i(a_7)$, $\alpha_i(a_9)$, and $\alpha_i(a_{10})$ are 1. Half of the agents have the first type of preference, and the others have the second type of preference. The arrival time of an agent is stochastic, and it obeys the normal distribution with the expected value of 8:30 a.m. and standard deviation of 60 min. Similarly, the departure time of an agent obeys the normal distribution with expected value of 4:30 p.m. and standard deviation of 60 min. The geography of the theme park is described in Figure 2. There exist three zones and each zone consists of three or four attractions. The travel time of each edge is 2.5 min, and $d(a, p)$ is the normalized sum of the travel time of the shortest path from the current position p to the attraction a (e.g., if a visitor i is at Entrance and evaluates the travel cost to a_5, $d(a_5, \text{Entrance})$ is $7.5/10.0 = 0.75$, where 7.5 is the sum of the travel times of the shortest path and 10.0 is the sum of the travel times of the longest path of the network). If the visitor i does not receive the congestion information, then $c_i(a) = 1$ for each attraction a. If the visitor i receives the congestion information F, then $c_i(a) = F(t, a)$ for each attraction

a at the time step in I_t. We define $\beta_1 := -1$ and $\beta_2 := -1$. Each agent deterministically selects an attraction that maximizes her/his utility. Each attraction has 100 units, the capacity of each unit is 6, and the running time is 10 min.

Small-scale instance The small-scale instance is similar to the large-scale instance. However, the number of visitors is 500, and the number of attractions is 3 with only 1 unit. The simulation is performed from 8:00 a.m. to 12:00 p.m. (i.e., $T = 4$). The geography simply comprises one zone with one popular attraction a_1 and two normal attractions a_2, a_3. There exist 3×4 parameters for optimization. For simplicity, we assume that $d(a, p) = 0$ for every attraction a and every position p.

In the following experiments, we set the parameters of Algorithm 1 by $\varepsilon' := 0.01$ and $\Delta := 10$.

4.2 Comparison with Black-Box Optimization Methods

In this subsection, we compare our algorithm with the Nelder–Mead method (NM method) [11] and Bayesian optimization (BO) [12]. The NM method and BO, which are called derivative-free optimization methods or black-box optimization methods, are frequently used for simulation optimization because they do not need derivatives of the system. The experiments were conducted for the small-scale instance, and we attempted to find an optimal congestion forecast for the case where $q = 0.5$. Because the NM method and BO cannot optimize a discrete variable, we only treat continuous variables (a congestion forecast). We compared the number of iterations in each case. The NM method was implemented in Scikit-learn [13] and BO was implemented in the Bayesian optimization module [14]. It should be remarked that both hyper-parameters were not tuned. We compared our algorithm with the black-box optimization method in terms of the maximum queue length $\mathsf{max\_len}(F; \xi)$ and forecasting error $\mathbf{Err}(F; \xi)$. Because the NM method and BO can treat a single objective function, we define a new objective function

$$g(F; \xi) := z \cdot \frac{\mathbf{Err}(F; \xi)}{N_1} + (1 - z) \cdot \frac{\mathsf{max\_len}(F; \xi)}{N_2}.$$

That is, $g(F; \xi)$ is the weighted sum of two objectives $\mathsf{max\_len}(F; \xi)$ and $\mathbf{Err}(F; \xi)$. We use the weight $z = 0.5$ and the normalization factors $N_1 = 1$, $N_2 = 100$. Figure 3 shows the transitions of $g(F; \xi)$, the maximum queue length $\mathsf{max\_len}(F; \xi)$, and the forecasting error $\mathbf{Err}(F; \xi)$. The NM method and BO need greater number of iterations to reach the algorithm result. Especially, because the forecasting error is not at all reduced, we expect that they require a huge number of iterations. It may be possible to reduce the number of iterations by using appropriate hyper parameters. However, to identify the appropriate hyper parameters, we need domain knowledge about simulation models or tuning algorithms requiring numerous iterations.

