Optimizing Cross-Line Dispatching for Minimum Electric Bus Fleet

Chonghuan Wang, Yiwen Song, Guiyun Fan, Haiming Jin *Member, IEEE,*
Lu Su, *Member, IEEE,* Fan Zhang, *Member, IEEE,* Xinbing Wang, *Senior Member, IEEE*

Abstract—Recent years have witnessed the increasing popularity of electric buses (e-buses) around the globe due to their environment friendly nature. However, various factors, such as the prohibitive purchasing costs and the scarcity of large-scale charging facilities, hinder the wider adoption of e-buses. Thus, to effectively cut the cost of building and maintaining urban e-bus systems, we optimize the dispatching strategy for urban e-bus systems to satisfy public transportation demands with the minimum e-bus fleet. Specifically, we propose to systematically exploit at city-scale cross-line dispatching, a smart dispatching strategy allowing one bus to serve multiple bus lines when necessary. Technically, we construct a novel and generalizable graph-theoretic model for urban e-bus systems integrating e-buses non-negligible charging time, the spatio-temporal constraints of bus trips, and various other real-world factors. We prove that it is NP-hard, and has no $(2-\epsilon)$-approximation algorithm. Next, we propose a polynomial-time algorithm solving the problem with a guaranteed approximation ratio. Furthermore, we conduct extensive experiments on a large-scale real-world bus dataset from Shenzhen, China, which validate the effectiveness of our algorithms. As shown by our experimental results, to serve 300 bus lines, our dispatching strategy needs 38.2% less e-buses than the one currently used in practice.

Index Terms—Cross-line dispatching, electric bus, minimum fleet, combinatorial optimization.

1 Introduction

Our society is witnessing a rapid vehicle electrification process due to the environment-friendly nature of electric vehicles (EVs). Apart from electric taxis and private EVs, several countries such as the United States and China have increasingly adopted electric buses (e-buses) into their public transportation systems to advocate green commuting. In various cities around the globe (e.g., Shenzhen, Los Angeles), e-buses have started to serve urban residents' massive needs for public transportation with zero exhaust gas emission.

However, nowadays, e-buses are still far from fundamentally replacing traditional diesel buses, because of the prohibitive costs of building and maintaining an urban e-bus system. On one hand, an e-bus is usually several times more expensive than a traditional diesel bus. For example, the price of a BYD e-bus is around $\$263,000$ [1], which is approximately 2.5 times that of a diesel bus. On the other hand, a city-scale e-bus fleet usually requires a large number of charging facilities to support its daily operation, which will incur high construction and maintenance costs. Therefore, to promote the adoption of e-buses in a wider scope of cities, it is essential to reduce the above costs to a moderately affordable level. In practice, cost reduction could naturally be achieved, if the public transportation demands in a city could be served with as few number of e-buses as possible, i.e., *with the minimum e-bus fleet*.

A naive solution to shrink the size of a bus fleet serving a city is to adopt a less frequent dispatching schedule and even remove some existing bus lines. However, such method is rather undesirable, as the bus lines and schedules of a modern city are oftentimes well-developed, and thus any drastic change to them could inevitably cause severe inconvenience to the passengers. Therefore, in this paper, we propose to exploit *cross-line dispatching*, a smart dispatching strategy that allows one bus to serve multiple lines, to achieve minimum fleet without altering existing lines and schedules. The philosophy behind cross-line dispatching is to reuse idle buses of one line to serve other busy lines. For example, the lines that travel within central business districts of a city may have relatively busy schedules, whereas the ones that commute between urban and suburban areas may oftentimes schedule fewer daily trips. Clearly, a bus of the latter could be dispatched to serve the former given that it is idle. Thus, by sharing buses among different lines when necessary, cross-line dispatching is apparently a promising approach to significantly reduce the number of buses needed. Nowadays, although cross-line dispatching

- *Chonghuan Wang is with Center for Computational Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, US. He was with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. The work was done in Shanghai Jiao Tong Universtiy. E-mail: wangchonghuan@sjtu.edu.cn.*
- *Yiwen Song is with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: gavinsyw@sjtu.edu.cn.*
- *Guiyun Fan is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: fgy726@sjtu.edu.cn.*
- *Haiming Jin is with the John Hopcroft Center for Computer Science and the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: jinhaiming@sjtu.edu.cn.*
- *Lu Su is with School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana 47907, US. Email: lusu@purdue.edu.*
- *Fan Zhang is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: zhangfan@siat.ac.cn.*
- *Xinbing Wang is with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: xwang8@sjtu.edu.cn.*

(Corresponding authors: Guiyun Fan and Haiming Jin.)

has already been adopted by some cities, it is usually of very limited scale. For example, based on our extensive analysis of the bus trajectory dataset from Shenzhen[1], the 4th largest city in China, the percentage of the buses that serve multiple lines is less than 1%. Therefore, in this paper, we aim to *systematically optimize cross-line dispatching at city-scale* to minimize the overall number of e-buses needed in a urban bus system. However, such task is rather challenging in the following aspects.

The first challenge comes from the *non-negligible recharging time* of e-buses compared with the refueling time of traditional diesel buses which is usually short enough to be omitted. The omitted refueling time naturally brings negligible endurance limits for diesel buses, while the covering range limits can not be ignored in e-bus systems. Such difference requires a much more meticulous design of dispatching strategies that take into account e-buses recharging time and status. Also, such difference can not be seen as a mild one, because it makes the minimum e-bus fleet problem becomes NP-hard and in contrast, the same problem for diesel bus systems can be solved in polynomial time, as shown in Section 3. Next, it is highly non-trivial to construct a *realistic model* of an urban e-bus system with the other myriad of practical factors that have to be taken into consideration, including heterogeneous electricity consumption of different bus lines, unevenly distributed charging facilities across the city, the plethora of choices of when and where to charge the e-buses, and many others. Furthermore, optimizing city-wide e-bus dispatching is usually a *large-scale problem* due to the large numbers of lines and buses a modern city may have, which inevitably calls for dispatching algorithms with low computational complexity.

To address these challenges, we develop a novel graph-theoretic model of urban e-bus systems that captures collectively the spatio-temporal relationships among bus trips, e-buses' recharging actions, as well as various other aforementioned real-world factors. Such model has a strong generalization ability, and could be flexibly adapted according to the conditions of different real-world e-bus systems. Theoretically, we formulate the cross-line dispatching optimization problem as a combinatorial optimization over the constructed graph, which is similar to the traditional path cover problem [2], but fundamentally different from it and other existing problems. Based on our hardness analysis of the formulated problem, we show that it is not only NP-hard, but also has no $(2-\epsilon)$-approximation algorithm. Furthermore, we propose a polynomial-time approximation algorithm that approximately minimizes the number of required e-buses with a guaranteed approximation ratio.

The contributions of this paper are summarized as follows.

- To the best of our knowledge, this paper is the first work that *systematically optimizes cross-line dispatching at city-scale to minimize the number of e-buses* needed in urban e-bus systems.
- We propose a novel and practical graph-theoretic model for urban e-bus systems, which integrate a wide scope of real-world factors, and could be *easily generalized* according to the conditions of different e-bus systems.
- Theoretically, we prove that the cross-line dispatching optimization problem is NP-hard, and *has no* $(2-\epsilon)$*-approximation algorithm*. Furthermore, we propose a polynomial-time algorithm to solve the problem with a $\frac{\gamma+1}{2}\left(1+\ln\frac{2\gamma}{\gamma+1}\right)$ *approximation ratio*, where γ is the length of the longest path in the constructed graph.
- We analyze a large-scale e-bus trajectory dataset containing the trajectories of 16,359 buses within the time span from June 1st to June 30th, 2017, from Shenzhen, the 4th largest city in China, and provide a series of meaningful insights. Furthermore, we validate the performance of our algorithms based on extensive experiments over the real-world dataset.

The rest of this paper is organized as follows. Section 2 introduces some basic concepts we use throughout this paper. The formal mathematical problem formulation and the hardness analysis of the formulated problem are provided in Section 3. Then, in Section 4, we present the proposed algorithms, as well as the corresponding analysis. Section 5 shows results of our extensive experiments on a large-scale real-world dataset. Section 6 surveys the past literature related to this paper, and finally Section 7 concludes this paper.

2 PRELIMINARIES

We define several critical concepts that will be used in this paper to describe the e-bus systems in Section 2.1. Then, we construct graph-theoretical models to describe e-bus systems. In Section 2.2, we state our goal to minimize the e-bus fleet and formulate our problem onto the constructed graphs.

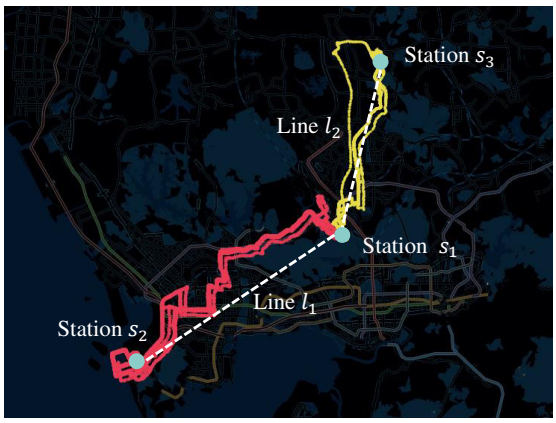
2.1 E-Bus System Model

For urban bus systems, frequent modifications of the timetables are inappropriate and may cause troubles to the public. Thus, in this paper, we consider bus systems operating under fixed timetables. The *timetable* of a bus line is a sequence of time instances within a specific period of time (e.g., one day), where each time instance represents one of the moments when a bus line dispatches a bus from its origin station. Next, we define a *dispatching task* formally in Definition 1.

Definition 1 (Dispatching Task)**.** *A dispatching task τ_i, represented by a 4-tuple $(o_i, d_i, t_{o_i}, t_{d_i})$, requires the bus system to dispatch a bus from the origin station o_i to the terminal station d_i at the departure time t_{o_i}, and the bus is expected to reach d_i at the arrival time t_{d_i}. All dispatching tasks in a bus system constitute the set $\mathcal{D} = \{\tau_i | i \in \{1, 2, \cdots, N\}\}$, where N is the total number of tasks.*

With the advancement of machine learning techniques, the time when one bus reaches its destination in a specific day can usually be well predicted through various methods, such as, spatio-temporal attentive neural network [3], graph multi-attention network [4], and many others [5, 6]. Thus, given o_i, d_i, and t_{o_i}, we could usually predict t_{d_i} with high accuracy. As mentioned in Section 1, we consider bus systems that use the *cross-line dispatching* strategy defined in Definition 2.

1. Please refer to Section 5.1 for the detailed data analysis results.

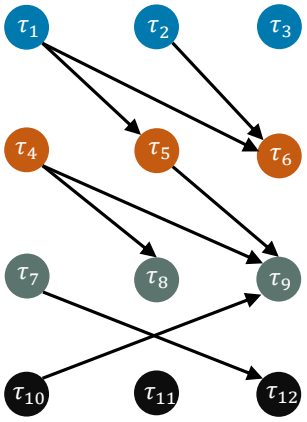


(a) Two round-trip lines l_1 and l_2 from Shenzhen's bus system.

