Energy-Efficient Timely Truck Transportation for Geographically-Dispersed Tasks

Qingyu Liu
Dept. of ECE, Virginia Tech

Haibo Zeng
Dept. of ECE, Virginia Tech

Minghua Chen
Dept. of IE, CUHK

ABSTRACT

We consider the scenario where a long-haul heavy-duty truck drives across a national highway system to fulfill multiple geographically-dispersed tasks in a specific order. The objective is to minimize the total fuel consumption subject to the pickup and delivery time window constraints of individual tasks, by jointly optimizing task execution times, path planning, and speed planning. The need to coordinate execution times for multiple tasks differentiates our study from existing single-task ones. In this paper, we first prove that the problem is NP-hard, and argue that optimizing task execution times by itself is a non-convex puzzle. We then exploit the problem structure to develop (i) a Fully-Polynomial-Time Approximation Scheme (FPTAS) for solving small-/medium- scale problem instances, and (ii) a fast and efficient heuristic, called SPEED (Subgradient-based Price-driven Energy-Efficient Delivery), for solving large-scale problem instances. We provide performance guarantees of both algorithms. We further characterize a sufficient condition under which SPEED generates an optimal solution. We evaluate the performance of our solutions using real-world traces over the US national highway system. We observe that our solutions can save up to 22% fuel as compared to fastest-/shortest- path baselines, and up to 10% fuel than a conceivable alternative that directly generalizes the state-of-the-art single-task algorithm to the multi-task settings studied in this paper. The fuel saving is robust to the number of tasks to be fulfilled. A set of simulations also show that our algorithms always obtain close-to-optimal solutions and meet all time window constraints for all the theoretically feasible instances, while the conceivable alternative fails to meet one or more time window constraints for 47% of the instances.

CCS CONCEPTS

• **Mathematics of computing → Paths and connectivity problems**; • **Applied computing → Transportation**;

KEYWORDS

Energy-efficient transportation, timely delivery, multiple tasks, task execution times optimization, path planning, speed planning

A major part of this work was done during Qingyu's visit to the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, in 2017.

e-Energy '18, June 12–15, 2018, Karlsruhe, Germany

ACM ISBN 978-1-4503-5767-8/18/06…$15.00
https://doi.org/10.1145/3208903.3208911

ACM Reference Format:
Qingyu Liu, Haibo Zeng, and Minghua Chen. 2018. Energy-Efficient Timely Truck Transportation for Geographically-Dispersed Tasks. In *e-Energy '18: The Ninth International Conference on Future Energy Systems, June 12–15, 2018, Karlsruhe, Germany*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3208903.3208911

1 INTRODUCTION

In 2016, the US trucking industry hauls 70.6% (up to 10.42 billion tons) of all freight tonnage [1]. It collects $676 billion in gross freight revenues, accounting for 79.8% of the freight bill [1]. This number would rank 19 worldwide if measured against the GDP of countries. Meanwhile, with only 4% of total vehicle population, heavy-duty trucks consume 18% of energy in the whole transportation sector including all vehicles, airplanes, pipelines, and railways [9]. This alerting observation, together with that fuel consumption accounts for the largest truck operating cost factor (~26%) [9], makes it critical to reduce fuel consumption for cost-effective and environment-friendly heavy-duty truck operation.

We consider a common truck operation scenario where a heavy-duty truck drives across a national highway system to fulfill multiple geographically-dispersed tasks in a specific order. Our objective is to minimize the total fuel consumption subject to the pickup and delivery time window constraints of individual tasks, by jointly optimizing task execution times, path planning, and speed planning. We remark that pickup and delivery time window constraints are common in truck transportation. For instance, mobile applications like uShip [27] and Uber Freight [24] provide lots of freight transportation requests for the truck operators, which are often associated with earliest and latest pickup/delivery time requirements.

Optimizing task execution times for individual tasks is critical for saving fuel and delivering freight in a timely manner, and it differentiates our study from existing works under the single task setting, e.g., [4]. For individual tasks, allocating longer execution times, or equivalently setting a larger deadline for the task fulfilling time, can allow more design space for path planning and speed planning to save fuel. According to the studies in [4] based on real-world traces, on average, one can save 1.7% fuel for heavy-duty trucks if the task fulfilling time budget is increased by 3%. Meanwhile, it is also crucial to catch the pick-up and delivery time window of individual tasks, to avoid substantial penalty due to deadline violation. For example, for the Tampa Bay Urbanized Area, the Texas Transportation Institute estimates a 2011 annual truck delay of over 3 million hours at a cost of $246 million [21].

Given the execution times of individual tasks, the driver can perform path planning and speed planning to optimize the fuel consumption. Optimizing the path planning can save a significant amount of fuel. Due to the difference in distances and road conditions such as the grade, the fuel consumption can be drastically

Table 1: Comparison of our work and existing energy-efficient timely truck transportation studies.

		RSP [7, 10, 16]	PASO [4]	PDPTW [13, 19, 20]	Our work
Design Spaces	Path planning	✓	✓	✓	✓
	Speed planning	✗	✓	✗	✓
	Task execution times optimization	✗	✗	✓	✓
Results	Algorithms	FPTAS [7, 16], Heuristic [10]	FPTAS [4], Heuristic* [4]	Heuristic [13, 19, 20]	FPTAS, Heuristic*

$\star$: the heuristic algorithm is guaranteed to solve associated problem optimally when certain conditions are met

different when traveling along different paths from the same source to the same destination (e.g., 21% difference in the fuel consumption as shown in the real-world studies [23]). Optimizing the speed planning is also important for saving fuel. For vehicles including heavy trucks, normally there is a most fuel-efficient speed for a given road condition. This efficient speed is around 55 miles per hour (mph) for many trucks [25], and the fuel economy will degrade if traveling below or above the efficient speed. For example, it is observed that every one mph increase in the speed above 55 mph decreases the fuel economy by 0.1 miles per gallon [2, 6]. Hence, it saves fuel to travel at a speed close to the most efficient one.

We remark that the feasibility of task execution times depends on path planning and speed planning, and the fuel consumption achievable by path planning and speed planning depends on the allocated task execution times. It is thus non-trivial to save fuel while meeting time window constraints for multiple individual tasks, due to the unique challenge introduced by the coupling among task execution time, path planning, and speed planning.

Existing studies: Tab. 1 summarizes the comparison between our work and the most related ones. All the works are NP-hard and target at minimizing fuel consumption subject to various deadline constraints.

Restricted Shortest Path (RSP): RSP requires to find a path from a source to a destination, such that the path cost is minimized and the path travel time is upper bounded by a deadline constraint. RSP is NP-hard [7] with a heuristic algorithm proposed by [10] and an FPTAS developed by [7, 16]. However, RSP optimizes the fuel consumption with only path planning involved where the speed is fixed. Moreover, RSP considers only a single task, and cannot be directly applied to the scenario of multiple tasks.

