

# FASTS: A Satisfaction-Boosting Bus Scheduling Assistant

Songsong Mo<sup>1</sup>, Zhifeng Bao<sup>2</sup>, Baihua Zheng<sup>3</sup>, and Zhiyong Peng<sup>1\*</sup>

<sup>1</sup>Wuhan University, <sup>2</sup>RMIT University, <sup>3</sup>Singapore Management University

<sup>1</sup>{songsong945, peng}@whu.edu.cn, <sup>2</sup>zhifeng.bao@rmit.edu.au, <sup>3</sup>bhzheng@smu.edu.sg

## ABSTRACT

In this paper, we demonstrate a satisfaction-boosting bus scheduling assistant called FASTS, which assists users to find an optimal bus schedule. FASTS performs bus scheduling based on the constraints specified by the user in either a coarse-grained or a fine-grained manner, supports different explorations with a varying number of constraints, and provides analysis to quantify the performance of bus schedules and presents the results in a visually pleasing way. We demonstrate FASTS using real-world bus routes (396 routes) and one-week bus touch-on/touch-off records (28 million trip records) in Singapore.

### PVLDB Reference Format:

Songsong Mo, Zhifeng Bao, Baihua Zheng, Zhiyong Peng. FASTS: A Satisfaction-Boosting Bus Scheduling Assistant. *PVLDB*, 13(12): 2873-2876, 2020.

DOI: <https://doi.org/10.14778/3415478.3415497>

## 1. INTRODUCTION

Public transport services such as buses are essential to our daily life. However, the cost of buses and the operating fee are not cheap. Taking New York City as an example, the cost of each bus is around \$550,000 and the operating cost of transit agencies reaches \$215 per hour<sup>1</sup>. Assume we are able to re-organize the bus frequencies based on real travel demands and thereby reduce 10% bus departures, we can save \$20 operating costs per hour and \$55,000 per vehicle.

To this end, the primary goal of this work is to find the nearly optimal bus departure timetable to meet the actual travel demands, under the assumption that the historic travel demands (real world bus touch-on/touch-off records) reflect the actual travel demands<sup>2</sup>. To the best of our knowledge, there is no publicly accessible bus schedule planning

\*Zhiyong Peng is the corresponding author.

<sup>1</sup><https://www.liveabout.com/bus-cost-to-purchase-and-operate-2798845>

<sup>2</sup>How to predict the actual travel demands in the near future is a different problem, which is not the focus of this work.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org). Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

*Proceedings of the VLDB Endowment*, Vol. 13, No. 12

ISSN 2150-8097.

DOI: <https://doi.org/10.14778/3415478.3415497>

assistant at present. Most existing studies on bus scheduling problem share a common objective, which is to minimize the average travel cost (in terms of waiting time) of passengers [4, 5]. As compared to simply minimizing the average waiting time, we argue that it makes more sense in reality if a bus can serve a passenger within a waiting time threshold that she could accept. Many studies have confirmed that the user satisfaction drops faster as the waiting time increases [1, 3]. Motivated by this observation, we aim to schedule the buses such that they could serve more passengers within a given waiting time threshold that users could accept, rather than simply minimizing the average waiting time.

In this paper, we demonstrate a satisfaction-boosting bus scheduling assistant (FASTS). FASTS can automatically schedule bus departures within the constraints specified by the user via its *Coarse-grained Scheduling* and support different explorations with a varying number of constraints. The scheduling algorithms have been developed by our recent work [6]. It also provides users with fine-tuning capabilities via *Fine-grained Scheduling with User Interaction* so that users could adjust the bus departures returned by *Coarse-grained Scheduling*, e.g., changing the departure time of certain buses and adding (or removing) some bus departures. Users also can specify regions, lines and stops for the task of bus scheduling in *Fine-grained Scheduling with User Interaction*. FASTS also provides *Impact Analysis*, which can perform various analytics and provide a pleasing and intuitive visualization to help users evaluate and understand the performance of current bus scheduling. Additionally, *Impact Analysis* supports users to compare several saved bus schedules. Users then could further adjust the constraints via either *Coarse-grained Scheduling* or *Fine-grained Scheduling with User Interaction*. The corresponding changes brought by different constraints will also be highlighted and given as feedbacks to users, leading to in-depth user understanding and interaction. A prototype of the demo is available online<sup>3</sup>.