Line	Direction	Timetable	Trip Time
l_1	s_1 to s_2	7:00 7:10 7:20	30 min
l_1	s_2 to s_1	7:25 7:35 7:45	30 min
l_2	s_1 to s_3	7:45 8:00 8:10	20 min
l_2	s_3 to s_1	7:45 8:00 8:10	20 min

Dispatching Tasks	
τ_1=(s_1, s_2, 7:00, 7:30)	τ_7=(s_1, s_3, 7:45, 8:05)
τ_2=(s_1, s_2, 7:10, 7:40)	τ_8=(s_1, s_3, 8:00, 8:20)
τ_3=(s_1, s_2, 7:20, 7:50)	τ_9=(s_1, s_3, 8:10, 8:30)
τ_4=(s_2, s_1, 7:25, 7:55)	τ_{10}=(s_3, s_1, 7:45, 8:05)
τ_5=(s_2, s_1, 7:35, 8:05)	τ_{11}=(s_3, s_1, 8:00, 8:20)
τ_6=(s_2, s_1, 7:45, 8:15)	τ_{12}=(s_3, s_1, 8:10, 8:30)

(b) Timetables and dispatching tasks of the two lines.



(c) The spatio-temporal task graph.

Figure 1: An illustration of the construction of the spatio-temporal task graph.

Definition 2 (Cross-Line Dispatching)**.** *Cross-line dispatching is a kind of dispatching strategy that allows one bus to serve multiple bus lines.*

Under such cross-line dispatching strategies, we are interested in whether different dispatching tasks could be served by one bus. Note that, if two tasks τ_i and τ_j can be served by one bus, they must satisfy the following *spatio-temporal constraints*. On one hand, task τ_i's terminal station d_i has to be the same with task τ_j's origin station o_j, i.e., $d_i = o_j$. On the other hand, the departure time of τ_j must be after the arrival time of task τ_i, i.e., $t_{d_i} < t_{o_j}$. In order to represent the tasks and their spatio-temporal relationships more clearly, we introduce the concept of *spatio-temporal task graph* as defined in Definition 3.

Definition 3 (Spatio-Temporal Task Graph)**.** *We define a spatio-temporal task graph as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents all the dispatching tasks, and there is a directed edge from τ_i to τ_j, if τ_i and τ_j satisfy the spatio-temporal constraints $d_i = o_j$ and $t_{d_i} < t_{o_j}$.*

From Definition 3, we need the timetables and locations of the origin and terminal stations as inputs to construct the graph. The edges in the graph captures all the spatio-temporal relationships among dispatching tasks we need. In real life, the timetable usually takes one minute as the smallest time unit. Hence, in this paper, the time resolution is one minute. We provide a toy example in Figure 1 to give a more clear clarification. We choose two round-trip lines from Shenzhen's bus system and assume that they follow the timetables shown in the Figure 1(b). Then, we have all the corresponding dispatching tasks given in Figure 1(b), each of which is represented by a vertex in the spatio-temporal task graph shown in Figure 1(c). For example, tasks $\tau_1 = (s_1, s_2, 7{:}00, 7{:}30)$, $\tau_5 = (s_2, s_1, 7{:}35, 8{:}05)$, and $\tau_9 = (s_1, s_3, 8{:}10, 8{:}30)$ are connected, as they satisfy the spatio-temporal constraints. In this work, we only use the terminal stations and destinations of each lines. Although some lines may share intermediate stops, one e-bus serving for two lines at the same time during one journey is rare in real life, and otherwise, some passengers will be dropped off halfway, which greatly degrades the quality of services.

A naive upper bound of the complexity of constructing the spatio-temporal graph is $O(N^2)$, where N is the total number of the dispatching tasks. However, in e-bus systems setting, at least there are no edges between the dispatching tasks from the same non-loop line and non-loop lines take up the majority of whole e-bus systems. Also, the complexity of constructing the graph heavily relies on the topologies and timetables of lines. For example, the bus lines which only travel in the suburban areas may be hard to share one e-bus with the ones commuting in busy central business districts. Therefore, our complexity is upper bounded by an acceptable polynomial complexity and can be greatly reduced given topologies and timetables of all lines.

However, the fact that two tasks are connected in the spatio-temporal task graph does not necessarily mean that they can be served by one e-bus, because the spatio-temporal task graph cannot capture e-buses' remaining electricity statuses. For example, we consider again tasks τ_1, τ_5, and τ_9 in Figure 1(c), and assume that each of these three tasks requires 20% of an e-bus's battery capacity to finish[2]. Clearly, although the three tasks are connected in the spatio-temporal task graph given in Figure 1(c), an e-bus with less than 60% remaining electricity cannot carry out them in a row without recharging. By the above observation, we define a *virtual task* in the following Definition 4.

Definition 4 (Virtual Task)**.** *A virtual task $\tau_{i,k}$ of dispatching task τ_i is defined as a 6-tuple $(o_i, d_i, t_{o_i}, t_{d_i}, e_i, e_{i,k})$, where e_i is the electricity consumption for task τ_i, and $e_{i,k}$ represents the remaining electricity level of a bus that is ready to execute task τ_i. Furthermore, the virtual task set of task τ_i is defined as $\mathcal{S}_i = \{\tau_{i,k} | e_{i,k} \geq e_i\}$, which is the set of τ_i's virtual tasks with $e_{i,k} \geq e_i$.*

Here, we discretize the electricity level (e.g., 1% of a

2. As one city usually uses e-buses of the same type, we assume without loss of generality that e-buses in our model have the same battery capacity, and we use percentages to represent the amount of electricity consumption in the rest of this paper. Note that the methods proposed in this paper can be easily extended to the scenario where e-buses have different battery capacity.

full-charges battery). First, it is not practical to see the electricity level as a continuous variable in real life. The remaining electricity people can observe on e-buses or any other electric facilities is discrete. Also, it is unnecessary to be so precise since there will surely be some variance on how much electricity one trip will consume due to different weathers, dynamic road conditions and various passengers load. Discretization can provide enough tolerance for such variance. Besides, although charging time is a continuous variable, considering the relatively slow charging speed, it will not much difference if we see one hour and five seconds as one hour. Based on such definition of virtual task, we formally define the *bus shareability graph* in Definition 5.

Definition 5 (Bus Shareability Graph)**.** *A bus-shareability graph is a directed graph $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{E}_e)$, where $\mathcal{V}_e$ represents the virtual tasks in all dispatching tasks' virtual task sets. There is a directed edge from virtual task $\tau_{i,k}$ to $\tau_{j,m}$, if an edge from τ_i to τ_j exists in the spatio-temporal task graph, and either of the following Conditions 1 and 2 on electricity levels e_i, $e_{i,k}$, and $e_{j,m}$ is satisfied.*

- ***Condition 1**: If no charging facility exists at station d_i, $e_{j,m} = e_{i,k} - e_i$ should hold.*
- ***Condition 2**: If charging facilities exist at station d_i, $e_{j,m}$ should fall into the range $[e_{i,k} - e_i, \min\{e_{i,k} - e_i + \eta(t_{d_i} - t_{o_i}), 100\%\}]$, where η denotes the charging rate.*

By such definition, an edge between two virtual tasks $\tau_{i,k}$ and $\tau_{j,m}$ exists only when task τ_i and τ_j satisfy the spatio-temporal constraints specified by the spatio-temporal task graph, and either of the conditions given in Definition 5 depending on whether charging facilities exist at station d_i.

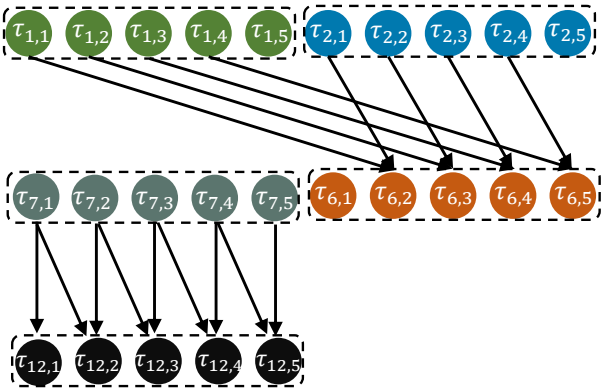


Figure 2: The bus shareability graph for tasks τ_1, τ_2, τ_6, τ_7, and τ_{12} in spatio-temporal task graph in Figure 1(c).

Figure 2 gives an example of the bus shareability graph for tasks τ_1, τ_2, τ_6, τ_7, and τ_{12} in the spatio-temporal task graph in Figure 1(c). Without loss of generality, we take 20% as the step to quantify the $e_{i,k}$ for every virtual task $\tau_{i,k}$, and thus each virtual task set $\mathcal{S}_i$ consists of 5 elements as shown in Figure 2. We assume that only station s_3 is equipped with charging facilities. For virtual task $\tau_{1,1} = (s_1, s_2, 7{:}00, 7{:}30, 20\%, 100\%)$, according to Condition 2 in Definition 5, the remaining electricity of task τ_6's virtual tasks connected with $\tau_{1,1}$ should fall into the the range $[80\%, 100\%]$. Therefore, there is an edge from $\tau_{1,1}$ to $\tau_{6,2}$. There are two directed edges from $\tau_{7,1}$ corresponding to charging and no charging. The directed edge from $\tau_{7,1}$ to $\tau_{12,1}$ means the time between these two dispatching tasks is enough to fully charge one e-bus after serving τ_7.

In fact, the bus shareability graph could be easily generalized to different scenarios by considering additional sets of rules. For example, if the bus system forbids e-buses to charge during hours when the electricity prices are excessively high, edges that involve charging during those hours could be deleted. As another example, if t_m time is required for maintenance after every task, the temporal constraint for a directed edge from task τ_i to τ_j to exist in the spatio-temporal task graph will naturally change from $t_{d_i} < t_{o_j}$ to $t_{d_i} + t_m < t_{o_j}$. Some e-bus systems allow the empty e-bus routing, which means that one bus can travel without undertaking any dispatching task just to reposition itself. In this case, we can relax the constraint $d_i = o_j$ to that one bus can travel from d_i to o_j before t_{o_j}. The specific rules could be set accordingly in different scenarios, which shows a strong generalization ability of the bus-shareability graph defined in this paper.

2.2 Problem Description

In this paper, we aim to address the problem of *finding the cross-line dispatching strategy that minimizes the number of buses in an e-bus system under given timetables*. Next, we formally describe such problem as an optimization over the bus shareability graph. First, we define the concept of a *path* of the bus shareability graph in the following Definition 6.

Definition 6 (Path)**.** *In a bus shareability graph $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{E}_e)$, a path is a sequence of vertices, where each two adjacent vertices are connected by an edge in $\mathcal{E}_e$. Note that a single vertex is regarded as a path, as well. Furthermore, all the paths of the graph $\mathcal{G}_e$ constitute the path set $\mathcal{P}$.*

By our definition of the bus shareability graph, all tasks along a path can be executed by one e-bus, and thus one path in fact corresponds to one e-bus. A path traversing a virtual task $\tau_{i,k}$ indicates that the bus represented by it can carry out the dispatching task τ_i, and a dispatching task τ_i is executed, if and only if one virtual task $\tau_{i,k}$ in its virtual task set is traversed by a selected path. Clearly, so as to satisfy the timetable, each virtual task set should be traversed by one and only one chosen path. Thus, the minimum e-bus fleet problem is equivalent to finding a series of paths with the smallest cardinality covering each virtual task set once and only once. The above problem can be described as a *virtual task set path cover (VTSPC) problem* in Definition 7.