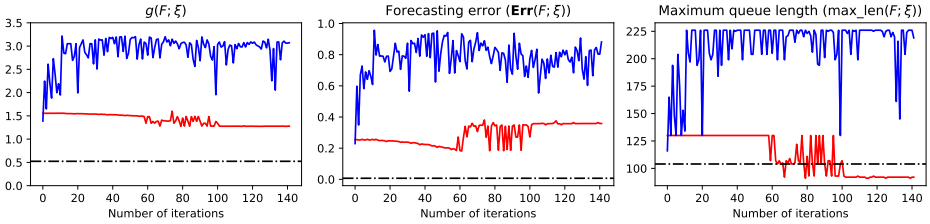

Fig. 3. Comparison of the algorithms. The red and blue lines denote the NM method and BO, respectively, and the black lines denote the result after 142 iterations of our algorithm (the solution converged).

4.3 Optimization in Large-Scale Simulation

In this subsection, we evaluate our algorithm for the large-scale instance. From the change in queue lengths (Figure 4), we can see that the congestion at the popular attractions a_5, a_8 can be reduced by the forecast. Moreover, the obtained congestion forecast (Table 1) matches the forecast obtained from the algorithm (the right graph in Figure 4). This is the same for any q when the calculation converges (i.e., $\delta \leq \Delta$ for all t). The number of total iterations is 3638, and the calculation time is 3 days, 9 h, and 5 min. Actually, we parallelized the code, and thus the calculation time was shortened from 81 h to 10.5 h. The main reason of the increase in the calculation time is the increase in the simulation time. In the small-scale instance, a single execution of the simulation requires merely 1 to 2 s. On the other hand, the corresponding duration for the large-scale instance is 80 to 90 s. The other reason is the increase in the number of iterations. Strangely, the number of iterations increases by just 3 times (from 1190 to 3086) even though the number of optimization variables is increased by approximately 10 times (from 12 to 100).

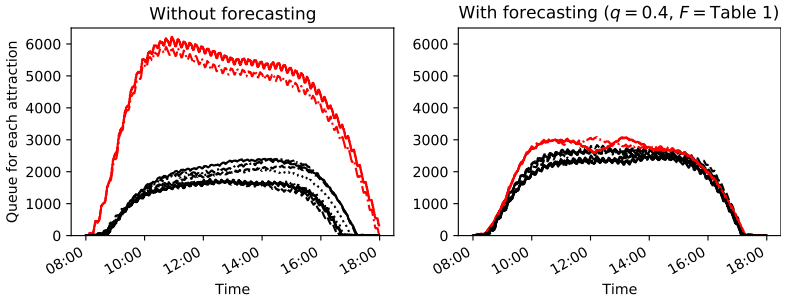

Fig. 4. Lengths of the waiting queues. The red lines are the popular attractions a_5, a_8. In the right figure, we use the information distribution obtained by our algorithm ($q = 0.4$, $F =$ Table 1).

Table 1. Congestion forecast obtained by our algorithm.

T	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.720418	0.675817	0.696171	0.692837	1.	0.670981	0.762788	0.995877	0.685597	0.696621
2	0.808647	0.811263	0.812272	0.79651	1.	0.718593	0.736128	0.997293	0.713978	0.717667
3	0.841241	0.803143	0.823742	0.835404	0.991737	0.737548	0.745148	1.	0.73498	0.741999
4	0.899661	0.863094	0.894522	0.889992	1.	0.784652	0.770107	0.984395	0.799494	0.785912
5	0.902241	0.924524	0.893276	0.862496	0.933075	0.807642	0.818644	1.	0.809866	0.790043
6	0.870545	0.888406	0.906036	0.844671	1.	0.784408	0.799651	0.94668	0.805791	0.824702
7	0.926221	0.955718	0.937145	0.917229	1.	0.893041	0.89844	0.993736	0.899123	0.884014
8	0.935964	1.	0.961879	0.920934	0.986462	0.867541	0.866823	0.97186	0.870089	0.911179
9	0.922942	1.	0.911364	0.87543	0.91263	0.811624	0.780999	0.878249	0.827324	0.791795
10	0.829386	1.	0.763565	0.722922	0.9501	0.622605	0.497716	0.961467	0.529893	0.450597

Our algorithm sequentially solves a sub-problem for each time slot. Thus, the number of iterations linearly increases with regard to the number of time slots. In this case, the number of time slots is increased by 2.5 times (from 4 to 10), and thus the number of iterations is increased by approximately 2.5 times. This implies that our algorithm is scalable with regard to the number of time slots (i.e., T). However, the scalability with regard to the number of forecasting variables (i.e., the number of attractions in A) is unclear. We analyze this point in the next subsection.