## 2. SYSTEM ARCHITECTURE

Figure 1 shows the system architecture of FASTS.

**Front End.** The UI of FASTS presents key information—including a map view, current parameter values, bus timetables, and the analytics results output by the back end data analysis module—to users in a visually pleasing and effective way, as shown in Figure 2. In addition, it provides an interface where the inputs from users can be captured and passed

<sup>3</sup><http://civilcomputing.co/busscheduling>

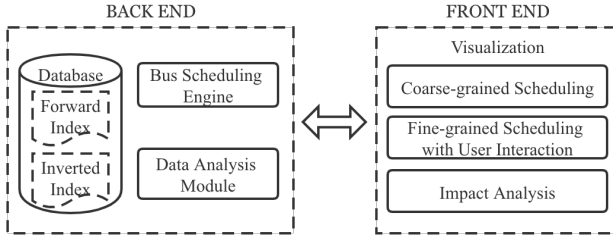


Figure 1: System Architecture

to the back end engines to reschedule the buses based on the new inputs from the users. For example, by interacting with the front end, users can adjust parameters, submit inquiries, change bus schedules, etc.

**Back End.** The back end contains a bus scheduling engine, a data analysis module, a forward index and an inverted index. The bus scheduling engine aims to find the optimal bus schedule based on the constraints, if any, specified by the user. The data analysis module analyzes the passenger satisfaction, number of bus departures, travel demand, and passenger waiting time. The two index structures maintained in the back end facilitate the bus scheduling. The forward index uses buses as keys, where the value corresponding to a key is a set of tuples in the form of  $(p_i, wt_i)$ , where  $p_i$  refers to a passenger who wants to take this bus and  $wt_i$  refers to the waiting time of the passenger if she/he takes this bus. The inverted index uses passengers as keys, where the value is a set of tuples in the form of  $(b_j, wt_j)$ , where  $b_j$  refers to one bus that could serve the passenger, and  $wt_j$  refers to the waiting time required for the bus  $b_j$ . Based on these two indexes, we can retrieve which bus can serve a passenger in the given waiting time threshold and vice versa.

### 3. BACK END TECHNIQUES

In this section, we first show the problem formulation, and then introduce how to find the optimal bus schedule effectively. Please refer to our work [6] for details.

#### 3.1 Problem Formulation

In a bus route database  $\mathcal{R}$ , a route  $r$  is a sequence of bus stations  $(s_1, s_2, \dots, s_i, \dots, s_m)$ , where  $s_i$  is a bus station represented by (latitude, longitude). In a passenger database  $\mathcal{P}$ , a passenger  $p \in \mathcal{P}$  is in form of a tuple  $\{s_b, s_e, t\}$ , where  $s_b$  denotes the boarding station,  $s_e$  denotes the alighting station, and  $t$  denotes the time when  $p$  reaches  $s_b$ . A bus  $b_{ij}$  is in form of a tuple  $\{r_i, dt_j\}$ , where  $r_i$  and  $dt_j$  denote the bus service route and the departure time from  $r_i.s_1$  respectively.

We define that a bus  $b_{ij}$  can serve a passenger  $p$ , if  $r_i$  contains  $p.s_b$  and  $p.s_e$  in order, and  $0 \leq dt_j + T(r_i.s_1, p.s_b) - t \leq \theta$ , where  $T(r_i.s_1, p.s_b)$  denotes the travel time required by bus  $b_{ij}$  from  $r_i.s_1$  to  $p.s_b$  via the bus route  $r_i$ , and  $\theta$  is a given waiting time threshold. There are multiple ways available to approximate  $T(r_i.s_1, p.s_b)$ . In this paper, we utilize the historical average travel time from  $r_i.s_1$  to  $p.s_b$  via the route  $r_i$  to compute  $T(r_i.s_1, p.s_b)$ . We formally introduce  $\mathcal{S}(b_{ij}, p_k)$  to denote the service of  $b_{ij}$  to  $p_k$ , as presented in Equation (1).