Definition 7 (VTSPC Problem)**.** *Given a bus shareability graph $\mathcal{G}_e$, the virtual task set path cover (VTSPC) problem finds the path set $\mathcal{P}^* \subseteq \mathcal{P}$ with the minimum cardinality, referred to as the minimum VTS path cover, such that each virtual task set is traversed by one and only one path in $\mathcal{P}^*$.*

By the above definition, the VTSPC problem is different from the traditional minimum path cover problem [2] which requires each vertex of the graph to be traversed by one path. Vertex-disjoint path covering problem was studied in [7] with the objective of maximizing the total weights, which is totally different with our minimizing the number of paths. Besides, VTSPC problem is not covering the whole graph,

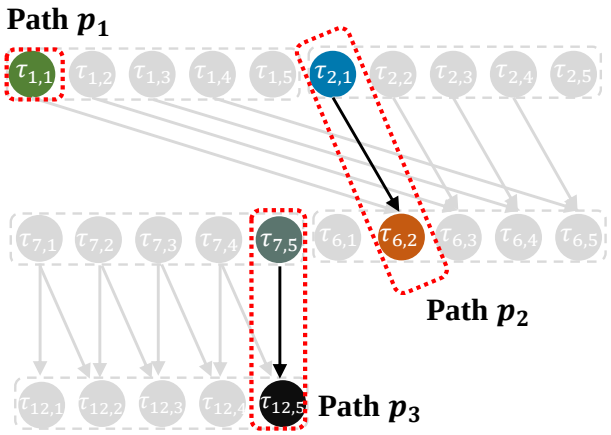


Figure 3: An illustration of the minimum VTS path cover of the bus shareability graph in Figure 2.

but exactly one virtual task in each virtual task set, which further makes our problem different from the current ones in literatures. In Figure 3, we demonstrate the minimum VTS path cover $\mathcal{P}^* = \{\{\tau_{1,1}\}, \{\tau_{2,1}, \tau_{6,2}, \{\tau_{7,5}, \tau_{12,5}\}\}$ of the bus shareability graph in Figure 2.

3 Mathematical Formulation

In this section, we formulate the VTSPC problem given in Definition 7. We start with formulating in Section 3.1 a special yet meaningful case of VTSPC, where the size of each virtual task set is one. Note that such special case serves as preliminary for, and sheds lights upon the formulation of the general VTSPC problem formulated in Section 3.2.

3.1 VTSPC Problem with Single Element Virtual Task Set

In this section, we formulate the VTSPC problem with single element virtual task set (se-VTSPC problem), which is equivalent to *finding the minimum path cover over the spatio-temporal task graph*. Note that, once fueled, the distance that a diesel bus could cover is orders of magnitude greater than the route length[3], and the refueling time of a diesel bus is usually short enough to be omitted. Thus, a path in the spatio-temporal task graph could be regarded as a diesel bus, and the vertices traversed by the path correspond to the tasks carried out by it. Therefore, similar to the VTSPC problem, the se-VTSPC problem can be regarded as *finding the cross-line dispatching strategy that minimizes the number of buses with timetable constraints in traditional diesel bus systems*.

The mathematical formulation of the se-VTSPC problem is given in the following integer-linear optimization program.

$$\textbf{se-VTSPC}: \min \sum_{i:p_i \in \mathcal{P}} x_i \tag{1}$$

$$\text{s.t.} \sum_{i:\tau_j \in p_i, p_i \in \mathcal{P}} x_i = 1, \quad \forall \tau_j \in \mathcal{V}, \tag{1a}$$

$$x_i \in \{0,1\}, \quad \forall p_i \in \mathcal{P}. \tag{1b}$$

3. For example, the fuel consumption of a HIGER KLQ6129GAE52 diesel bus is 29L every 100 kilometers [8], and the capacity of its fuel tank is 270L.

The se-VTSPC problem takes the spatio-temporal task graph $\mathcal{G}$ and the path set $\mathcal{P}$ as inputs. Each path $p_i \in \mathcal{P}$ corresponds to a boolean variable x_i. $x_i = 1$ indicates that path p_i is chosen to be in the selected path set $\mathcal{P}^*$, and otherwise $p_i \notin \mathcal{P}^*$. The objective function $\sum_{i:p_i \in \mathcal{P}} x_i$ given by Equation (1) is exactly the total number of chosen paths. Constraint (1a) means for each task τ_j, only one chosen path can traverse τ_j. Constraint (1b) guarantees that each x_i is a boolean variable.

Finding the minimum path cover of an arbitrary directed graph is NP-hard [9], and hence computationally infeasible for large graphs. However, the minimum path cover can be found in polynomial time if the graph is acyclic [9]. In the following Theorem 1, we prove that the spatio-temporal task graph is acyclic, which indicates that the se-VTSPC problem can be solved optimally in polynomial time.

Theorem 1. *Any spatio-temporal task graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as defined in Definition 3 is acyclic.*

Proof. We carry out the proof of this theorem by contradiction. Assume a cyclic path exists in some spatio-temporal task graph $\mathcal{G}$. Without loss of generality, let $P = \{(v_1, v_2), (v_2, v_1)\}$ be a cyclic path. v_1 and v_2 are two vertices representing two dispatching tasks $(s_1, s_2, t_{o_1}, t_{d_1})$ and $(s_2, s_1, t_{o_2}, t_{d_2})$. $t_{o_1}, t_{d_1}, t_{o_2}$ and t_{d_2} are the time of departure and arrival of the two dispatching tasks, respectively. If there are two edges between v_1 and v_2, the two tasks have to satisfy the temporal constraints $t_{d_1} < t_{o_2}$, $t_{d_2} < t_{o_1}$. With the $t_{o_2} < t_{d_2}$, we have $t_{d_1} < t_{o_2} < t_{d_2} < t_{o_1}$ which implies that $t_{d_1} < t_{o_1}$, i.e., the bus finishes the trip before it starts. Thus, we have a contradiction, and arrive at the conclusion that any spatio-temporal task graph defined in Definition 3 is acyclic. □

More specifically, the problem of finding the minimum path cover in a directed acyclic graph is equivalent to the well known maximum bipartite matching problem [10], which can be solved by the Hungarian algorithm [11] or using the Hopcroft-Karp algorithm [12] in time $O(|\mathcal{E}|\sqrt{|\mathcal{V}|})$.

Different from se-VTSPC problem we discussed in this section, the general VTSPC problem is NP-hard. Although the difference of sizes of each virtual task set seems mild, such difference significantly impacts the difficulty of our problem. Moreover, as se-VTSPC problem can be seen as the dispatching of traditional diesel buses, optimizing on e-bus systems is much more difficult and worths studying.

3.2 General VTSPC Problem

As defined in Section 2.2, the VTSPC problem aims to find the minimum VTS path cover $\mathcal{P}^*$, which traverses each virtual task set exactly once. The formal mathematical formulation of the VTSPC problem is given in the following integer-linear optimization program.

$$\textbf{VTSPC}: \min \sum_{i:p_i \in \mathcal{P}} x_i \tag{2}$$

$$\text{s.t.} \sum_{i:v_{j,k} \in \mathcal{V}_j \bigcap p_i} x_i = 1, \quad \forall \tau_j \in \mathcal{D}, \tag{2a}$$

$$x_i \in \{0,1\}, \quad \forall p_i \in \mathcal{P}. \tag{2b}$$

The general VTSPC problem takes the bus shareability graph $\mathcal{G}_e$, the task set $\mathcal{D}$, and the path set $\mathcal{P}$ as inputs. Each path $p_i \in \mathcal{P}$ corresponds to a boolean variable x_i. $x_i = 1$ indicates that path p_i is chosen in the minimum VTS path cover $\mathcal{P}^*$, otherwise $p_i \notin \mathcal{P}^*$. The objective function $\sum_{i:p_i \in \mathcal{P}} x_i$ given by Equation (2) is the total number of paths chosen in $\mathcal{P}^*$. Constraint (2a) is to ensure that exactly one chosen path in $\mathcal{P}^*$ traverses each virtual task set $\mathcal{V}_j$, and Constraint (2b) restricts x_i to be a boolean variable.

The major difference between the se-VTSPC and VTSPC problems lies in Equation (1a) and (2a). Instead of enforcing each vertex to be traversed by one path, Equation (2a) requires that exactly one vertex $\tau_{i,k}$ in each $\mathcal{S}_i$ lies in one chosen path. Note that such difference fundamentally changes the hardness of the problem, as will be shown in the rest of this section. Intuitively, such difference is just incurred by whether we should consider the charging and energy constratins, which differs our problem from the tradition diesel bus optimization and highlights our novelty and contributions.

Although the se-VTSPC problem has been shown solvable in polynomial time in Section 3.1, the decision version of VTSPC is in fact NP-complete. Next, we state the decision version of the VTSPC problem as follows.

- **INSTANCE:** Bus shareablility graph $\mathcal{G}_e$ and positive integer K.
- **QUESTION:** Does $\mathcal{G}_e$ has a path set of size at most K which traverses each virtual task set exactly once?

Next, in Theorem 2, we prove the NP-completeness of VTSPC's decision version.

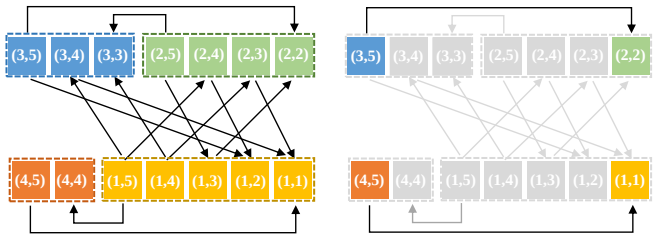
Theorem 2. *The decision version of the VTSPC problem is NP-complete.*

Proof. We construct a reduction from the Bin Packing problem to the VTSPC problem.

Let U be the finite set of items, with a size $s(u_i) \in \mathbb{Z}^+$ for each $u_i \in U$, and every bin has a fixed integer capacity B. The *bin packing problem* asks whether there is a partition of U into disjoint sets $U_1, U_2, \ldots, U_K$ such that the sum of the sizes of the items in each U_i is B or less, given any positive integer K [13]. Note that if the capacity B goes to infinity, the bin packing problem can be directly solved. Therefore, we assume $B \leq C$ with C being a sufficiently large constant, which does not affect the hardness of the bin packing problem.

Next, we describe our way of constructing a shareability graph. First, we treat each item u_i as a task, and define the virtual task and virtual task set of it as follows. A virtual task of u_i is defined as a two-tuple $u_{i,k} = (s(u_i), c_{i,k})$, where $c_{i,k}$ is the remaining space of one bin when u_i is decided to be packed into it. Then, the virtual task set of u_i is defined as $\mathcal{U}_i = \{(s(u_i), c_{i,k}) | s(u_i) \leq c_{i,k}\}$. Thus, the shareability graph of any given bin packing instance can be constructed as $\widetilde{G} = (\widetilde{V}, \widetilde{E})$, where $\widetilde{V}$ represents all the virtual tasks with each vertex $\widetilde{v}_{i,k} \in \widetilde{V}$ representing a virtual task $u_{i,k}$. There is a directed link from vertex $\widetilde{v}_{i,k}$ to $\widetilde{v}_{j,m}$, if $c_{i,k} - s(u_i) = c_{j,m}$, i.e., $\widetilde{E} = \{(\widetilde{v}_{i,k}, \widetilde{v}_{j,m}) | c_{i,k} - s(u_i) = c_{j,m}\}$. With the shareability graph $\widetilde{G}$, the bin packing problem is to find K paths in the graph to traverse each virtual task set exactly once, and every path can be seen as a bin. Then, the bin packing problem can be reduced to VTSPC on the shareability graph $\widetilde{G}$. We claim that the bin packing problem has a solution within K bins, if and only if we can find at most K paths in $\widetilde{G}$ to cover all the set once. If we can solve VTSPC efficiently, the same algorithm can be applied to tackle the bin packing problem. Therefore, we successfully construct a reduction from bin packing to VTSPC.