4.4 Scalability Analysis

In this subsection, we investigate the scalability of our algorithm. First, we apply our algorithm to various combinations of the number of time slots (i.e., T) and the number of forecasting variables (i.e., the number of attractions $|A|$). Figure 5 shows that the number of iterations increases approximately linearly with regard to the number of time slots and the number of forecasting variables. Second, we apply our algorithm to a variety of congestion situations. It is possible to create various degrees of congestion by changing the number of popular attractions and the proportion of visitors preferring the popular attractions. The result shows that our algorithm can reduce congestion in any congestion situation (see Figure 6). Moreover, although the number of iterations changes depending on the degree of congestion, the number of iterations in different situations is at most two times.

4.5 Convergence Analysis

In this subsection, we numerically investigate the convergence of our algorithm. Figure 7 is a congestion forecast calculated by our algorithm on error distribution. The graphs show that the algorithm can find a set of values mostly corresponding to the minimum error in three error distribution structures ($q =$

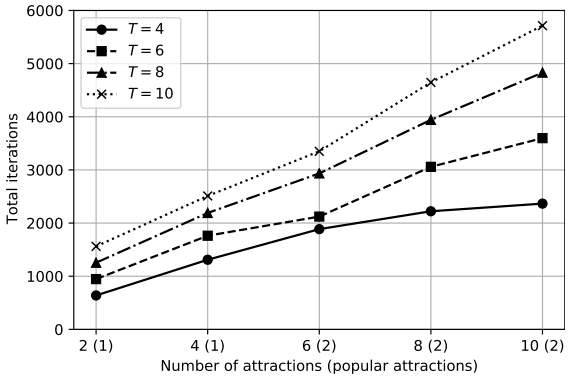

Fig. 5. Relation between the total number of iterations of our algorithm, number of attractions (horizontal axis), and the number of time slots (series). The experiments were conducted for a small-scale instance.

$0.2, 0.5, 0.7$). The graphs imply that there are situations with no forecast satisfying the user equilibrium conditions (i.e., $\min \mathbf{Err}_t(F;\xi) > \varepsilon'$). Even in such situations, the algorithm can find values that mostly minimize the error. Figure 8 shows the congestion forecasts calculated by the algorithm from various initial forecasts (F_0). The graphs show that the algorithm can find a forecast that mostly realizes the minimum error regardless of the initial forecasts. Table 2 shows the average number of iterations and the reason for stopping in 100 different initial forecast trials. In every trial, the algorithm stops before reaching the upper bound $\Delta = 10$. If the minimum error can be smaller than ε', the algorithm is stopped because $\min \mathbf{Err}_t(F;\xi) \leq \varepsilon'$ and a forecast satisfying the user equilibrium conditions can be found. If there is no minimum error smaller than ε', the algorithm is stopped because $\gamma = 0$ and naturally, no forecast satisfying the user equilibrium condition can be found. In summary, our algorithm can converge within several iterations of various error distribution structures. Even when there is no forecast satisfying the user equilibrium conditions, the algorithm can find values that mostly minimize the error.

5 Conclusion

In this paper, we formalized and evaluated the pedestrian flow control method by using congestion forecasts satisfying the user equilibrium conditions proposed in [2]. More precisely, we proposed a simulation optimization algorithm for finding appropriate congestion forecasts and information distribution policy. Then, we evaluated the usefulness of our algorithm through computational experiments. As shown in Section 4.2, even though the number of variables is small, it is not easy for existing black-box algorithms to solve the problem with a small number of iterations. On the other hand, our algorithm can approximately solve