$$\mathcal{S}(b_{ij}, p_k) = \begin{cases} 1 & \text{if } b_{ij} \text{ can serve } p_k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Next, we introduce the concept of bus service frequency and formulate our problem in Definition 3.1. A bus service frequency ( $f_i$ ) for  $r_i$  refers to a set of buses  $(b_{i1}, b_{i2}, \dots, b_{in_i})$  that serves the route  $r_i$ , where  $n_i$  ( $n_i \geq 1$ ) denotes the total number of bus departures corresponding to the route  $r_i$  within a day. Let the bus service frequency  $\mathcal{F}$  for  $\mathcal{R}$  be a set, with each element  $f_i \in \mathcal{F}$  corresponding to a bus route  $r_i \in \mathcal{R}$ , i.e.,  $\mathcal{F} = \{\cup_{r_i \in \mathcal{R}} f_i\}$ . Then, the service of  $\mathcal{F}$  to a passenger  $p_k$  can be computed by  $\mathcal{S}(\mathcal{F}, p_k) = 1 - \prod_{b_{ij} \in \mathcal{F}} (1 - \mathcal{S}(b_{ij}, p_k))$ . Note  $\mathcal{S}(\mathcal{F}, p_k) = 1$  as long as any  $b_{ij} \in \mathcal{F}$  can serve  $p_k$ ; otherwise,  $\mathcal{S}(\mathcal{F}, p_k) = 0$ .

**DEFINITION 3.1.** Given a bus route database  $\mathcal{R}$ , a passenger database  $\mathcal{P}$ , a waiting time threshold  $\theta$ , and a vector  $\mathcal{N}(n_1, n_2, \dots, n_i, \dots, n_{|\mathcal{R}|})$  where  $n_i$  ( $\geq 1$ ) denotes the total number of bus departures of bus route  $r_i \in \mathcal{R}$ , we output a bus service frequency  $\mathcal{F}$  which can maximize  $\mathcal{G}(\mathcal{F}) = \sum_{p_k \in \mathcal{P}} \mathcal{S}(\mathcal{F}, p_k)$ , where  $\mathcal{G}(\mathcal{F})$  denotes the total number of passengers served by  $\mathcal{F}$ .

### 3.2 Methodology

Based on the given parameters, FASTS will find the near-optimal bus schedule that can maximize the satisfaction of the passengers.

The basic greedy algorithm is applicable because the objective function of FASTS is submodular. To accelerate the marginal gain computation, which is the main bottleneck of the basic greedy algorithm, we propose two mapping indexes, *forward list* ( $\mathcal{I}_f$ ) and *inverted list* ( $\mathcal{I}_i$ ).

1)  $\mathcal{I}_f$  is for buses  $b_i \in \mathcal{B}$ . It maintains a list of passengers  $L_P$  that could be served by bus  $b_i$ . In addition, to avoid counting the same passenger multiple times when we calculate the marginal gain, we maintain another parameter  $N_{ToBeServed}$  to capture the number of passengers in  $L_P$  that are still waiting for services.

2)  $\mathcal{I}_i$  is for passengers  $p \in \mathcal{P}$ . It maintains a list of buses that could serve the passenger  $p$ . The boolean *Served* is used to indicate whether any of the optional buses has been scheduled with an initial value being *false*.

Motivated by the fact that a bus network is designed to cover different parts of the city and it tries to avoid unnecessary overlapping among routes [2], we propose a partition-based greedy method that achieves a  $(1-\rho)(1-1/e)$  approximation ratio. Here  $\rho$  is a service overlap threshold that is used to control the cardinality of partitioned clusters. The main idea of this method contains three steps. First, it partitions the bus routes (and buses) into a set of clusters according to their service overlap. Second, the basic greedy algorithm will find the local optimal bus schedule for each of the clusters. Third, it aggregates these local optimal bus schedules from clusters to obtain the global solution.

### 4. DEMONSTRATION SCENARIOS

In this section, we first introduce the user interface of FASTS, we then present three main features of FASTS, with each feature explained using one or two scenarios. We have collected real-world bus routes (396 routes) and one-week bus touch-on/touch-off records (28 million trip records in total) in Singapore [6], to be used in this demonstration.