PAth selection and Speed Optimization (PASO): PASO [4] generalizes RSP with speed planning taken into account. The authors in [4] develop an FPTAS and a heuristic algorithm to solve PASO. The heuristic is guaranteed to obtain the optimal fuel consumption when certain conditions are met. However, similar to RSP, PASO considers only a single task. For our problem of multiple tasks, approaches directly generalizing RSP or PASO involves solving a non-convex task execution time optimization puzzle, which is uniquely challenging.

Pickup and Delivery Problem with Time Window (PDPTW): as a generalization of Traveling Salesman Problem (TSP) [14] and Vehicle Routing Problem (VRP) [12], PDPTW finds a set of paths such that each customer with known timely pickup or delivery demand can be served exactly once. PDPTW is challenging to solve. TSP, a special case of PDPTW, is APX-hard [18] and no PTAS exists for solving TSP (and consequently PDPTW) unless P = NP. As compared to PDPTW, our work considers all the three design spaces in task execution times, path planning, and speed planning, while PDPTW only involves the first two. Moreover, PDPTW does not fix the execution order of the tasks, which is a more general setting than the fixed order setting considered in our paper, making it more challenging to solve than ours. In the literature, only heuristic algorithms are available for solving small-/medium- scale PDPTW [13, 19, 20] and it is impossible to develop PTAS for it [18]. In contrast, for our problem, we develop an FTPAS and a heuristic algorithm both with performance guarantees. These results reveal the fundamental difference of the two problems.

Contributions. Our work differentiates from all existing studies in that we consider the fuel consumption minimization problem under a multiple tasks setting, subject to the pickup and delivery time window constraints of individual tasks. Solving this problem requires us to jointly optimize the task execution times, path planning, and speed planning, making our problem uniquely challenging. We make the following contributions in this paper.

▷ We prove our problem is NP-hard, and even optimizing the task execution times by itself is a non-convex puzzle.

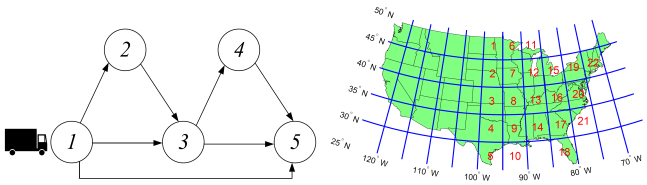
▷ We design an FPTAS with a $(1+\epsilon)$-approximation ratio guaranteed even in the worst theoretical problem instance, in a polynomial time of $O(mK^3n^3/\epsilon^2)$ where n (resp. m) is the number of nodes (resp. edges) and K is the number of tasks. Practically, our FPTAS is suitable for solving small-/medium- scale problem instances.

▷ To obtain high-quality solutions without incurring high complexity for large-scale problem instances, we develop a fast heuristic algorithm, called SPEED (Sub-gradient-based Price-driven Energy-Efficient Delivery), based on dual sub-gradient updates of task execution times. We provide performance guarantees for SPEED, and also characterize a condition for it to generate an optimal solution to the problem.

▷ We evaluate the performance of our solutions using real-world traces over the US national highway system. We observe that our solutions achieve close-to-optimal performance, saving up to 22% fuel as compared to the fastest-/shortest- path baselines, and up to 10% fuel than a conceivable alternative that directly generalizes the state-of-the-art single-task algorithm to the multi-task settings studied in this paper. The fuel saving is robust to the number of tasks to be fulfilled. A set of simulations also show that our algorithms always obtain close-to-optimal solutions and meet the pickup/delivery time window constraints for all the theoretically feasible instances, while the conceivable alternative fails to meet one or several time window constraints for 47% of the instances.

2 MODEL AND PROBLEM FORMULATION

As illustrated in Fig. 1(a), we study the scenario of one heavy-duty truck fulfilling multiple tasks in a specific order across a national



(a) System model: a simple example of a truck driving in a highway network.

(b) USA highway network partitioned into 22 regions used in our simulations.

Figure 1: A truck timely fulfills multiple geo-dispersed tasks in a national highway system to optimize fuel consumption.

highway system. Each task requires to transport given cargos from a source to a destination satisfying both a source pickup time window and a destination delivery time window. The objective is to minimize total incurred fuel consumption. Key notations are summarized in Tab. 5, Appendix 7.1.

2.1 System Model

We model a national highway network as a directed graph $G \triangleq (V, E)$ where an edge $e \in E$ represents a road segment, and a node $v \in V$ represents an intersection or a connecting point of two adjacent road segments. Here a road segment is assumed to have homogeneous grade and environmental conditions like the surface resistance. We define $n \triangleq |V|$ and $m \triangleq |E|$. For each $e \in E$, we denote $D_e > 0$ as its positive distance, $r_e^l > 0$ (resp. $r_e^u \geq r_e^l$) as its positive minimum (resp. maximum) traveling speed, and $c_e^l > 0$ (resp. $c_e^u \geq c_e^l$) as its positive minimum (resp. maximum) fuel consumption.

We consider an eco-routing problem for one heavy-duty truck across G to fulfill K transportation tasks denoted by $\vec{\tau} = \{\tau_i : i = 1, 2, ..., K\}$. In this paper we have the following two practically reasonable assumptions for the truck to fulfill the given K tasks $\vec{\tau}$:

(1) the truck must fulfill tasks in a specific given order, namely the i^{th} task τ_i must be fulfilled before the j^{th} task τ_j for any $1 \leq i < j \leq K$, and
(2) the truck cannot transport cargoes belonging to different tasks simultaneously.

In this paper, each task τ_i is characterized by a $\rho_i \geq 0$ which is the weight of its cargoes, a source $s_i \in V$ where the truck can pick up cargoes, a destination $d_i \in V$ to which cargoes are delivered, a pickup window $s\omega_i$ defining the allowed time for picking up at s_i, and a delivery window $d\omega_i$ representing the allowed time for delivering at d_i. We use $\alpha(\cdot)$ (resp. $\beta(\cdot)$) to denote the starting time (resp. ending time) of a window, namely

$$s\omega_i \triangleq [\alpha(s\omega_i), \beta(s\omega_i)], \beta(s\omega_i) \geq \alpha(s\omega_i) \geq 0,$$
$$d\omega_i \triangleq [\alpha(d\omega_i), \beta(d\omega_i)], \beta(d\omega_i) \geq \alpha(d\omega_i) \geq 0.$$

When the truck is expected to pick up τ_i at source s_i, earliest arrival before $\alpha(s\omega_i)$ is possible. But the truck cannot leave s_i until $\alpha(s\omega_i)$ when allowed to pick up cargoes. Arrival later than $\beta(s\omega_i)$ is not permitted since the latest pickup time is missed. The same assumption holds for the delivery window $d\omega_i$ when the truck is expected to deliver τ_i at destination d_i.