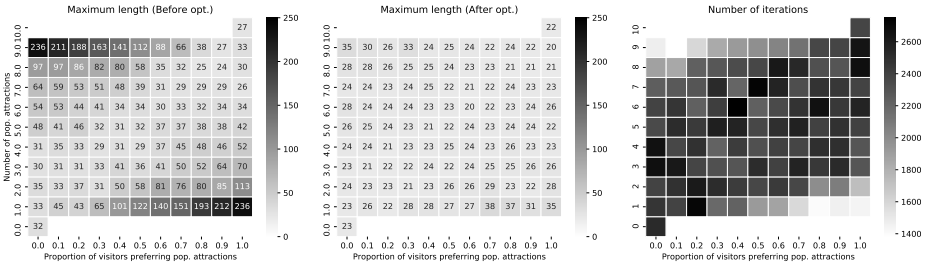

Fig. 6. Maximum length of the waiting queue for various combinations of the number of popular attractions and proportion of visitors preferring the popular attractions (left graph). The center graph is the result of distributing the optimal congestion forecast after optimization. The right graph shows the number of iterations for the small-scale instance, $T = 4$ and $|A| = 10$. Heavy congestion is observed on the bottom right of each graph, as many visitors gather at some popular attractions. Heavy congestion is also observed on the upper left of each graph, as many visitors gather at some "normal" attractions. For example, in the situation where 10% of the visitors prefer 9 popular attractions (upper left), the remaining visitors (90%) prefer the remaining attraction (1 normal attraction). The closer the situation is to the diagonal from the upper right to the lower left, the greater is the decrease in congestion.

q	Time slot	Average number of iterations	Reasons for stopping			Average error
			Iterations $> \Delta$	Error $\leq \varepsilon'$	$\gamma = 0$	
0.2	T1	3.21	0%	0%	100%	0.0446
	T2	2.07	0%	100%	0%	0.0000
	T3	2.01	0%	100%	0%	0.0000
	T4	1.73	0%	100%	0%	0.0000
0.5	T1	3.83	0%	0%	100%	0.0357
	T2	1.73	0%	95%	5%	0.0008
	T3	3.77	0%	5%	95%	0.0101
	T4	1.84	0%	100%	0%	0.0000
0.7	T1	3.36	0%	0%	100%	0.0882
	T2	1.87	0%	94%	6%	0.0011
	T3	3.63	0%	2%	98%	0.0110
	T4	1.79	0%	100%	0%	0.0000

Table 2. Average number of iterations, the reason for stopping, and the average error of the 100 trials. The average number of iterations is shown for 100 trials for various initial forecasts. The 2nd to 4th column shows the percentage of the three reasons for stopping: the number of iterations reaches the upper bound (Iterations $> \Delta$), the error is below the stopping accuracy (Error $\leq \varepsilon'$), or the degree of update is 0 ($\gamma = 0$). The 5th column shows the average error of 100 trials.

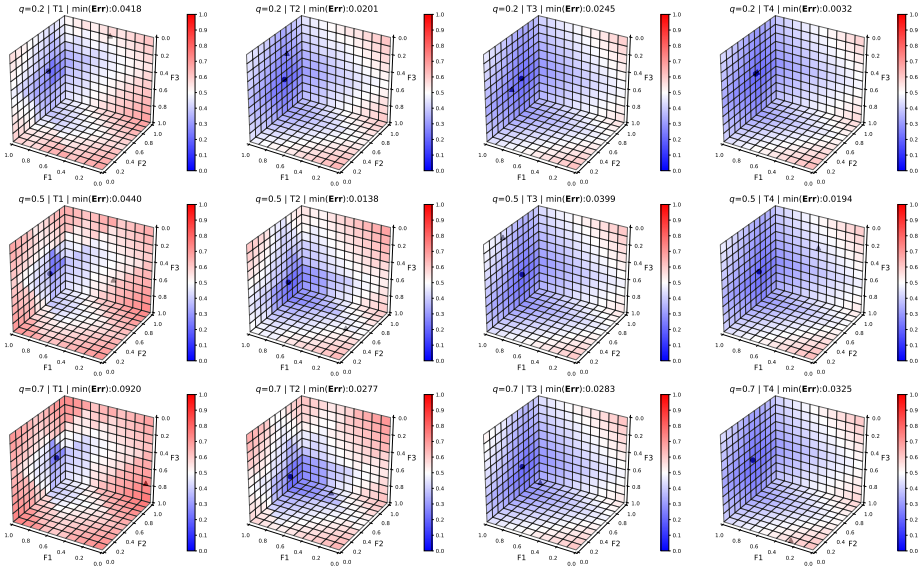

Fig. 7. Error distribution for the various proportion of visitors receiving congestion information ($q = 0.2, 0.5, 0.7$). Each grid represents a situation where a certain congestion forecast (corresponding to the $F1$ axis, $F2$ axis, $F3$ axis) is provided. If the congestion forecast is normalized, then one of the values ($F1$, $F2$, $F3$) is always 1. The color of each grid represents a forecasting error ($\mathbf{Err}_t(F;\xi)$) in each situation. The errors are calculated using a grid search. The circle represents the congestion forecast calculated by our algorithm. The triangle is the initial forecast (F_0) of our algorithm. F_0 is chosen randomly.