Here we give an example to show how the reduction works in Figure 4. We have four four items with size 3, 2, 4 and 1 with capacity $B = 5$. Figure 4(a) is the shareability graph we constructed. The vertices in the same color are from the same virtual task set. The first blue vertex labeled with $(3, 5)$ represents the size of this item is three and before it is put in a bin, the remaining capacity of the bin is 5. There is an edge between (3.4) and $(1, 1)$ because after taking the item with size 3 into the bin, the remaining capacity 1 is enough to hold the item with size 1. In Figure 4(b), we show an eligible virtual task set path covering. The path $\{(3, 5), (2, 2)\}$ is interpreted as we can put the items corresponding to the blue and green vertices together in one bin. Then, we want to ask whether K paths are enough to cover all the set exactly once. That is just our VTSPC problem.



(a) An example of the shareability graph for bin packing problem.

(b) An example of an eligible virtual task set path covering.

Figure 4: An illustration on how to reduce from bin packing to VTSPC.

The complexity of our reduction is just the complexity of construction the graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$. It takes $\sum_i s(u_i)$ steps to construct all the nodes, which can be upper bounded by $|U|C$. As for constructing the edges, the complexity can never exceed $O(|U|^2C^2)$, which is the complexity of constructing a complete graph. Thus, the reduction can be finished in polynomial time with order $O(|U|^2C^2)$. Besides, given a set of path set, it is easy to check whether all the virtual task set is traversed exactly once, which guarantees the decision version of VTSPC is NP. Therefore, the decision version of VTSPC is NP-complete. □

As the decision version of VTSPC is already NP-complete, we naturally have the following Corollary 1 on the NP-hardness of the VTSPC problem itself.

Corollary 1. *The VTSPC problem is NP-hard.*

Next, we show that VTSPC is not only NP-hard, but also hard to approximate, by *a gap-introducing reduction* from the *VTS Hamiltonian path (VTS-HP) problem* defined in Definition 8 to VTSPC.

Definition 8 (VTS-HP Problem). *Given a shareability graph $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{E}_e)$, a VTS Hamiltonian path is a path in $\mathcal{G}_e$ that*

traverses each virtual task set exactly once. The problem of finding such VTS Hamiltonian path is referred to as the VTS Hamiltonian path (VTS-HP) problem.

Next, we state in the following Lemma 1 that the VTS-HP problem is NP-complete.

Lemma 1. *The VTS-HP problem is NP-complete.*

Proof. We prove this theorem by performing a reduction from the NP-complete directed Hamiltonian path problem [13]. There is an obvious observation that if each virtual task set $\mathcal{S}_i$ owns only one element, i.e., every vertex itself is a virtual task set, the VTS-HP problem is just the traditional *directed Hamiltonian Path* problem which is *NP-complete* [13]. Thus if the VTS-HP problem can be solved in polynomial time perfectly, the directed Hamiltonian Path problem can also be solved efficiently, because it is simply a special case of the VTS-HP problem. In other words, the directed Hamiltonian path problem is a special case of the VTS-HP problem. It is obvious that, given a path, we can easily judge whether it traverses each virtual task set. Therefore, the VTS-HP problem is NP-complete. □

Based on Lemma 1, we next perform a gap-introducing reduction from VTS-HP to VTSPC to show the hardness of approximation in the following Theorem 3.

Theorem 3. *$\forall \epsilon > 0$, there is no factor $(2-\epsilon)$-approximation polynomial-time algorithm for the VTSPC problem.*

Proof. Assuming that $f(\cdot)$ is a mapping from one instance of VTS-HP to one instance of VTSPC, we set $f(G) = G$, where G is the graph constructed in the VTS-HP problem. It means VTS-HP and VTSPC share the same input graph. If there exists a VTS Hamiltonian path in graph G, the path itself is the optimal solution for the VTSPC problem. Otherwise, the optimal solution of VTSPC will contain at least two paths. Thus, we introduce an NP-hard gap $[1, 2-\epsilon]$. According to the (α, β) gap reduction in [2], we know that there is no factor $(2-\epsilon)$-approximation algorithm for VTSPC. □

The idea behind the proof is that if there exits a VTS Hamiltonian path in the shareability graph $\mathcal{G}_e$, then the optimal value of VTSPC problem is just 1, since the VTS Hamiltonian path can be one of the optimal solutions. However, in Lemma 1, we have shown that deciding the existence of VTS Hamiltonian path is NP-complete. That means the best result a polynomial-time algorithm can guarantee is 2. Otherwise, the VTS-HP problem can be well solved. Here, we have a gap ratio of 2.

4 Proposed Algorithms

As shown in Section 3.2, although se-VTSPC can be solved efficiently, the general VTSPC problem is NP-hard and even hard to approximate. Therefore, in Section 4.1, we propose the *BM-se-VTSPC* algorithm based on the Hungarian algorithm [11] to get the optimal solution of the se-VTSPC problem in polynomial time, and in Section 4.2, we propose the *LPF-VTSPC* algorithm for the general VTSPC problem and derive its approximation ratio.

4.1 BM-se-VTSPC Algorithm

As formulated in Section 3.1, the se-VTSPC problem is equivalent to finding the minimum path cover in the spatio-temporal task graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, which can be realized by utilizing the existing algorithm given in [9], referred to as the *bipartite matching-based se-VTSPC (BM-se-VTSPC) algorithm* in this paper. As details of the BM-se-VTSPC algorithm could be found in [9], instead of providing the detailed procedures, we only describe the major intuitions behind the algorithm in the rest of this section.

The BM-se-VTSPC algorithm starts with constructing a bipartite graph $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}'\}$ based on the spatio-temporal task graph $\mathcal{G}$, by splitting each vertex τ_i into two vertices τ_i^x and τ_i^y, and connecting τ_i^x to τ_j^y, if an edge from τ_i to τ_j exists in $\mathcal{G}$, i.e., $\mathcal{V}' = \{\tau_1^x, \cdots, \tau_N^x\} \bigcup \{\tau_1^y, \cdots, \tau_N^y\}$, and $\mathcal{E}' = \{(\tau_i^x, \tau_j^y) | (\tau_i, \tau_j) \in \mathcal{E}\}$.

By graph theory [10], we have that the minimum path cover of the directed acyclic graph $\mathcal{G}$ and the maximum matching of the bipartite graph $\mathcal{G}'$ satisfies

$$|\text{Min. path cover of } \mathcal{G}| = |\mathcal{V}| - |\text{Max. matching of } \mathcal{G}'| \,. \quad (3)$$

Hence, the BM-se-VTSPC algorithm then utilizes the Hopcroft-Karp algorithm [12] to find the maximum matching $\mathcal{M}$ of graph $\mathcal{G}'$, and let $\{(\tau_i, \tau_j) | (\tau_i^x, \tau_j^y) \in \mathcal{M}\}$ be the edges that constitute the chosen paths.

It is proved that the above BM-se-VTSPC algorithm has the computational complexity with the order $O(|\mathcal{E}| \sqrt{|\mathcal{V}|})$ [9].

4.2 Longest Path First VTSPC Algorithm

In this section, we present our *Longest Path First VTSPC (LPF-VTSPC) algorithm*, which solves the VTSPC problem with a guaranteed approximation ratio.

Our design philosophy of this algorithm is as follows. If the virtual task to be traversed in each virtual task set is given, the VTSPC problem reduces to the se-VTSPC problem, whose optimal solution can be obtained in polynomial time. As going through all combinations of vertices is computationally intractable, we devise an efficient searching scheme by looking for the longest path in the bus shareability graph until all the virtual task sets are traversed. Although searching the longest path in a general graph is NP-hard [13], such operation can be done in polynomial time over the bus shareability graph, because it is a directed acyclic graph. The LPF-VTSPC algorithm is shown in the following Algorithm 1, and a detailed complexity analysis of the algorithm is provided in Theorem 5.

The input of the LPF-VTSPC algorithm is the bus shareability graph $\mathcal{G}_e$. Algorithm 1 starts with initializing an empty chosen path set $\widehat{\mathcal{P}}$ (line 1). The main loop (line 2-7) is the process of iteratively searching the longest paths, as long as the remaining graph $\mathcal{G}_e$ is non-empty. Specifically, in each iteration, the algorithm searches the longest path p in the current graph using topological sorting with $O(|\mathcal{V}_e| + |\mathcal{E}_e|)$ computational complexity (line 3). Next, it lets $\overline{\mathcal{V}}$ be the set of vertices in the virtual task sets traversed by path p (line 5), removes $\overline{\mathcal{V}}$ from the current vertex set $\mathcal{V}_e$ (line 6), and removes the edges incident to these vertices from the current edge set $\mathcal{E}_e$ (line 7). Finally, the algorithm returns the VTS path cover $\widehat{\mathcal{P}}$ (line 8).

Algorithm 1: The LPF-VTSPC Algorithm

Input: bus shareability graph $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{E}_e)$;
Output: VTS path cover $\widehat{\mathcal{P}}$;

$\widehat{\mathcal{P}} \leftarrow \emptyset$;
while $\mathcal{G}_e \neq \emptyset$ **do**
  find the longest path p of $\mathcal{G}_e$ using topological sorting;
  $\widehat{\mathcal{P}} \leftarrow \widehat{\mathcal{P}} \cup \{p\}$;
  // Delete all vertices in the virtual task sets traversed by p, and the edges incident to them.
  $\overline{\mathcal{V}} \leftarrow \{\tau_{i,k} \in \mathcal{S}_i | \mathcal{S}_i \bigcap p \neq \emptyset\}$;
  $\mathcal{V}_e \leftarrow \mathcal{V}_e \setminus \overline{\mathcal{V}}$;
  $\mathcal{E}_e \leftarrow \mathcal{E}_e \setminus \{(\tau_{i,k}, \tau_{j,l}) \in \mathcal{E}_e | \tau_{i,k} \in \overline{\mathcal{V}} \text{ or } \tau_{j,l} \in \overline{\mathcal{V}}\}$;
return $\widehat{\mathcal{P}}$;

Now, we need to verify the eligibility of the longest path, i.e., the longest path should not traverse the same virtual task set more than once. We formally state the result in Therorem 4.

Theorem 4. *For any given bus shareability graph $\mathcal{G}_e$, any path $\mathcal{P}$ in $\mathcal{G}_e$ will not traverse one virtual task set more than once.*

Proof. We carry out the proof by contradiction. Assume a path traverse one virtual task twice. Without loss of generality, let $\mathcal{P}$ traverse the virtual task sets corresponding to $v_1 = (s_1, s_2, t_{o_1}, t_{d_1})$, $v_2 = (s_2, s_1, t_{o_2}, t_{d_2})$ and v_1 sequentially (i.e., $\mathcal{P}$ traverse virtual task of v_1 twice). The edge from v_1 to v_2 indicates $t_{d_1} < t_{o_2}$ and the one from v_2 to v_1 implies $t_{d_2} < t_{o_1}$. With the $t_{o_2} < t_{d_2}$, we have $t_{d_1} < t_{o_2} < t_{d_2} < t_{o_1}$ which implies that $t_{d_1} < t_{o_1}$, i.e., the bus finishes the trip before it starts. Thus, we have a contradiction, and draw the conclusion that any path $\mathcal{P}$ in $\mathcal{G}_e$ will not traverse one virtual task set more than once. □

Next, in Theorem 5, we prove that Algorithm 1 has a polynomial-time computational complexity.