In this paper, we use "deadline" as a constraint imposed on the maximum travel time to fulfill a task, while "time window" defines a specific earliest and latest pickup/delivery time at a certain node.

The truck fuel consumption depends on many factors [3]. Similar to [4], as we consider a specific heavy-duty truck driving over a specific highway, any environment-specific parameter such as the gradient is road-segment-dependent and assumed to be fixed for each specific road segment. Hence given a road segment, the fuel consumption rate is assumed reasonably to be a function of the vehicle load, namely the cargo mass of a task, and the vehicle speed. Same to [4], we neglect the acceleration/deceleration both when driving inside any road segment (Jensen's inequality implies driving at a constant speed is most fuel-economic inside a road segment [4, Lem. 1]) and when switching between road segments (negligible compared with the length of real-world road segment). Both load and speed are key factors in any vehicle fuel consumption models (see a survey in [3]). According to [3], as estimated by certain model, the fuel consumption rate can increase by 103% when speed is increased from 50km/h to 70km/h, and can increase by 27% when the truck load factor is increased from 0% to 30%.

We define $f_e(r_e, \rho) : \left[r_e^l, r_e^u\right] \to \mathbb{R}^+$ as the positive *fuel consumption rate* for the truck to pass $e \in E$ following a constant speed r_e with load ρ, which is a function of r_e given a ρ. Same to the assumption in [4] verified by both physical laws and comprehensive simulations using real-world data, we assume $f_e(r_e, \rho)$ to be *strictly convex* over $\left[r_e^l, r_e^u\right]$ with r_e given a truck load ρ.

Since each $e \in E$ has a fixed distance D_e, with the fuel rate $f_e(r_e, \rho)$, we can define a positive fuel consumption function $c_e(t_e, \rho)$

$$c_e(t_e, \rho) \triangleq t_e \cdot f_e\left(\frac{D_e}{t_e}, \rho\right), \tag{1}$$

which is the total fuel consumption for the truck to pass e with load ρ and travel time t_e. Due to the strict convexity of $f_e(\cdot)$, following the proof of [4, Lem. 2], it can be proved: (i) $c_e(t_e, \rho)$ is strictly convex over $t_e \in \left[t_e^l, t_e^u\right]$ with t_e given a truck load ρ, where $t_e^l \triangleq D_e / r_e^u$ is the minimum travel time and $t_e^u \triangleq D_e / r_e^l$ is the maximum travel time, and (ii) there exists a travel time $\hat{t}_e(\rho) \in \left[t_e^l, t_e^u\right]$ such that $c_e(t_e, \rho)$ is first strictly decreasing over $\left[t_e^l, \hat{t}_e(\rho)\right]$ and then strictly increasing over $\left[\hat{t}_e(\rho), t_e^u\right]$, given a specific load ρ. Hence, to fulfill a specific τ_i, for any $e \in E$, the possible travel time in the optimal solution of our problem must belong to the range of $\left[t_e^l, \hat{t}_e(\rho_i)\right]$, straightforwardly. Without loss of generality, we assume $c_e\left(\hat{t}_e(\rho_i), \rho_i\right) \geq c_e^l$ and $c_e\left(t_e^l, \rho_i\right) \leq c_e^u$, since otherwise we still can figure out the possible travel time range which is a subset of $\left[t_e^l, \hat{t}_e(\rho_i)\right]$ in polynomial time, considering the strictly decreasing property of $c_e(t_e, \rho_i)$ over $\left[t_e^l, \hat{t}_e(\rho_i)\right]$.

2.2 Problem Formulation

Considering any two consecutive tasks τ_i and τ_{i+1} $(1 \leq i \leq K-1)$, without loss of generality, we assume the delivery node of the i^{th} task is same as the pickup node of the $(i+1)^{th}$ task, i.e. $d_i = s_{i+1}$, because otherwise we can always add an additional task τ_{K+i} with $s_{K+i} = d_i$, $d_{K+i} = s_{i+1}$, $s\omega_{K+i} = d\omega_i$, $d\omega_{K+i} = s\omega_{i+1}$, $\rho_{K+i} = 0$ to

the task sequence $\vec{\tau}$ and in the end $|\vec{\tau}| \leq 2K$. Similarly, we assume s_1 to be the source and d_K to be the destination of the whole trip. By above assumptions, the input tasks $\vec{\tau}$ is equivalent to a sequence $\vec{\sigma}$ of $K+1$ nodes for the truck to pass:

$$\vec{\sigma} = \{\sigma_i : \sigma_1 = s_1, \sigma_{K+1} = d_K, \sigma_i = s_i = d_{i-1}, \forall i = 2, 3, ..., K\}.$$

For any node σ_i, because it is the destination of τ_{i-1} and the source of τ_i, it will have two time window constraints, namely the delivery window $d\omega_{i-1}$ and the pickup window $s\omega_i$. Comparing the two time window constraints, the earliest allowed leaving time is in fact $\max\{\alpha(d\omega_{i-1}), \alpha(s\omega_i)\}$, denoted by T_i^{out},

$$T_1^{\text{out}} \triangleq \alpha(s\omega_1), T_i^{\text{out}} \triangleq \max\left\{\alpha(d\omega_{i-1}), \alpha(s\omega_i)\right\}, i = 2, 3, ..., K,$$

which indicates that the truck cannot leave σ_i until he/she can finish both the delivery of task τ_{i-1} and the pickup of task τ_i. Similarly, the latest allowed arrival time, denoted by T_i^{in}, is

$$T_{K+1}^{\text{in}} \triangleq \beta(d\omega_K), T_i^{\text{in}} \triangleq \min\left\{\beta(d\omega_{i-1}), \beta(s\omega_i)\right\}, i = 2, 3, ..., K,$$

which indicates that the truck must arrive at the node σ_i no later than both the latest delivery time of task τ_{i-1} and the latest pickup time of task τ_i. Overall, all time window constraints associated with tasks $\vec{\tau}$ are now converted equivalently to the earliest leaving time constraint and latest arrival time constraint for nodes in $\vec{\sigma}$.

In this paper there are two different kinds of design variables: binary variable x_i^e defining a path from σ_i to σ_{i+1} to fulfill τ_i,

$$x_i^e = \begin{cases} 1, & e \in E \text{ is on the path to fulfill the task } \tau_i; \\ 0, & \text{otherwise,} \end{cases}$$

and non-negative variable t_i^e representing the specific travel time for the truck to pass edge e to fulfill the task τ_i.