#### 4.1 User Interface Overview

The user interface is shown in Figure 2. The red dots on the map represent the bus stops. To facilitate the presentation, we include five rectangular boxes in Figure 2, labelled

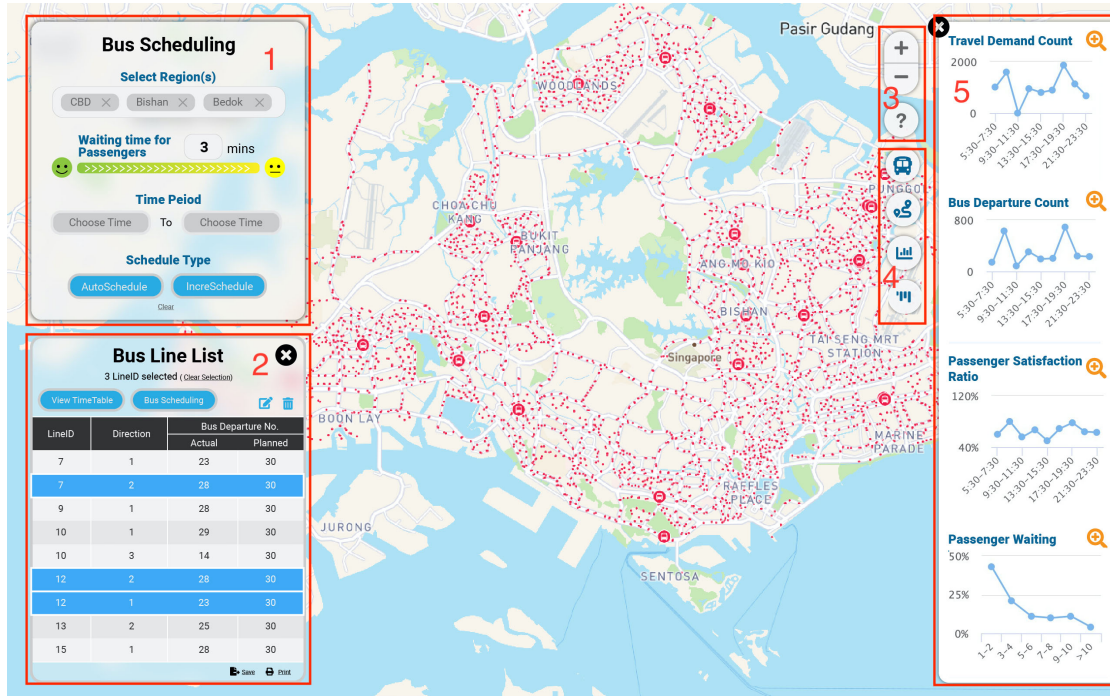


Figure 2: An Overview of User Interface

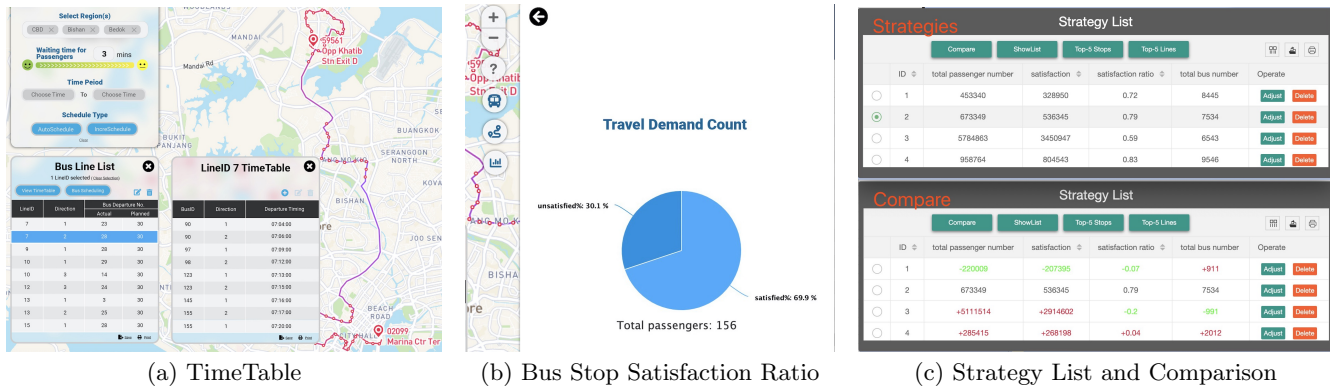


Figure 3: A Breakdown of the User Interface

by numbers from 1 to 5. Box 1 is a query panel where users could provide inputs for FASTS. Box 2 lists all the bus service routes, where users can select any route of interest to view the route on map and the bus schedule of that route. A user can zoom in or zoom out the map view to explore different levels of details by the button “+” or “-” in Box 3. The “?” button in Box 3 allows users to start a tutorial to learn how to use FASTS. Box 4 is a control panel where the four icons decide whether to show or hide bus lines, bus stop icons, the statistics reported, and the strategy list. Box 5 refers to the place where the statistics are visualized. Next, we will present the three main features of FASTS and five different scenarios where a user named Erin uses FASTS to schedule the buses.

## 4.2 Coarse-grained Scheduling

FASTS can automatically schedule bus departures within the constraints specified by the user and support different explorations with a varying number of constraints. The constraints represent the different requirements from the user. The constraints that are considered and supported

by FASTS include the planned bus departure number ( $\mathcal{P}_n$ ), the time period, the waiting time threshold ( $\lambda$ ), and the travel demands ( $\mathcal{D}$ ). Users can select one or several days bus touch-on/touch-off records from Monday to Sunday as  $\mathcal{D}$ .

**Scenario 1: Basic Scheduling.** Erin works in the transit agency. She is in charge of bus schedules in Singapore. Erin accesses our system on a browser. The first thing that she notices is the map view. Via panning, zooming-in and zooming-out, she can locate Singapore on the map. The default  $\lambda$  is 3 minutes and the default time period when bus departures could happen is set to the time window from 5:30 to 23:30. Next, she needs to input  $\mathcal{P}_n$  for each bus in the bus line list by using the “Edit” button in Box 2. The default  $\mathcal{D}$  contains the whole one-week bus touch-on/touch-off records. Then, all she needs to do is to click the “AutoSchedule” button. Based on the  $\mathcal{P}_n$  and the default parameter setting, FASTS will find an ideal bus schedule