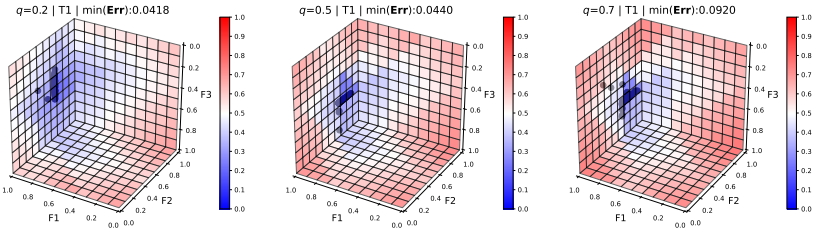

Fig. 8. Congestion forecast calculated by our algorithm from various initial forecasts (F_0). We conducted 100 trials with different initial forecasts. The circle represents the congestion forecast calculated for each trial. The error distribution of $T1$ is the same for all trials, but the error distributions of $T2, T3, and T4$ are different among for each trial. Therefore, only $T1$ is shown.

a large-scale simulation having many optimization variables in a reasonable time (see Section 4.3). Moreover, the efficiency of our algorithm may not be limited to certain conditions (see Section 4.5), and the convergence of the algorithm is satisfactory (see Section 4.4).

Acknowledgments This work was supported by JST, PRESTO Grant Number JPMJPR1753, Japan, and Fujitsu Laboratories, Ltd. The authors would like to thank Hiroaki Iwashita, Takuya Ohawa, and Kotaro Ohori for the useful discussions.

References

1. Imai, T., Nishinari, K.: Optimal information provision for maximizing flow in a forked lattice. Physical Review E **91**(6) (2015) 062818–1–062818–7
2. Yamada, H., Kamiyama, N.: An information distribution method for avoiding hunting phenomenon in theme parks (extended abstract). In: Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems. (2020) 2050–2052
3. Jian, N., Henderson, S.G.: An introduction to simulation optimization. In: Proceedings of the 2015 Winter Simulation Conference. (2015) 1780–1794
4. Audet, C., Hare, W.: Derivative-free and blackbox optimization. Springer (2017)
5. LeBlanc, L., Morlok, E., Pierskalla, W.: An efficient approach to solving the road network equilibrium traffic assignment problem. Transportation Research **9**(5) (1975) 309–318
6. Ohori, K., Iida, M., Takahashi, S.: Virtual grounding for facsimile model construction where real data is not available. SICE Journal of Control, Measurement, and System Integration **6**(2) (2013) 108–116
7. Bufala, N.D., Kant, J.: An evolutionary approach to find optimal policies with an agent-based simulation. In: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems. (2019) 610–618
8. Shigenaka, S., Takami, S., Ozaki, Y., Onishi, M., Yamashita, T., Noda, I.: Evaluation of optimization for pedestrian route guidance in real-world crowded scene (extended abstract). In: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems. (2019) 2192–2194
9. Masuda, Y., Tsuji, A.: Congestion control for a system with parallel stations and homogeneous customers using priority passes. Networks and Spatial Economics **19**(1) (2019) 293–318
10. Amaran, S., Sahinidis, N., Sharda, B., Bury, S.: Simulation optimization: a review of algorithms and applications. Annals of Operations Research **240** (2016) 351–380
11. Nelder, J.A., Mead, R.A.: A simplex method for function minimization. The Computer Journal **7** (1965) 308–313
12. Mockus, J.: On Bayesian methods for seeking the extremum. In: Proceedings of Optimization Techniques IFIP Technical Conference Novosibirsk. Volume 27 of Lecture Notes in Computer Science. (1974) 400–404
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830
14. Fernando, N.: Bayesianoptimization. https://github.com/fmfn/BayesianOptimization (2019)