By vectoring variables as $\vec{x}_i \triangleq \{x_i^e : e \in E\}$ and $\vec{t}_i \triangleq \{t_i^e : e \in E\}$, our problem Timely tRansportation for Energy-efficient trucKing, denoted by TREK, can be formulated as

$$\min_{\vec{x}_i \in \mathcal{X}_i, \vec{t}_i \in \mathcal{T}_i} \quad \sum_{i=1}^{K} \sum_{e \in E} x_i^e \cdot c_e(t_i^e, \rho_i) \tag{2a}$$

$$\text{s.t.} \quad a_i = \max\left\{a_{i-1}, T_i^{\text{out}}\right\} + \sum_{e \in E} x_i^e t_i^e \leq T_{i+1}^{\text{in}},$$

$$a_0 = 0, \forall i = 1, 2, ..., K, \tag{2b}$$

where set $\mathcal{X}_i$ defines one path from σ_i to σ_{i+1},

$$\mathcal{X}_i \triangleq \Big\{\vec{x}_i : x_i^e \in \{0, 1\}, \forall e \in E, \text{ and}$$
$$\sum_{e \in \text{out}(v)} x_i^e - \sum_{e \in \text{in}(v)} x_i^e = 1_{\{v = \sigma_i\}} - 1_{\{v = \sigma_{i+1}\}}, \forall v \in V\Big\},$$

where $1_{\{\cdot\}}$ is the indicator function, $\text{in}(v) \triangleq \{(u, v) : (u, v) \in E\}$ is the set of incoming edges of node $v \in V$, $\text{out}(v) \triangleq \{(v, u) : (v, u) \in E\}$ is the set of outgoing edges of node $v \in V$. Set $\mathcal{T}_i$ defines the possible road travel time,

$$\mathcal{T}_i \triangleq \left\{\vec{t}_i : t_e^l \leq t_i^e \leq \hat{t}_e(\rho_i), \forall e \in E\right\}.$$

In formulation (2), a_i is the exact arrival time at σ_{i+1}, which should be no later than the latest allowed arrival time T_{i+1}^{in}. The formula $\max\{a_{i-1}, T_i^{\text{out}}\}$ guarantees that the truck cannot leave σ_i immediately if he/she arrives at σ_i before the earliest allowed leaving time T_i^{out}. Objective (2a) minimizes the total fuel consumption of the whole trip fulfilling task sequence $\vec{\tau}$.

Table 2: An example illustrating our problem TREK based on Fig. 1(a), assuming $t_e^l = t_e^u = 1$ and $c_e^l = c_e^u$ for any $e \in E$.

e	$(1,2)$	$(2,3)$	$(1,3)$	$(3,4)$	$(4,5)$	$(3,5)$	$(1,5)$
c_e	1	1	3	1	1	4	8
τ_i	$\tau_1 : 1 \to 3$			$\tau_2 : 3 \to 5$			
$s\omega_i$	$[0, 2]$			$[0, 3]$			
$d\omega_i$	$[0, 2]$			$[0, 3]$			

In the paper we denote a solution to our problem TREK as

$$p = p_1 \cup p_2 \cup ... \cup p_K,$$

which is a path from σ_1 to σ_{K+1} passing all $\sigma_i \in \vec{\sigma}$, with each p_i being a simple path from σ_i to σ_{i+1}, and each edge $e \in p_i$ assigned a specific travel time $t_i^e \in [t_e^l, \hat{t}_e(\rho_i)]$.

An example of TREK is introduced in Tab. 2. The optimal solution that minimizes the total cost to timely fulfill the two tasks is to follow the path $(1, 3, 4, 5)$ with the optimal cost to be $OPT = 5$.

2.3 Challenge and NP-hardness

Intuitively, TREK is hard because (i) path planning and speed planning are coupled with each other and should be tackled simultaneously, and (ii) more importantly, it requires to optimize execution times for individual tasks jointly such that total cost can be minimized, which a non-convex puzzle by itself as illustrated latter using a simple example. Such an optimization on task execution times differentiates our work from existing single-task ones (e.g. the one from [4]) where no execution times optimization will be involved when they are simply generalized to handle our problem in the multiple tasks setting.

Suppose we denote $C_i(T_i)$ as the optimal cost to fulfill the single task τ_i with travel time bounded above by a deadline T_i. Then our problem TREK can also be formulated easily as following with execution times T_i allocated to τ_i:

$$\min_{\vec{x}_i \in \mathcal{X}_i, \vec{t}_i \in \mathcal{T}_i} \quad \sum_{i=1}^{K} C_i(T_i) \tag{3a}$$

$$\text{s.t.} \quad a_i = \max\left\{a_{i-1}, T_i^{\text{out}}\right\} + T_i \leq T_{i+1}^{\text{in}},$$

$$a_0 = 0, \forall i = 1, 2, ..., K, \tag{3b}$$

$$\sum_{e \in E} x_i^e t_i^e \leq T_i, \forall i = 1, 2, ..., K, \tag{3c}$$

where we try to achieve the optimal task execution times allocation results $T_i, i = 1, 2, ..., K$ with all time window constraints satisfied. If we can find the optimal $T_i, i = 1, 2, ...K$ for all individual tasks, we can run existing single-task algorithm (e.g. the one proposed by [4]) K times independently and obtain the solution in the end. However, we argue that it is hard to optimize the execution times for individual times, due to the non-convexity of the function $C_i(\cdot)$.

Consider network in Fig. 1(a) where there is only one task with source 1 and destination 5. Edge travel time and cost are the same as defined in Tab. 2. It is clear that in Fig. 1(a) there are 5 different paths from node 1 to node 5. By enumerating all those 5 paths with associated path cost and path travel time, we have $C(T) = 4$ when $T = 4$ following the path $(1, 2, 3, 4, 5)$, $C(T) = 5$ when $T = 3$ following the path $(1, 3, 4, 5)$, $C(T) = 7$ when $T = 2$ following the

path $(1, 3, 5)$, and $C(T) = 8$ when $T = 1$ following the path $(1, 5)$. Clearly, the optimal cost $C(T)$ is neither convex nor concave in T.

The following theorem shows that TREK is NP-hard, which is not surprising since its special cases RSP [7, 16] and PASO [4] are both NP-hard.

Theorem 2.1. *TREK is NP-hard.*

Proof. Refer to Appendix 7.3. □

In this paper we develop an FPTAS and a heuristic to solve TREK. FPTAS optimizes the task execution times by intelligently enumerating possible results without incurring excessive complexity, and then selecting the best. And heuristic optimizes the task execution times by allocating deadlines for individual tasks jointly and iteratively towards the optimal based on dual sub-gradient information.

3 AN FPTAS FOR TREK

In this section we develop an FPTAS for TREK, based on Dynamic Programming (DP) and an extension of the FPTASes proposed by [4, 16] solving problem RSP and PASO, respectively. Our FPTAS solves the NP-hard problem TREK with an approximation ratio of $(1 + \epsilon)$ in fully polynomial time, for any $\epsilon > 0$.