consisting of a set of selected buses and update the actual bus departure number ( $\mathcal{A}_n$ ) of each line in the bus line list. Note that  $\mathcal{P}_n$  represents the number of departures expected by the user, while  $\mathcal{A}_n$  represents the number of departures actually planned by

the system. Since the system will no longer plan a bus for a certain line when the remaining buses in this line cannot serve any more passengers,  $A_n$  is no larger than  $P_n$ .

If Erin wants to see the bus timetable, she can select several lines by clicking the row in the bus line list and open the bus timetable by clicking on the “view TimeTable” button (See Figure 3a). In addition, the bus line list and bus timetable can be exported as CSV or EXCEL files.

### 4.3 Fine-grained Scheduling with User Interaction

To further enhance its usability, FASTS supports fine-grained scheduling with user interaction. Users can specify the departure time of some buses or adjust the bus timetable according to users’ needs. In addition, users can select specific region for the task of bus scheduling.

#### Scenario 2: Incremental Adjustment on Scheduling.

There could be cases where the returned bus timetable in the first scenario is not able to meet Erin’s needs. For example, some buses originally scheduled to serve certain bus lines need to be arranged for some specific departure time because of the needs of urban development. If Erin is not happy with the current bus timetable, she has two options. First, by clicking the “+” button (Figure 3a), she can indicate a specific departure time for some bus lines before scheduling. She can schedule the remaining bus timetables by clicking the “IncreSchedule” button in Box 1 (Figure 2). Second, after the scheduling, she can remove or change the departure time of some buses in the timetable. As a result, FASTS will show changes in the number of passengers satisfied.

#### Scenario 3: Region-based Scheduling.

Erin can also focus on a specific region for the task of bus scheduling. Say Erin is tasked to find out a proper number of autonomous buses required to serve two planning areas that are currently identified as pilot regions for autonomous buses. FASTS supports two ways for users to retrieve bus scheduling results for a specific region. The first is by specifying the regions similar to a range query (see Box 1), and the second is to specify a list of bus lines or stops. Then, similar to the previous actions, by clicking the “AutoSchedule” button, the optimal bus timetable for the region will be returned.