Theorem 5. *The computational complexity of Algorithm 1 is $O\big(|\mathcal{V}_e|^2 + |\mathcal{E}_e||\mathcal{V}_e|\big)$.*

Proof. In the bus shareability graph $\mathcal{G}_e$, which is a DAG, the computational complexity of finding the single source longest path by topological sorting is $O(|\mathcal{V}_e| + |\mathcal{E}_e|)$ [10]. In the bus shareability graph, only one virtual task for each bus line that leaves the origin station earliest can be the candidate source for the longest path. The complexity in each iteration is $O\big(k(|\mathcal{V}_e| + |\mathcal{E}_e|)\big)$ where k is the number of lines. It is obvious that $k \ll |\mathcal{V}_e| + |\mathcal{E}_e|$, and thus $O\big(k(|\mathcal{V}_e| + |\mathcal{E}_e|)\big) \approx O(|\mathcal{V}_e| + |\mathcal{E}_e|)$. In the worst case, the number of iterations can be no larger than $|\mathcal{V}_e|$. Therefore, the total complexity is $O(|\mathcal{V}_e|^2 + |\mathcal{E}_e|\,|\mathcal{V}_e|)$. □

Next, we provide formal theoretical analysis of the approximation ratio of Algorithm 1 in the following Theorem 6.

Theorem 6. *The approximation ratio of Algorithm 1 is $\frac{\gamma+1}{2}\big(1 + \ln\frac{2\gamma}{\gamma+1}\big)$, where γ denotes the length of the longest path in the bus shareability graph $\mathcal{G}_e$.*

Proof. We start the proof of this theorem by introducing some extra notations. We use OPT to denote the optimal solution for VTSPC, OPT_i to denote the optimal solution after we have picked up i longest paths, i.e., it is the optimal solution after the ith iteration of Algorithm 1, and f_i to denote the number of virtual task sets that have been covered after i iterations. Thus, the remaining number of virtual task sets that need to be covered is $N - f_i$ after i iterations, and we arrive at the following conclusion that

$$\text{OPT}_i \leq N - f_i, \tag{4}$$

which means that the optimal number of paths can never exceed the remaining number of virtual task sets, as one path has to cover at least one virtual task. Furthermore, we have another observation on the relationship between OPT and f_i, shown in the Lemma 2.

Lemma 2. *The optimal solution after i iterations OPT_i cannot be greater than the sum of the number of vertices covered in the ith iteration and OPT_{i-1}, i.e.,*

$$\text{OPT}_i \leq (f_i - f_{i-1}) + \text{OPT}_{i-1}. \tag{5}$$

As the proof of Lemma 2 is intuitive, we omit it in this paper. Intuitively, if we take one vertex off one path, the path breaks at most into two paths. If we take two off, the path breaks at most into three paths. Based on this idea, the OPT_{i-1} paths will break at most into $(f_i - f_{i-1}) + \text{OPT}_{i-1}$ paths, because in the ith iteration we cover $f_i - f_{i-1}$ virtual task sets. Therefore, we can confirm that there is a feasible solution generated from the optimal solution of last iteration whose cardinality is at most $(f_i - f_{i-1}) + \text{OPT}_{i-1}$. The optimal solution of OPT_i is thus surely no greater than $(f_i - f_{i-1}) + \text{OPT}_{i-1}$. In this way, we iteratively expand OPT_i using Inequality (5) as

$$\begin{aligned}\text{OPT}_i &\leq (f_i - f_{i-1}) + \text{OPT}_{i-1} \leq (f_i - f_{i-2}) + \text{OPT}_{i-2} \\ &\leq \cdots \leq f_i + \text{OPT}.\end{aligned} \tag{6}$$

Thus far, we have two constraints on the OPT_i, i.e., Constraints (4) and (6). Note that the term f_i on the right hand side of the two constraints has different signs. The right hand side of Inequality (4) monotonically decreases with the process, and Inequality (6) is exactly the opposite, which are vital properties that will be used in the following proof.

Based on the different monotonicity of the right hand side of Inequalities (4) and (6), they alternate to be a tighter bound for OPT_i. When Inequality (6) is tighter,

$$f_i + \text{OPT} \leq N - f_i,$$

which can easily be simplified as $f_i \leq \frac{N - \text{OPT}}{2}$. We then divide the scale-down process into two steps by the threshold k (i.e., $f_k \leq \frac{N-\text{OPT}}{2} < f_{k+1}$).

For the iteration $i \leq k$, as we use a greedy algorithm to find the longest path, we have

$$f_i - f_{i-1} \geq \frac{N - f_{i-1}}{\text{OPT}_{i-1}}.$$

With Constraint (6), which is tighter, we have that

$$f_i - f_{i-1} \geq \frac{N - f_{i-1}}{\text{OPT}_{i-1}} \geq \frac{N - f_{i-1}}{f_{i-1} + \text{OPT}} \geq \frac{N - f_{i-1}}{\frac{1}{2}(N + \text{OPT})}.$$

The above inequality can be transformed into

$$\Big(\frac{1}{2}N+\frac{1}{2}\mathrm{OPT}\Big)(N-f_i)\le\Big(\frac{1}{2}N+\frac{1}{2}\mathrm{OPT}-1\Big)(N-f_{i-1}).$$

Again, by using an iterative method, we have

$$\begin{aligned}(N-f_i)&\le\Big(1-\frac{2}{N+\mathrm{OPT}}\Big)(N-f_{i-1})\le\cdots\\&\le\Big(1-\frac{2}{N+\mathrm{OPT}}\Big)^i N\le N\exp\Big(-\frac{2k}{N+\mathrm{OPT}}\Big).\end{aligned}$$

With $f(A_i)\le\frac{N-\mathrm{OPT}}{2}$, we have

$$\frac{N+\mathrm{OPT}}{2}\le N\exp\Big(-\frac{2k}{N+\mathrm{OPT}}\Big).$$

Then, we can obtain

$$k\le\frac{N+\mathrm{OPT}}{2}\ln\Big(\frac{2N}{N+\mathrm{OPT}}\Big).$$

In the second stage, the rest number of vertices are no greater than $\frac{N+\mathrm{OPT}}{2}$, and the total step g satisfies

$$g\le k+\frac{N+\mathrm{OPT}}{2}=\frac{N+\mathrm{OPT}}{2}\Big(1+\ln\frac{2N}{N+\mathrm{OPT}}\Big).$$

Then, ratio of $\frac{g}{\mathrm{OPT}}$ satisfies that

$$\frac{g}{\mathrm{OPT}}\le\frac{\frac{N}{\mathrm{OPT}}+1}{2}\left(1+\ln\frac{2}{1+\frac{\mathrm{OPT}}{N}}\right).$$

As $\frac{N}{\mathrm{OPT}}\le\gamma$, where γ is the length of the longest path,

$$\frac{g}{\mathrm{OPT}}\le\frac{\gamma+1}{2}\left(1+\ln\Big(\frac{2\gamma}{\gamma+1}\Big)\right).$$

Thus far, we have finished the proof of this theorem. □

If the length γ of the longest path in the bus shareability graph is 1, which means no vertices are connected with each other, the approximation ratio is 1. In such extreme case, the output of the LPF-VTSPC algorithm coincides with the optimal solution. In real-world systems, the value of γ is typically small enough so that the approximation ratio given in Theorem 6 is meaningful in practice.

In real world e-bus systems, sometimes we want to avoid keeping the same bus continuously traveling for extremely long time in one day. For instance, eight hours may seem reasonable for one e-bus. Under such constraint, our LPF-VTSPC algorithm can be easily modified to satisfy. Since we are finding the longest path in each iteration, if one path is longer than eight hours, we just take the dispatching tasks covered in the first eight hours and keep the remaining tasks in the graph waiting for other paths to cover them.

5 Experiments

In this section, we describe and analyze the large-scale e-bus dataset utilized in our experiments, provide our insights based on analysis of the dataset, evaluate our algorithms by extensive experiments, and present our experiment results.

5.1 Dataset and Analysis

Our experiments in this paper are based on a 311GB historical GPS dataset of 16,359 e-buses from Shenzhen, the 4th largest city in China. The time span of the dataset is from June 1st to June 30th, 2017. Every piece of record consists of 16 parameters describing the current operating status of a bus, among which the ones that are relevant to this paper include *system time*, *bus license plate number*, *line number*, *current longitude and latitude*, *current speed*, and *current direction*. Moreover, if a bus is approaching a bus stop, there will be 3 more parameters, i.e., *ID of the bus stop*, *approaching time*, and *ID of the next bus stop*. An example piece of data in the dataset is shown in Table 1. Unfortunately, we do not have the information related to electricity consumption in our dataset. We measure the lengths of the trajectories of each line and compare them with the maximal covering range of a fully-charged battery to get estimations. Next, in the following Figures 8-11, we show the results of our data analysis based on this dataset.

Figure 5: An illustration of the spatial distribution of the 12,229 bus stops' locations in Shenzhen, where the larger and brighter the point is, the more bus lines pass the stop.

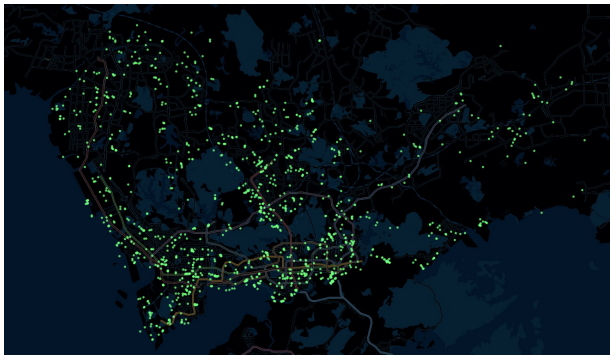


Figure 6: An illustration of the spatial distribution of the 1,670 origin and terminal stations in Shenzhen.

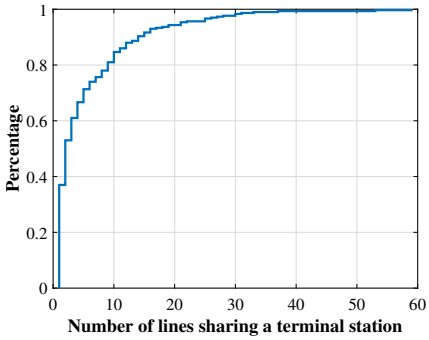


Figure 7: CDF of the number of lines passing a terminal station

System Time	Bus License Number	Line Number	Longitude	Latitude
2017-6-1 0:0:29	BS03981D	03060	114.133835	22.564110
Speed	**Direction**	**Bus Stop ID**	**Approaching Time**	**ID of Next Stop**
3036.47	44	F_BJ0108	17-6-1 0:0:28	F_BJ0141

Table 1: An example piece of data extracted from the dataset.

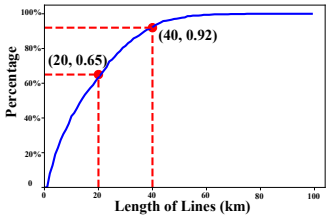


Figure 8: CDF of the lengths of bus

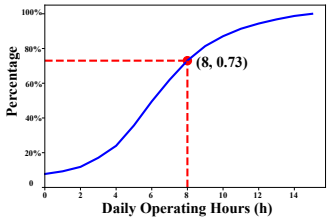


Figure 9: CDF of e-buses' operating

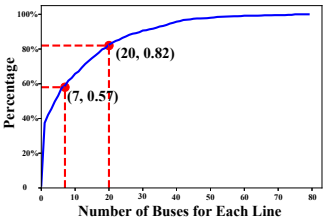


Figure 10: CDF of each line's number of buses.