3.1 Dynamic Programming for TREK

We observe that our problem TREK satisfies Bellman's principle of optimality, and hence can be solved by DP. Specifically, we divide problem TREK to two sub-problems: (i) we first obtain all possible solutions to independently fulfill the single task τ_i for all $i = 1, 2, ..., K$, by enumerating either all travel-time-bounded min-cost path or all cost-bounded min-travel-time path, and (ii) we then select exactly one solution per task, with the combined solution satisfying all time window constraints and minimizing the total cost. Both sub-problems can be handled by DP.

We need discrete (e.g., integral) edge travel time or cost in order to use DP to solve TREK. If we require integral edge travel time, the Bellman's equation for sub-problem 1 is simply the equation in (12), Appendix 7.2, and the Bellman's equation for sub-problem 2 is the equation in (14), Appendix 7.2. If we require integral edge cost instead of integral travel time, the Bellman's equation for sub-problem 1 is the equation in (13), Appendix 7.2, and the Bellman's equation for sub-problem 2 is the equation in (15), Appendix 7.2. Due to the space limitation, we put algorithmic details of proposed DP approach in Appendix 7.2. Overall, we can use DP to solve TREK, and optimal solution is guaranteed if either all edge travel time or all edge cost are required to be integers. However, the DP approach suffers from a pseudo-polynomial time complexity in theory.

A nature idea to solve TREK in polynomial time is to follow our DP with rounding and scaling on edge travel time (for Bellman's equation in (12) and in (14)) or edge cost (for Bellman's equation in (13) and in (15)). Considering that in our problem any violation of time window constraints is not allowed, clearly it is better to discretize, quantize and approximate edge cost such that precise edge travel time information can be kept. In this paper, we extend the FPTASes proposed by [4, 16] both of which use similar ideas of edge cost quantization, and design an FPTAS for TREK.

3.2 A Test Procedure

The essence of our FPTAS is a test procedure which approximately compares the optimal cost of TREK to an arbitrary input, with details introduced in Algorithm 1. The structure of Algorithm 1 is the same as the above DP approach, except that edge cost has been quantized to guarantee a polynomial time complexity: the loop of line 8 is equivalent to the Bellman's equation in (13) handling the sub-problem 1, and the loop of line 13 is equivalent to the Bellman's equation in (15) solving the sub-problem 2.

Algorithm 1 Test$(G, \vec{\sigma}, L, U, \epsilon)$

1: **procedure**
2: $S = \frac{L\epsilon}{K(n+1)}$, $\hat{U} = \lfloor \frac{U}{S} \rfloor + K(n+1)$, $p_{\text{test}} = \text{NULL}$
3: **for** $\forall i = 1, 2, ..., K, \forall e \in E, \forall \hat{c} = 1, 2, ..., \hat{U}$ **do**
4: $\hat{c}_e^l = \lceil c_e(\hat{t}_e(\rho_i), \rho_i)/S \rceil, \hat{c}_e^u = \lceil c_e(t_e^l, \rho_i)/S \rceil$
5: $t_i^e(\hat{c}) = \begin{cases} t_e^l, & \text{if } \hat{c} = \hat{c}_e^u \\ c_e^{-1}(\hat{c}S, \rho_i), & \text{if } \hat{c}_e^l \le \hat{c} < \hat{c}_e^u \\ +\infty, & \text{otherwise} \end{cases}$
6: Set $\mathcal{T}_i(v, 0) = +\infty, \forall i = 1, 2, ..., K, \forall v \neq \sigma_i : v \in V$
7: Set $\mathcal{T}_i(\sigma_i, 0) = 0, \forall i = 1, 2, ..., K$
8: **for** $\forall i = 1, 2, ..., K, \forall \hat{c} = 1, 2, ..., \hat{U}, \forall v \in V$ **do**
9: $\mathcal{T}_i(v, \hat{c}) = \mathcal{T}_i(v, \hat{c} - 1)$
10: **for** $\forall e = (u, v) \in E, \forall \tilde{c} = 1, 2, ..., \hat{c}$ **do**
11: $\mathcal{T}_i(v, \hat{c}) = \min\left\{\mathcal{T}_i(v, \hat{c}), \mathcal{T}_i(u, \hat{c} - \tilde{c}) + t_i^e(\tilde{c})\right\}$
12: Set $\mathcal{T}(\sigma_1, \hat{c}) = 0, \forall \hat{c} = 0, 1, ..., \hat{U}$
13: **for** $\forall i = 2, 3, ..., K + 1, \forall \hat{c} = 1, 2, ..., \hat{U}$ **do**
14: $\mathcal{T}(\sigma_i, \hat{c}) = \mathcal{T}(\sigma_i, \hat{c} - 1)$
15: **for** $\forall \tilde{c} = 1, 2, ..., \hat{c}$ **do**
16: $\mathcal{T} = \max\left\{\mathcal{T}(\sigma_{i-1}, \hat{c} - \tilde{c}), T_{i-1}^{\text{out}}\right\} + \mathcal{T}_{i-1}(\sigma_i, \tilde{c})$
17: $\mathcal{T} = \begin{cases} \mathcal{T}, & \text{if } \mathcal{T} \le T_i^{\text{in}}; \\ +\infty, & \text{otherwise,} \end{cases}$
18: $\mathcal{T}(\sigma_i, \hat{c}) = \min\{\mathcal{T}(\sigma_i, \hat{c}), \mathcal{T}\}$
19: $\hat{c}^*$ is the minimal $\hat{c} \in \{1, 2, ..., \hat{U}\} : \mathcal{T}(\sigma_{K+1}, \hat{c}) \le T_{K+1}^{\text{in}}$
20: If $\hat{c}^*$ exists, p_{test} is defined by $\mathcal{T}(\sigma_{K+1}, \hat{c}^*)$
21: **return** p_{test}

To be specific, as proved in our Lem. 7.1, Appendix 7.4, in Algorithm 1 we first quantize edge cost c to be $\hat{c}$ where $\hat{c} = \lceil c/S \rceil$ in the loop of line 3. As a result, a polynomial time complexity can be guaranteed for Algorithm 1 as described in our Lem. 7.5, Appendix 7.8. Moreover, such a quantization technique can provide bounded performance difference comparing the quantized cost to the precise cost before quantization, which is introduced in our Lem. 7.2, Appendix 7.5. Algorithm 1 is critical for our FPTAS because it can approximately compare the optimal cost of TREK, i.e. OPT, to an arbitrary input U, in the sense that if it returns a non-empty solution, it must hold that $\text{OPT} \le U + L\epsilon$ (Lem. 7.3, Appendix 7.6); otherwise, we must have $\text{OPT} > U$ (Lem. 7.4, Appendix 7.7).

We remark that the returned solution of Algorithm 1 must meet all the time window constraints, which is guaranteed by the Bellman's equation in (15) in the loop of line 13.