### 4.4 Impact Analysis

FASTS provides an impact analysis of the bus schedules and easy-to-interpret visualizations to help users to understand how a schedule impacts user’s satisfaction. For each bus timetable, the impact analysis mainly includes satisfaction ratio analysis and statistics sliced in time (Scenario 4). Among different bus timetables, the impact analysis mainly compares some statistics (Scenario 5).

#### Scenario 4: Impact Analysis of Current Scheduling.

After scheduling, Erin wants to see the performance of the current scheduling, and metrics such as how many passengers will be served in each stop and the distribution of the passenger waiting time, travel demand, bus departure number and passenger satisfaction ratio could be very useful to Erin. FASTS uses some line charts to show these statistics intuitively. The third button in Box 4 shown in Figure 2 allows Erin to task FASTS to perform the analysis and display the results. The four line charts shown at the bottom of Figure 2 are just sample outputs, reporting the travel demand at different time intervals, the total bus departure number at different time intervals, passenger satisfaction ratio in dif-

ferent time intervals, and the distribution of the passenger waiting time respectively. For each chart, Erin could further enlarge it by clicking the magnifying glass symbol on the right top of the chart. In addition, Erin can explore the satisfaction ratio of the passengers waiting at this stop and the number of passengers that will be served in this stop by double-clicking the stop icon on the map (see Figure 3b).

**Scenario 5: Schedule Comparison.** Finally, Erin has several bus schedules. To make the decision, she needs to know the differences among different schedules. She has already saved schedules as strategies before. Erin can show the strategy list by clicking the fourth button in Box 4. As shown in the “Strategies” sub-part of Figure 3c, stored records are saved in the schedule list. Erin can easily explore the previous results by clicking the “Adjust” button of each line. Moreover, she can choose a schedule as a benchmark by checking the checkbox. By clicking the “Compare” button, information such as the total passenger number, number of satisfied passengers, satisfaction ratio, and total bus departure number will be shown (see the “Compare” sub-part of Figure 3c). She can also explore the top-5 bus stops or lines with highest passenger satisfaction by clicking the “Top-5 Stops” or “Top-5 Lines” button.

## 5. ACKNOWLEDGMENTS

Zhiyong Peng is supported in part by the National Key Research and Development Program of China (Project Number: 2018YFB1003400), Key Project of the National Natural Science Foundation of China (Project Number: U1811263) and the Research Fund from Alibaba Group. Zhifeng Bao is supported in part by ARC DP200102611, DP180102050. Baihua Zheng is supported in part by Prime Minister’s Office, Singapore under its International Research Centres in Singapore Funding Initiative.

## 6. REFERENCES

- [1] G. Antonides, P. C. Verhoef, and M. Van Aalst. Consumer perception and evaluation of waiting time: A field experiment. *Journal of consumer psychology*, 12(3):193–202, 2002.
- [2] M. Fletteman et al. *Designing multimodal public transport networks using metaheuristics*. PhD thesis, University of Pretoria, 2009.
- [3] M. C. Kong, F. T. Camacho, S. R. Feldman, R. T. Anderson, and R. Balkrishnan. Correlates of patient satisfaction with physician visit: differences between elderly and non-elderly survey respondents. *Health and Quality of Life Outcomes*, 5(1):62, 2007.
- [4] N. Lin, W. Ma, and X. Chen. Bus frequency optimisation considering user behaviour based on mobile bus applications. *IET Intelligent Transport Systems*, 13(4):596–604, 2019.
- [5] H. Martínez, A. Mauttone, and M. E. Urquhart. Frequency optimization in public transportation systems: Formulation and metaheuristic approach. *European Journal of Operational Research*, 236(1):27–36, 2014.
- [6] S. Mo, Z. Bao, B. Zheng, and Z. Peng. Bus frequency optimization: When waiting time matters in user satisfaction. In *International Conference on Database Systems for Advanced Applications*, 2020. To appear.