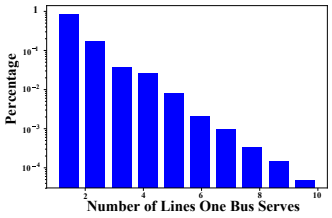


Figure 11: PDF of the number of lines a bus serves.

- **Bus stops**. There are 12,229 bus stops recorded in the dataset. The operating frequency and spatial distribution of these stops are shown in Figure 5. The larger and brighter the point is, the more bus lines pass this stop. Figure 5 demonstrates that the most frequently used stops are located in urban areas, although the stops are widely distributed across the whole city.
- **Origin and terminal stations**. Origin and terminal stations are different from intermediate bus stops in our problem, because charging facilities are usually located near these stations [1]. Figure 6 shows the spatial distribution of the 1,670 origin and terminal stations in Shenzhen, which we obtain from our dataset. We can observe that the density of the origin and terminal stations is much higher in downtown than in rural areas. Figure 7 shows that about 30% stations play a role as the terminal station of 1 line. For the busiest station, 59 lines take it as their terminal station.
- **Bus line lengths**. The CDF of the lengths of bus lines in Shenzhen is given in Figure 8, which shows that 65% of the bus lines are no more than 20 kilometers, and 92% of the bus lines are no more than 40 kilometers. Therefore, short- and medium-range ones constitute the majority of the bus lines in Shenzhen.
- **Daily operating hours**. Figure 9 shows the CDF of the daily operating hours of the e-buses in Shenzhen, from which we observe that such distribution is highly unbalanced. Specifically, more than 80% of the e-buses serve less than 8 hours every day. Furthermore, we observe that more than 83% bus lines only operate during 6:00-23:00, whereas there are still 16% of them operating during 23:00-6:00. Among those that operate during 23:00-6:00, there are 4 lines that only operate during those hours.
- **Number of buses per line**. From our dataset, we observe that some lines have many buses serving them, whereas others have fewer. Figure 10 shows that 57% bus lines have less than 7 buses, but 20% of them have more than 20 buses.
- **Cross-line serving**. Figure 11 shows the number of lines that an e-bus serves in Shenzhen based on our dataset. We can observe that although there are already e-buses serving multiple lines, more than 99% e-buses still serve only one line, which leaves great room for us to explore the power of cross-line dispatching in Shenzhen's e-bus system.

5.2 Experiment Results

5.2.1 *Experiments for Evaluating BM-se-VTSPC*

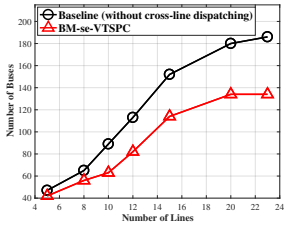


Figure 12: BM-se-VTSPC with multiple lines that share the station s_5.

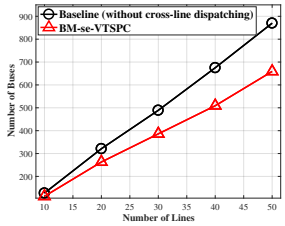


Figure 13: BM-se-VTSPC with multiple randomly chosen bus lines.

We evaluate the performance of our BM-se-VTSPC algorithm on our real-world dataset, and compare it with the baseline dispatching algorithm that *does not consider cross-line dispatching*, where we run BM-se-VTSPC for each line separately and sum all the results.

Figure 12 shows our experiment result for the scenario where 23 bus lines share the bus station s_5 with GPS coordinate (114.058, 22.537) as the origin or terminal stations. This figure shows that our BM-se-VTSPC algorithm obviously outperforms the baseline strategy without cross-line dispatching. As the number of lines increases, the advantage of BM-se-VTSPC becomes more and more obvious. Figure 13 corresponds to the experiments, where we consider 10, 20, 30, 40, and 50 randomly chosen bus lines in Shenzhen, which do not necessarily share origin or terminal stations. From this figure, we can observe that BM-se-VTSPC reduces 25% buses needed by the baseline strategy, when 50 lines are jointly optimized.

Figure 14 corresponds to a city-scale experiment where 836 lines operating in Shenzhen are considered. We don't take into account all bus lines in Shenzhen because some lines are with noisy or missing GPS data. In the figure, we show that with 20, 40, 60, 80, 100 percent of the total 836 lines covered, BM-se-VTSPC algorithm reduces about 600 to 2,000 buses compared to the true value and about

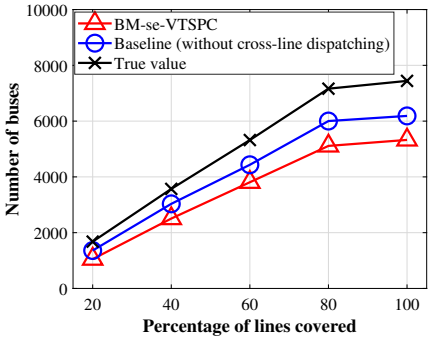


Figure 14: BM-se-VTSPC with city-scale 836 bus lines.

300 to 1,000 buses compared to the scheme without cross line dispatching. Although the true value considers a small amount of cross-line dispatching already, it is not optimized in scheduling a single bus line, which means there are some redundant buses. However, our baseline is optimized for each single bus line separately without cross-line dispatching. Therefore, the true value is worse than our baseline result.

5.2.2 Experiments for Evaluating LPF-VTSPC with Two Common Bus Line Topologies

(a) Topology 1. (b) Topology 2.

Figure 15: An illustration of the two common topologies with which we evaluate LPF-VTSPC. Topology 1 in Figure 15(a) has two lines l_3 and l_4 that share the same origin and terminal stations. Topology 2 in Figure 15(b) has three lines l_5, l_6, and l_7 that form a triangle.

In our experiments, we evaluate the LPF-VTSPC algorithm with two common bus line topologies in Shenzhen's bus system, shown in Figure 15, where Figure 15(a) shows topology 1 with bus lines l_3 and l_4 that share the same origin and terminal stations, and Figure 15(b) shows topology 2 with three lines l_5, l_6, and l_7 that form a triangle. We set the timetables for different lines to be different yet fixed in our experiments, and let the electricity consumption and trip duration to be the same for dispatching tasks of the same line. Detailed parameter settings for lines l_3-l_7 are in Table 2. The baseline strategy is applying LPF-VTSPC to each line separately and sum all the results up, i.e., without considering cross-line disaptching.

Figure 16 shows that, as line l_4's electricity consumption varies, LPF-VTSPC ensures that the number of buses needed to serve lines l_3 and l_4 remains relatively small, and almost stops increasing after line l_4's electricity consumption reaches 15%, whereas more e-buses are needed, if we use the baseline algorithm that does not consider cross-line dispatching. Figures 18 and 17 demonstrate similar

Line	Task No.	Con.	Tr.	Int.	St.
l_3	40	60%	120	5	7:00
l_5	40	60%	120	5	7:00
l_6	30	35%	70	15	6:35
l_4 (Set. 1)	30	[10%, 40%]	70	15	6:35
l_4 (Set. 2)	30	35%	[40, 70]	15	6:35
l_4 (Set. 3)	30	35%	70	[15, 40]	6:35
l_7 (Set. 1)	20	[5%,65%]	40	5	6:15
l_7 (Set. 2)	20	20%	[30, 80]	5	6:15
l_7 (Set. 3)	20	20%	40	[3,12]	6:15
l_8	40	50%	120	5	7:00
l_9	30	35%	70	15	6:35
l_{10} (Set. 1)	20	[10%,40%]	40	5	6:15
l_{10} (Set. 2)	[20,45]	20%	40	5	6:15
l_{10} (Set. 3)	20	20%	40	[3,15]	6:15

Table 2: Detailed parameter settings for lines l_3-l_{10}, where Con. refers to the electricity consumption of each trip, Tr. refers to the trip time (min), Int. refers to the dispatching interval (min), St. refers to the starting time of each line, and Set. refers to Setting.

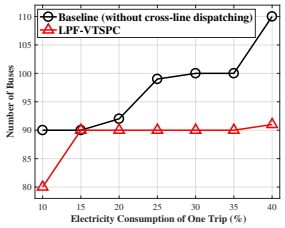


Figure 16: Vary l_4's electricity consumption under Topo 1.

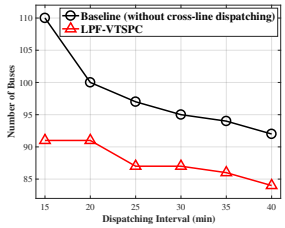


Figure 17: Vary l_4's dispatching interval under Topo 1.

patterns as Figure 16, in which we vary l_4's trip duration and dispatching interval, respectively.

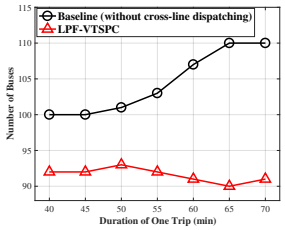


Figure 18: Varying line l_4's trip duration under Topo 1.

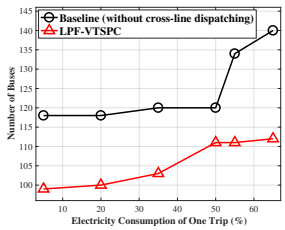


Figure 19: Vary l_7's electricity consumption under Topo 2.

Figures 19-21 shows the results of evaluating LPF-VTSPC under topology 2 where lines l_5-l_7 form a triangle. Figures 19-21, respectively, show that LPF-VTSPC requires much less number of e-buses to serve the three lines compared with the baseline algorithm without cross-line dispatching, if we vary line l_7's electricity consumption, trip duration, and dispatching interval.

5.2.3 Experiments for Evaluating LPF-VTSPC with Different Charging Facility Placements

In this set of experiments, we evaluate the effect of different charging facility placements on LPF-VTSPC's performance. Such experiments are meaningful in practice as the high cost of constructing and maintaining e-bus charging facilities makes it infeasible to build them at all bus stations. In

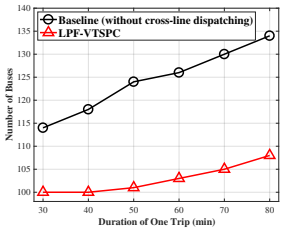


Figure 20: Varying line l_7's trip duration under Topo 2.

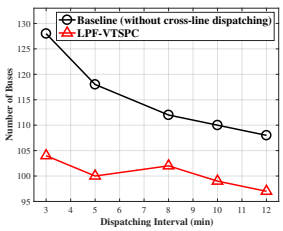


Figure 21: Vary l_7's dispatching interval under Topo 2.

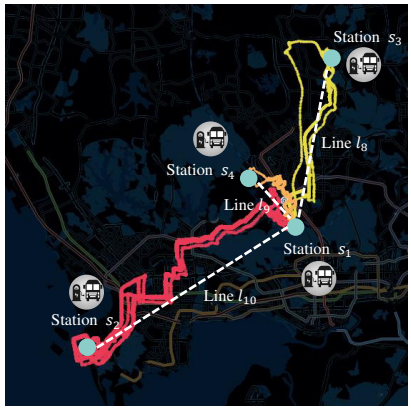
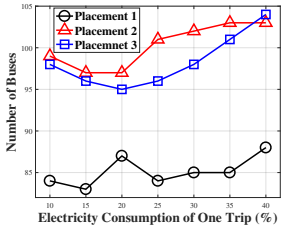
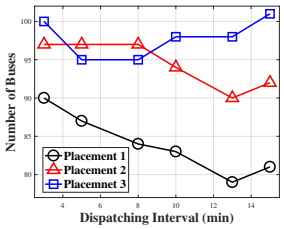


Figure 22: An illustration of the possible locations of charging facilities.

our experiments, we consider three bus lines l_8, l_9, and l_{10} sharing station s_1. The stations s_1-s_4 are the possible locations of charing facilities shown in Figure 22. Specifically, we place charging facilities at all stations s_1-s_4 (placement 1), at suburban stations s_2-s_4 (placement 2), and at the downtown station s_1 (placement 3), respectively. The detailed parameter settings of lines l_8-l_{10} are in Table 2.