Algorithm 2 FPTAS-TREK($G, \vec{\sigma}, LB, UB, \epsilon$)

1: **procedure**
2: $p_{\text{fptas}} = \text{NULL}, B_L = LB, B_U = \lceil UB/2 \rceil$
3: **while** $B_U/B_L > 2$ **do**
4: $B = \sqrt{B_L \cdot B_U}$
5: **if** $\text{Test}(G, \vec{\sigma}, B, B, 1) = \text{NULL}$ **then**
6: $B_L = B$
7: **else**
8: $B_U = B$
9: $p_{\text{fptas}} = \text{Test}(G, \vec{\sigma}, B_L, 2B_U, \epsilon)$
10: **return** p_{fptas}

3.3 Proposed FPTAS

With our test procedure of Algorithm 1, we follow the same structure of the FPTAS proposed by [16] for solving RSP, and design an FPTAS in Algorithm 2 for our problem TREK in this section.

With an initial lower bound $LB \leq \text{OPT}$ and an initial upper bound $UB \geq \text{OPT}$, our FPTAS first narrows down the gap B_U/B_L where $2B_U$ (resp. B_L) is always a valid upper (resp. lower) bound of OPT (proved in our Lem 7.6, Appendix 7.9) using a binary search scheme. When the gap is below 2, the $(1+\epsilon)$-approximate solution is achieved by calling test procedure $\text{Test}(G, \vec{\sigma}, B, 2B, \epsilon)$ in line 9.

By the following two theorems, Algorithm 2 is proved to be an FPTAS for TREK, providing a $(1+\epsilon)$-approximation ratio in polynomial time with all problem input and $1/\epsilon$.

Theorem 3.1. *Given a lower bound LB and an upper bound UB of the optimal cost OPT, and assume that the feasible set of TREK is not empty, then Algorithm 2 will return a non-empty solution p_{fptas} that satisfies*

$$c\left(p_{fptas}\right) \leq (1+\epsilon)OPT,$$

where $c\left(p_{fptas}\right)$ represents the total cost of our solution p_{fptas}.

Proof. Refer to Appendix 7.10 due to the space limitation. □

Theorem 3.2. *Algorithm 2 has a time complexity of*

$$O\left(mK^3n^3\left(\log\log\frac{UB}{LB} + \left(1+\frac{1}{\epsilon}\right)^2\right)\right).$$

Proof. A direct result following Lem. 7.5. □

To initialize our FPTAS, we need to first obtain a lower bound LB and an upper bound UB for the optimal cost OPT. It is straightforward that $LB \geq K \cdot \min_{e\in E} c_e^l$ and $UB \leq Kn \cdot \max_{e\in E} c_e^u$, leading to that $UB/LB \leq n \cdot \max_{e\in E} c_e^u / \min_{e\in E} c_e^l$. Considering the time complexity described in Thm. 3.2, clearly that our proposed Algorithm 2 has a polynomial time complexity with all problem input $\{n, m, K, c_e^l, c_e^u : \forall e \in E\}$ and $1/\epsilon$. Since $\log\log\frac{UB}{LB}$ is relative small as compared to mK^3n^3/ϵ^2, we can roughly represent the time complexity of our FPTAS as $O(mK^3n^3/\epsilon^2)$.

In summary, the proposed FPTAS achieves an approximation ratio of $(1+\epsilon)$ for any $\epsilon > 0$ with fully polynomial time complexity. This makes it suitable for solving small-/medium- scale TREK instances. For large-scale instances such as those over the US national highway networks, however, we observe empirically that the FPTAS may take an excessive amount of time to generate its solution. This motivates us further to design a fast and efficient alternative for solving large-scale TREK instances.

4 SPEED: A FAST AND EFFICIENT HEURISTIC

In this section, we develop an efficient dual-based algorithm, named SPEED (in short for Sub-gradient-based Price-driven Energy-Efficient Delivery), for solving large-scale TREK instances. SPEED iteratively allocates execution times for individual tasks towards the optimal, by following the sub-gradient of the Lagrangian dual relaxation. We derive a performance bound between the solution of SPEED and the optimal solution, and we further characterize a condition under which SPEED generates an optimal solution.

4.1 A New Problem Formulation of TREK

In this section, we first reformulate TREK and give a new formulation in (4), which is characterized by the travel time upper bound constraint imposed on any consecutive tasks $\{\tau_k, \tau_{k+1}, ..., \tau_r, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r\}$:

$$\min_{\vec{x}_i \in X_i, \vec{t}_i \in \mathcal{T}_i} \quad \sum_{i=1}^{K}\sum_{e\in E} x_i^e \cdot c_e(t_i^e, \rho_i) \tag{4a}$$

$$\text{s.t.} \quad \sum_{j=k}^{r}\sum_{e\in E} x_j^e t_j^e \leq T_{r+1}^{\text{in}} - T_k^{\text{out}}, \quad \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r, \tag{4b}$$

Theorem 4.1. *The problems in* (2) *and* (4) *are equivalent in that they share the same objective function and the constraint sets are equivalent.*

Proof. Refer to Appendix 7.11 due to the space limitation. □

The deadline constraints in (4b) require that given any r and any $k \leq r$, the total travel time from σ_k to σ_{r+1}, i.e., the aggregated execution times of $\tau_k, \tau_{k+1}, ...\tau_r$, should be bounded above by $T_{r+1}^{\text{in}} - T_k^{\text{out}}$ which is the gap between the latest arrival time of σ_{r+1} and the earliest leaving time of σ_k. While the number of constraints in the formulation in (4) increases from K to K^2 as compared to the formulation in (2), the new formulation in (4) is critical for developing our sub-gradient based algorithm SPEED.

4.2 The Dual Problem of TREK

We relax the deadline constraints in (4b) to the objective function by introducing a Lagrangian dual variable λ_k^r for each (k, r) pair and obtain the following Lagrangian function

$$L\left(\vec{x}, \vec{t}, \vec{\lambda}\right) \triangleq \sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r \cdot \left(\sum_{j=k}^{r}\sum_{e\in E} x_j^e t_j^e - T_{r+1}^{\text{in}} + T_k^{\text{out}}\right) + \sum_{i=1}^{K}\sum_{e\in E} x_i^e \cdot c_e\left(t_i^e, \rho_i\right).$$

The specific dual variable λ_k^r in the Lagrangian function is only associated with the travel time of tasks from τ_k to τ_r. Now given a specific task τ_i, we look at all the dual variables associated with it. To this end we observe that λ_k^r is associated with τ_i if and only

if $k \le i \le r$, and the set of the dual variables associated with τ_i is thus $\{\lambda_k^r : \forall k \le i, \forall r \ge i\}$. We have

$$\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r\sum_{j=k}^{r}\sum_{e\in E}x_j^e t_j^e = \sum_{i=1}^{K}\sum_{e\in E}x_i^e t_i^e\sum_{r=i}^{K}\sum_{k=1}^{i}\lambda_k^r, \tag{5}$$

since both sides of the equation in (5) are the sum of the dual variable times the task fulfilling time over all correlative tasks and dual variables. Using the equation in (5), our Lagrangian function can be presented as follows

$$\begin{aligned} L\left(\vec{x},\vec{t},\vec{\lambda}\right) &= \sum_{i=1}^{K}\sum_{e\in E}x_i^e\cdot\left[c_e(t_i^e,\rho_i)+t_i^e\sum_{r=i}^{K}\sum_{k=1}^{i}\lambda_k^r\right] \\ &\quad -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r\left(T_{r+1}^{\text{in}}-T_k^{\text{out}}\right). \end{aligned}$$

We define μ_i as the sum of all the dual variables associated with task τ_i in the following

$$\mu_i \triangleq \sum_{r=i}^{K}\sum_{k=1}^{i}\lambda_k^r, \tag{6}$$

then our Lagrangian function is

$$\begin{aligned} L\left(\vec{x},\vec{t},\vec{\lambda}\right) &= \sum_{i=1}^{K}\sum_{e\in E}x_i^e\cdot\left[c_e(t_i^e,\rho_i)+t_i^e\mu_i\right] \\ &\quad -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r\left(T_{r+1}^{\text{in}}-T_k^{\text{out}}\right). \end{aligned}$$

The dual problem of TREK is

$$\max_{\vec{\lambda}\ge 0} D\left(\vec{\lambda}\right) : D\left(\vec{\lambda}\right) \triangleq \min_{\vec{x},\vec{t}} L\left(\vec{x},\vec{t},\vec{\lambda}\right),$$

where $D\left(\vec{\lambda}\right)$ is the dual function. The following theorem gives an explicit formula for $D\left(\vec{\lambda}\right)$.

Theorem 4.2.

$$D\left(\vec{\lambda}\right) = -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r\left(T_{r+1}^{in}-T_k^{out}\right)+\sum_{i=1}^{K}\sum_{e\in p(\mu_i)}w_i^e(\mu_i),$$

where

$$w_i^e(\mu_i) \triangleq c_e\left(t_i^e(\mu_i),\rho_i\right)+\mu_i t_i^e(\mu_i), \tag{7}$$

with

$$t_i^e(\mu_i) \triangleq \arg\min_{t_e^l\le t_i^e\le \hat{t}_e(\rho_i)}\left(c_e(t_i^e,\rho_i)+\mu_i t_i^e\right). \tag{8}$$

Proof. Refer to Appendix 7.12 due to the space limitation. □

In the above Thm. 4.2, $t_i^e(\mu_i)$ is the optimal travel time that minimizes the edge cost but with a price μ_i imposed on the edge travel time given specific μ_i. Since we have assumed $c_e\left(t_i^e,\rho_i\right)$ to be strictly convex and strictly decreasing over $t_i^e \in \left[t_e^l,\hat{t}_e(\rho_i)\right]$, $t_i^e(\mu_i)$ is well defined. Hence, $w_i^e(\mu_i)$ is the optimal penalized edge cost including the fuel consumption cost and the travel time cost given the price μ_i, and we denote the minimal penalized cost path from σ_i to σ_{i+1} as $p(\mu_i)$.

According to Thm. 4.2, given a set of dual variables $\vec{\lambda}$, we can obtain the value of the dual function $D\left(\vec{\lambda}\right)$ by solving K shortest path problems, each of which minimizes the penalized cost for a specific task. One can solve these shortest path problems by using standard algorithms, e.g., the Dijkstra's algorithm [5]. Therefore, we can design an efficient heuristic for TREK by iteratively update dual variables $\vec{\lambda}$ to minimize the duality gap.

4.3 The SPEED Algorithm

We further define $\delta(\mu_i)$ as the travel time of the path $p(\mu_i)$:

$$\delta(\mu_i) \triangleq \sum_{e\in p(\mu_i)} t_i^e(\mu_i). \tag{9}$$

Similar to [4, Thm.3], a key observation of $\delta(\mu_i)$ is presented in the following theorem.

Theorem 4.3. *$\delta(\mu_i)$ is non-increasing in $\mu_i \in [0,+\infty)$ for any $i = 1, 2, ..., K$.*

Proof. The proof is similar to that of [4, Thm.3] and is skipped. □

In the next, we introduce a set of complementary-slackness-like conditions for $\vec{\lambda}$ and $\delta(\cdot)$. If the conditions are satisfied, then we can derive an optimal solution to TREK.

Theorem 4.4. *Suppose the dual variable $\{\lambda_k^r : \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r\}$, the $\{\mu_i, \forall i = 1, 2, .., K\}$ computed in* (6)*, and the $\{\delta(\mu_i), \forall i = 1, 2, ..., K\}$) computed in* (9) *satisfy that*

$$\left[T_k^{out}-T_{r+1}^{in}+\sum_{j=k}^{r}\delta(\mu_j)\right]_{\lambda_k^r}^{+}=0,$$
$$\forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r, \tag{10}$$

where function $[f]_g^+$ is defined as

$$[f]_g^+ = \begin{cases} \max\{f,0\}, & \text{if } g \le 0; \\ f, & \text{otherwise.} \end{cases}$$

Then each $p(\mu_i)$ specifies a path for fulfilling task τ_i with the assigned travel time $t_i^e(\mu_i)$ for each edge $e \in p(\mu_i)$, and this solution must be the optimal solution to our problem TREK. The starting time and ending time of tasks $\{\tau_i, i = 1, 2..., K\}$ can be computed following the time window constraints in (2b) *from τ_1 to τ_K iteratively one by one.*

Proof. Refer to Appendix 7.13 due to the space limitation. □

SPEED uses a sub-gradient based heuristic scheme in Algorithm 3 to iteratively update dual variables towards satisfying conditions in (10):

$$\lambda_k^r = \phi(\lambda_k^r)\cdot\left[T_k^{\text{out}}-T_{r+1}^{\text{in}}+\sum_{j=k}^{r}\delta(\mu_j)\right]_{\lambda_k^r}^{+},$$
$$\forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r, \tag{11}$$

where $\phi(\lambda_k^r)$ is a positive step size to update λ_k^r. The step is positive instead of negative due to Thm. 4.3. The following theorem shows that SPEED always gives a feasible solution for any theoretically-feasible TREK instances. We further present the performance bound between SPEED solution and the optimal in the following theorem.