From Figure 23, we can observe that no matter how the parameters change, the number of buses needed when all stations s_1-s_4 are equipped with charging facilities are always less than that under the other two placements. It shows that the increase of the number of charging facilities can reduce the number of buses needed. Furthermore, Figure 23 shows that it is not obvious to decide which one of placements 2 and 3 outperforms the other, and for each of the three placements, the number of buses needed does not change significantly, even if the parameters vary sharply.





(a) Vary l_{10}'s electricity consump- tion. (b) Vary l_{10}'s dispatching interval.

Figure 23: LPF-VTSPC with the charging facility placements 1, 2, and 3.

5.2.4 Experiments for Evaluating LPF-VTSPC with City-Scale Dataset

In this set of experiments, we choose three busy stations s_5, s_6, and s_7 with GPS coordinates (114.058, 22.537), (114.016, 22.529), and (114.084, 22.551), respectively, in Shenzhen, and consider the 23 lines that share station s_5, and the 67 lines that share all three stations s_5-s_7. We compare LPF-VTSPC with ground truths which represent the number of e-buses actually utilized by those lines in our dataset.

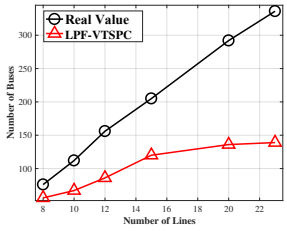


Figure 24: LPF-VTSPC with multiple lines that share the station s_5.

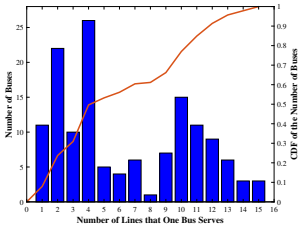


Figure 25: PDF and CDF of the number of lines each bus serves (23 lines).

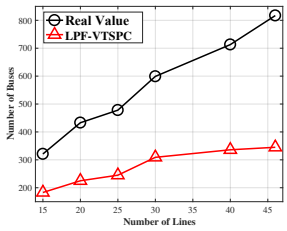


Figure 26: Evaluation of LPF-VTSPC with the 47 lines that share stations s_5 and s_6.

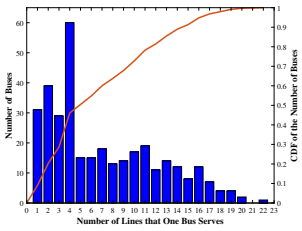


Figure 27: PDF and CDF of the number of lines each bus serves, when we optimize all 47 lines.

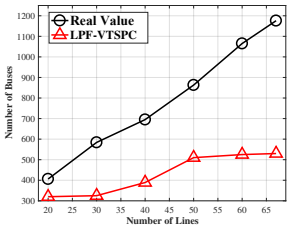


Figure 28: LPF-VTSPC with multiple lines that share the stations s_5 -s_7.

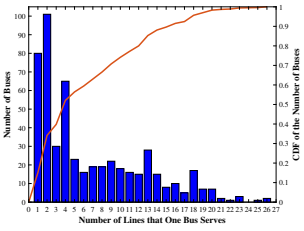


Figure 29: PDF and CDF of the number of lines each bus serves (67 lines).

From Figure 24, we can observe that the ground truth value increases almost linearly with the number of lines, while the number of e-buses required by LPF-VTSPC grows at a much slower rate. Figure 24 shows that, when we jointly optimize 23 lines, only 41.3% of the ground truth value is need to complete all dispatching tasks. Figure 25 shows the number of lines every e-bus serves and the CDF when we jointly optimize 23 lines. It shows that 92.1% e-buses serve multiple lines, and 50.4% serve over 5 lines.

Figure 26 and 28 show similar trends as Figure 24, and 42.2% and 45.1% of the ground truths are need by the LPF-VTSPC algorithm when we jointly optimize 47 lines (Figure 26) and 67 lines (Figure 28), respectively. In Figure 27, we

observe that when jointly optimizing 47 lines, 9.0% e-buses serve only one line and 53.9% of them serve more than 5 lines, whereas, if we consider 67 lines jointly, 15.0% e-buses serve only one line and 48.0% of them serve more than 5 lines.

Next, we further scale up the number of lines considered in our experiments, and randomly choose 20, 50, 100, 200, and 300 lines from Shenzhen's bus systems. When the number of lines is relatively small, the lines have little chance to share the same origin and terminal stations because of the randomness in choosing the lines in our experiments, and thus we can observe limited difference between the results of LPF-VTSPC and the ground truths for small line numbers in Figure 30. With the increase of the scale, more lines have the chances to share one bus, and the advance of LPF-VTSPC over the ground truths becomes increasingly obvious. Based on Figure 30, when we jointly consider 300 lines, LPF-VTSPC uses 61.8% of the ground truth to satisfy all dispatching tasks.

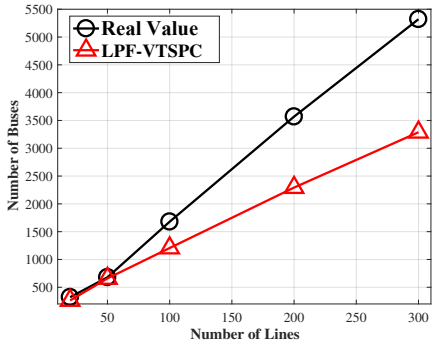


Figure 30: LPF-VTSPC with multiple (up to 300) randomly chosen lines.

6 Related Work

In the past few years, optimization for the urban transportation system [14–17] with EVs has always been a voguish research topic. Firstly, in order to establish an efficient EV network, fundamental researches have been done, such as the operating patterns of a typical EV network [18] and the electricity power allocation in the grid to supply the charging stations [19]. [17] proposed to predict demand for EV sharing systems with data-driven method. Secondly, based on these studies, advanced researches have been done on managing and settling charging stations. From the charging service providers' point of view, researches [20] have been carried out on developing a win-win scheme of pricing charging services, which can improve revenue and promote satisfaction. Furthermore, charging station coordination under different constraints are also widely studied [21]. In this paper, we optimize the dispatching scheme for bus service providers, while the above papers focus on optimizing for charging stations.

Thirdly, scheduling and routing problems for EVs, particularly in public transportation systems, have aroused attention. There are many approaches to solving the **general vehicle routing problem** [22]. [22] proposes novel machine learning framework to find the optimal path. Moreover, **novel routing methods for EVs and buses** have also been introduced and buses [23–25], where [23] tries to minimize passenger waiting time in the routing process, [24] is a data-driven method with multiple objectives, and [25] aims at planning the best path queries from a language processing perspective. However, some route planning algorithms [23, 24] are routing algorithms for a single vehicle, while we solve the bus dispatching problem in fleet scale. The others [25] are routing methods for regular vehicles, and thus do not consider charging for EVs, while our problem consists of both routing and charging scheduling, which makes the general routing algorithms inapplicable. Furthermore, a great number of papers [26–37] have studied **charging scheduling and battery management of EVs** in various situations, mainly considering when and where to charge in order to obtain less waiting time and cost. For example, [32] proposes a reputation-based fair power allocation for EVs in the smart grid. [33] considers ADMM-Based decentralized charging problem with trip duration limits. [28] balances cost and dissatisfaction in online EV Charging under real-time pricing. The previous researches on scheduling for EVs put attention on reducing operation cost or waiting time for charging by scheduling, with the conditions are set on a fixed number of buses, while our paper is the first study that aims to reduce cost by minimizing the number of buses. Our method would be more applicable for those who are building a new bus system.

[1, 38–42] study the scheduling algorithm for urban e-buses. [1] applies Markov Decision Process to determine whether and how long a bus should charge at a station, but they do not address the bus dispatching problem, while we combine bus dispatching with charging scheduling, and aim at minimizing the bus fleet. [38] focuses on a wirelessly charged bus system and proposes an optimal charging scheme that can minimize the charging cost for such a system. [39] creates a data-driven method to provide routing instructions from station to stations, while we optimize the bus system by determining which lines a bus should serve. [40] mines through bus transportation data to extract the mobility demand for passengers. [41] considers jointly the e-bus routing and the charing infrastructure under partial charging of the batteries. [42] considers a battery-lifetime-aware scheduling for e-bus fleet. The system can be applied to situations that require passenger mobility data, such as traffic anomaly detection, but we suppose that the timetable for buses has been pre-set to satisfy the passenger's mobility needs.

Besides, graph theory has attracted many attentions on dispatching in urban mobility systems [43, 44]. [43] considers the taxi-pooling problem, where they also named their graph as shareability networks. Although we have the same name, our work is significantly different from theirs in the following aspects. The goal of [43] is to balance the trade-off between collective benefits of sharing and degradation of the passengers' comfort. However, we try to minimize the total number of e-buses, which is the first obvious difference. Also, instead of only considering the characteristics of dispatching tasks, in our work we have to also consider the status of our e-buses. Furthermore, the difficulty of the two problems is different. Problem formulated in [43] can be well solved in polynomial time. Our problem, in contrast, is NP-hard. These points make our work very different with

theirs. The minimum fleet problem for urban ridesharing-free mobility-on-demand systems has already been studied by a pioneering work in Nature [44]. E-buses systems are quite different with mobility-on-demand systems, where the routings can be personalized and the covering limits can be ignored for diesel vehicles.

To summarize, previous works on charging scheduling mainly focuses on reducing operation cost or waiting time, and the works on routing either do not consider charging or are only designed for single EVs. Different from all the aforementioned prior literatures, this paper addresses a brand new problem of minimizing the number of e-buses with a novel cross-line dispatching algorithm that takes charging scheduling into consideration.

7 Conclusion

In this paper, we propose to systematically exploit at city-scale cross-line dispatching, a novel smart dispatching strategy that allows one bus to serve multiple bus lines, in order to minimize the number of e-buses needed to satisfy the public transportation demands in urban e-bus systems. We adopt novel graph-theoretic methods to construct a generalizable model capturing various real-world factors, such as, the spatio-temporal relationships among bus trips, e-buses' recharging actions, and many others. Theoretically, we formulate the cross-line dispatching optimization problem as a combinatorial optimization over the constructed graph. Moreover, we prove that such problem is NP-hard with no $(2-\epsilon)$-approximation algorithms. We next propose a polynomial-time algorithm that solves the problem with a guaranteed $\frac{\gamma+1}{2}\left(1+\ln\frac{2\gamma}{\gamma+1}\right)$ approximation ratio. Experimentally, we conduct extensive experiments on a large-scale real-world e-bus dataset from Shenzhen containing the trajectories of 16,359 buses within the time span from June 1st to June 30th, 2017, which validates the effectiveness of our algorithms.

In the future, we will focus on the robustness of our system and algorithm. For example, how to deal with e-bus breaking down halfway and temporary route adjustments are still open questions. Besides, battery aging and renewal is also an important factor worth considering, especially in the long term.

Acknowledgment

This work was supported in part by National Key R&D Program of China 2018AAA0101200, in part by NSFC China (No. 61902244, U20A20181, 62061146002, 61829201, 61832013, 61960206002, 42050105), in part by Shanghai Municipal Science and Technology Commission Grant (19YF1424600, 18XD1401800), in part by Tencent AI Lab Rhino-Bird Focused Research Program JR202034, and in part by Shanghai Key Laboratory of Scalable Computing and Systems.

References

[1] G. Wang, X. Xie, F. Zhang, Y. Liu, and D. Zhang, "bcharge: Data-driven real-time charging scheduling for large-scale electric bus fleets," in *IEEE Real-Time Systems Symposium (RTSS)*, 2018.

[2] V. V. Vazirani, "Approximation algorithms." Springer-Verlag Berlin Heidelberg, 2003.

[3] Z. He, C. Chow, and J. Zhang, "Stann: A spatio-temporal attentive neural network for traffic prediction," *IEEE Access*, vol. 7, pp. 4795–4806, 2019.

[4] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in *AAAI Conference on Artificial Intelligence*, 2020.

[5] W.-C. Lee, W. Si, L.-J. Chen, and M. C. Chen, "Http: a new framework for bus travel time prediction based on historical trajectories," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2012.

[6] A. AbdelAziz, A. Shoukry, W. Gomaa, and M. Youssef, "Trans-sense: Real time transportation schedule estimation using smart phones," in *IEEE International Conference on Sensing, Communication and Networking (SECON)*, 2019.

[7] S. Moran, I. Newman, and Y. Wolfstahl, "Approximation algorithms for covering a graph by vertex-disjoint paths of maximum total weight," *Networks*, vol. 20, no. 1, pp. 55–64, 2010.

[8] "The parameters for klq6129gq1," http://www.higer.com/bus/1159.html, 2017.

[9] F. T. Boesch and J. F. Gimpel, "Covering points of a digraph with point-disjoint paths and its application to code optimization," in *J. ACM*, 1977.

[10] J. A. Bondy, "Graph theory with applications." Elsevier Science Ltd, June 1976.

[11] J. Munkres, "Algorithms for the assignment and transportation problems," in *Journal of the Society for Industrial and Applied Mathematics*, 1957.

[12] J. Hopcroft and R. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," in *SIAM Journal on Computing*, 1973.

[13] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," 1978.

[14] J. Ye, L. Sun, B. Du, Y. Fu, X. Tong, and H. Xiong, "Co-prediction of multiple transportation demands based on deep spatio-temporal neural network," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019.

[15] M. Chiang, E. Lim, W. Lee, and A. T. Kwee, "Btci: A new framework for identifying congestion cascades using bus trajectory data," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017.

[16] M. Yang, Y. Li, X. Zhou, H. Lu, Z. Tian, and J. Luo, "Inferring passengers interactive choices on public transits via ma-al: Multi-agent apprenticeship learning," in *Proceedings of The Web Conference*, 2020.

[17] M. Luo, B. Du, K. Klemmer, H. Zhu, H. Ferhatosmanoglu, and H. Wen, "D3p: Data-driven demand prediction for fast expanding electric vehicle sharing systems," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2020.

[18] G. Wang, X. Chen, F. Zhang, Y. Wang, and D. Zhang, "Experience: Understanding long-term evolving patterns of shared electric vehicle networks," in *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[19] F. Kong, X. Liu, and I. Lee, "Joint rate control and demand balancing for electric vehicle charging," in *IEEE International Conference on Internet-of-Things Design and Implementation (IOTDI)*, 2018.

[20] J. Mrkos, A. Komenda, and M. Jakob, "Revenue maximization for electric vehicle charging service providers using sequential dynamic pricing," in *International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2018.

[21] B. Du, Y. Tong, Z. Zhou, Q. Tao, and W. Zhou, "Demand-aware charger planning for electric vehicle sharing," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2018.

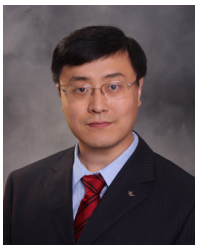
[22] S. A. Pedersen, B. Yang, and C. S. Jensen, "A hybrid learning approach to stochastic routing," in *IEEE International Conference on Data Engineering (ICDE)*, 2020.
[23] J. C. Gareau, E. Beaudry, and V. Makarenkov, "An efficient electric vehicle path-planner that considers the waiting time," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019.
[24] A. Sarker, H. Shen, and J. A. Stankovic, "Morp: Data-driven multi-objective route planning and optimization for electric vehicles," in *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2018.
[25] A. Haryanto, M. Islam, D. Taniar, and M. Cheema, "Ig-tree: an efficient spatial keyword index for planning best path queries on road networks," *Proceedings of The Web Conference*, 2019.
[26] H. Dai, Y. Liu, G. Chen, X. Wu, T. He, A. X. Liu, and Y. Zhao, "Scape: Safe charging with adjustable power," in *IEEE/ACM Transactions on Networking*, 2018.
[27] H. Dai, Y. Liu, G. Chen, X. Wu, T. He, A. X. Liu, and H. Ma, "Safe charging for wireless power transfer," in *IEEE/ACM Transactions on Networking*, 2017.
[28] H. Yi, Q. Lin, and M. Chen, "Balancing cost and dissatisfaction in online ev charging under real-time pricing," in *International Conference on Computer Communications (INFOCOM)*, 2019.
[29] H. Dai, H. Ma, A. X. Liu, and G. Chen, "Radiation constrained scheduling of wireless charging tasks," in *IEEE/ACM Transactions on Networking*, 2018.
[30] L. Yan, H. Shen, Z. Li, A. Sarker, J. A. Stankovic, C. Qiu, J. Zhao, and C. Xu, "Employing opportunistic charging for electric taxicabs to reduce idle time," in *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2018.
[31] Y. Zhang, J. Chen, L. Cai, and J. Pan, "Ev charging network design with transportation and power grid constraints," in *International Conference on Computer Communications (INFOCOM)*, 2018.
[32] A. Al Zishan, M. M. Haji, and O. Ardakanian, "Reputation-based fair power allocation to plug-in electric vehicles in the smart grid," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2020.
[33] G. He, Z. Chai, X. Lu, F. Kong, and B. Sheng, "Admm-based decentralized electric vehicle charging with trip duration limits," in *IEEE Real-Time Systems Symposium, RTSS*, 2019.
[34] P. Zhou, C. Wang, and Y. Yang, "Design and optimization of electric autonomous vehicles with renewable energy source for smart cities," in *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2020.
[35] G. Wang, W. Li, J. Zhang, Y. Ge, Z. Fu, F. Zhang, Y. Wang, and D. Zhang, "sharedcharging: Data-driven shared charging for large-scale heterogeneous electric vehicle fleets," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2019.
[36] Q. Tang, K. Wang, K. Yang, and Y. Luo, "Congestion-balanced and welfare-maximized charging strategies for electric vehicles," *IEEE Transactions on Parallel and Distributed Systems*, 2020.
[37] A. Aveklouris, Y. Nakahira, M. Vlasiou, and B. Zwart, "Electric vehicle charging: a queueing approach," *ACM SIGMETRICS Performance Evaluation Review*, 2017.
[38] C. Yang, W. Lou, J. Yao, and S. Xie, "On charging scheduling optimization for a wirelessly charged electric bus system," in *IEEE Transactions on Intelligent Transportation Systems*, 2018.
[39] X. Fei, O. Gkountouna, D. Pfoser, and A. Züfle, "Spatiotemporal bus route profiling using odometer data," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019.
[40] L. Meegahapola, T. Kandappu, K. Jayarajah, L. Akoglu, S. Xiang, and A. Misra, "Buscope: Fusing individual & aggregated mobility behavior for "live" smart city services," in *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2019.
[41] L. Karzel, "Vehicle scheduling and location planning of the charging infrastructure for electric buses under the consideration of partial charging of vehicle batteries," in *Operations Research*, 2020.
[42] S. Li, S. He, S. Wang, T. He, and J. Chen, "Data-driven battery-lifetime-aware scheduling for electric bus fleets," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 4, pp. 1–22, 2019.
[43] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proceedings of the National Academy of Sciences*, vol. 111, no. 37, pp. 13 290–13 294, 2014.
[44] M. M. Vazifeh, P. Santi, G. Resta, S. H. Strogatz, and C. Ratti, "Addressing the minimum fleet problem in on-demand urban mobility," *Nature*, vol. 557, no. 7706, pp. 534–538, 2018.

Chonghuan Wang received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 2020. He is now a Ph.D. student of Lab for Information and Decision Science (LIDS) and Center for Computational Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include network economics, operations research and management, and online learning.

Yiwen Song received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 2021. He is currently a Ph.D. student of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. His current research mainly falls in the field of wireless sensing, communication and internet-of-things systems.

Guiyun Fan received the B.S. degree in Electronic Information Science and Technology from Xidian University, Xian, China, in 2016, and the Ph.D. degree in Electronic Engineering from Shanghai Jiao Tong University, China, in 2021. She is currently a Post-Doctoral Research Associate with the Department of Computer Science and Engineering in Shanghai Jiao Tong University. Her current research interests include network economics and game theory, crowd and social sensing systems, mobile computing, and reinforcement learning.

Haiming Jin received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 2012, and the Ph.D. degree from the University of Illinois at Urbana-Champaign (UIUC), Urbana, IL, USA, in 2017. He is currently an Associate Professor at the John Hopcroft Center for Computer Science, Shanghai Jiao Tong University. He also worked as a Post-Doctoral Research Associate with the Coordinated Science Laboratory in UIUC from 2017 to 2018. He is broadly interested in addressing unfolding research challenges in the general areas of urban computing, cyber-physical systems, crowd and social sensing systems, network economics and game theory, reinforcement learning, and mobile pervasive and ubiquitous computing.



Lu Su is an associate professor in the School of Electrical and Computer Engineering at Purdue University. His research interests are in the general areas of Internet of Things and Cyber-Physical Systems, with a current focus on wireless, mobile, and crowd sensing systems. He received Ph.D. in Computer Science, and M.S. in Statistics, both from the University of Illinois at Urbana-Champaign, in 2013 and 2012, respectively. He has also worked at IBM T. J. Watson Research Center and National Center for Supercomputing Applications. He has published more than 100 papers in referred journals and conferences, and serves as an associate editor of ACM Transactions on Sensor Networks. He is the recipient of NSF CAREER Award, University at Buffalo Young Investigator Award, ICCPS17 best paper award, and the ICDCS17 best student paper award. He is a member of ACM and IEEE.

Fan Zhang received the Ph.D. degree in communication and information system from the Huazhong University of Science and Technology in 2007. He was a Post-Doctoral Fellow with the University of New Mexico and with the University of Nebraska-Lincoln from 2009 to 2011. He is currently a Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. He is also the Director of the Shenzhen Institute of Beidou Applied Technology (SIBAT). His research topics include intelligent transportation systems, urban computing, and AI technology.

Xinbing Wang received the B.S. degree (with hons.) in automation from Shanghai Jiao Tong University, Shanghai, China, in 1998, the M.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 2001, and the Ph.D. degree with a major in electrical and computer engineering and minor in mathematics from North Carolina State University, Raleigh, in 2006. Currently, he is a professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. His research interests include resource allocation and management in mobile and wireless networks, TCP asymptotics analysis, wireless capacity, crosslayer call admission control, asymptotics analysis of hybrid systems, and congestion control over wireless ad hoc and sensor networks. Dr. Wang has been a member of the Technical Program Committees of several conferences including ACM MobiCom 2012, ACM MobiHoc 2012, IEEE INFOCOM 2009-2013.