Algorithm 3 SPEED($G, \vec{\sigma}$)

1: **procedure**
2: Set ite=1, $\lambda_k^r = \dot{\lambda}_k^r = \lambda_{\max}, \forall r = 1, 2, ..., K, k = 1, 2, ..., r$.
3: **while** $\exists\, r, k : \left|\dot{\lambda}_k^r\right| >$ tol and ite $\leq$ ITE **do**
4: **for** $i = 1, 2, ..., K$ **do**
5: Set μ_i according to equality (6)
6: Obtain $t_i^e(\mu_i), \forall e \in E$ according to equality (8)
7: Set $w_i^e(\mu_i), \forall e \in E$ according to equality (7)
8: Get the $\sigma_i - \sigma_{i+1}$ shortest path $p(\mu_i)$ with $w_i^e(\mu_i)$
9: $p_h = \{p(\mu_i) : i = 1, 2, ..., K\}$, ite = ite + 1
10: Set $\dot{\lambda}_k^r, \forall r, \forall k$ according to equality (11)
11: Set $\lambda_k^r = \lambda_k^r + \dot{\lambda}_k^r, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r$
12: **if** $\dot{\lambda}_k^r = 0, \forall r = 1, 2, ..., K, k = 1, 2, ..., r$ **then**
13: **return** $p = p_h$
14: **if** p_h is feasible and $c(p_h) < c(p)$ **then**
15: $p = p_h$
16: **if** p_h is not feasible **then**
17: $\dot{\lambda}_k^r = \lambda_{\max}, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r$
18: **return** p

Theorem 4.5. *For any theoretically-feasible TREK instance, Algorithm 3 returns a feasible solution p. The optimality gap between the cost of p, namely c(p), and the optimal cost, namely OPT, must be bounded above as follows:*

$$c(p) - OPT \leq \begin{cases} 0, & \text{case 1;} \\ K^2 \cdot \max_{\forall r,k} \left|\dot{\bar{\lambda}}_k^r\right| \cdot \max_{\forall r,k} \left\{\bar{\lambda}_k^r / \phi(\bar{\lambda}_k^r)\right\}, & \text{case 2,} \end{cases}$$

where case 1 *is for p returned in line 13, case* 2 *is for p returned in line 18.* $\{\bar{\lambda}_k^r, \forall r = 1, ..., K, \forall k = 1, ..., r\}$ *is the set of dual variables corresponding to p.*

Proof. Refer to Appendix 7.14 due to the space limitation. □

From Thm. 4.5, it is clear that if Algorithm 3 is terminated in the line 13, the obtained solution is optimal. Another direct application of our Thm. 4.5 is to obtain a high-quality lower bound of the optimal cost in simulations, and thus efficiently evaluate performances of different algorithms for TREK. Practically, one critical application of Thm. 4.5 is that it can help us terminate Algorithm 3 and achieve a solution meeting all time window constraints, once the optimality gap of the achieved solution is below a user-defined tolerance, hence leading to much shorter algorithm running time without waiting for the convergence when $\dot{\lambda}_k^r = 0, \forall r = 1, ..., K, \forall k = 1, ..., r$.

Overall in this section, we develop an efficient heuristic SPEED (Algorithm 3) for TREK, by iteratively updating dual variables (equality (11)) and then solving the dual function (Thm. 4.2), in order to minimize the duality gap (Thm. 4.4). Hence SPEED can allocate task execution times efficiently using the feedback of dual sub-gradient information. We characterize a performance bound of the solution of SPEED, and moreover, give conditions under which the solution of SPEED is guaranteed to be optimal (Thm. 4.5).

Table 3: Performance comparison of FPTAS (F) and SPEED (S), where the unit of fuel consumption is gallon and the unit of the running time is second. LB is a lower bound for the optimal solution and obtained from Thm. 4.5.

Problem Input		Fuel-Cost			Run-Time	
$(\sigma_1, \sigma_2, \sigma_3, T_2^{\text{in}}, T_3^{\text{in}})$	ϵ	LB	S	F	S	F
(2, 1, 6, 7, 12)	0.1	123.45	125.55	125.9	44	5760
(14, 17, 18, 7, 12)	0.1	135.18	137.88	N/A	65	N/A

5 PERFORMANCE EVALUATION

We use real-world traces to evaluate our FPTAS and SPEED, by comparing them with fastest-/shortest- path baselines and a conceivable alternative which directly generalizes the state-of-the-art single-task algorithm from [4] to the multi-task settings studied in this paper. We implement all algorithms using C++ and run them on a laptop with 4-core Intel Core-i5-4200M (2.50 GHz) processor and 16 GB memory, running 64-bit Ubuntu 16.04 LTS, for comparison. We use the SNAP graph structure [15] and construct the U.S. national highway system (NHS) from the Clinched Highway Mapping Project [22] consisting of 84504 nodes and 178238 directed edges. Road grade is obtained from the elevations of each node provided by the Elevation Point Query Service [26] from the U.S. Geological Survey. The maximum speed r_e^u is set to be the historical average speed by collecting real-time speed data from HERE map [8] for 2 weeks, and the minimum speed r_e^l is manually set to be $r_e^l = \min\{30, r_e^u\}$. We use ADVISOR simulator [17] to collect fuel consumption rate data with driving speed (from 10mph to 70mph with a step of 0.2mph) for different road grade (from $-10.0°$ to $10.0°$ with a step of $0.1°$) and truck load (empty load, half load, and full load) for a class 8 heavy truck of Kenworth T800 [11]. Then we use the curve fitting toolbox in MATLAB to learn the fuel consumption rate function $f_e(r_e, \rho)$ modeled as a 3-order polynomial function with speed given specific road grade and truck load.

Same to [4], the NHS graph is preprocessed first: (i) the graph is cut to the "eastern" part, (ii) non-intersection roads with the same grade level are merged into a single road segment, and (iii) the "eastern" U.S. is divided into 22 regions (see Fig. 1(b)), where the node nearest to each region's center is used as the source and destination nodes in the experiment. After preprocessing, the number of nodes is 38213 and the number of edges is 82781.

In our simulations, we set all earliest leaving time constraints to be 0, namely $T_i^{\text{out}} = 0, \forall i = 1, ..., K$. As for the experimental parameters of our SPEED in Algorithm 3, we fix $\phi(\cdot)$ to be 0.1 for all dual variables, tol to be 0.01 and ITE to be 50.

5.1 Performance Comparison of FPTAS and SPEED

We consider the setting where the truck fulfills two tasks both with full load. Results for two instances are shown in Tab. 3. In both instances, the solutions of SPEED are close-to-optimal, with the optimality gap being upper bounded by 3 gallons (2% of the optimal cost). FPTAS can only handle the smallest instance (first instance with $n = 1518$ and $m = 3274$) for $\epsilon = 0.1$. Compared with SPEED, FPTAS incurs much longer (100×) running time for the first instance. For the second instance with $n = 6187$ and $m = 13708$ where source

and destination of each task are located in neighboring regions, FPTAS fails to output a solution even after running it for 12 hours.

5.2 Performance Comparison of SPEED and Other Alternatives

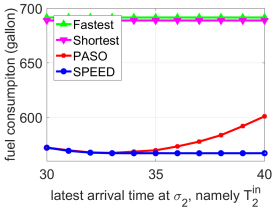
In this section we compare SPEED with two baselines and a conceivable alternative approach:

(1) A fastest-path-based approach without task execution times optimization: the travel speed r_e for each $e \in E$ is fixed as $r_e = r_e^u$, and each task τ_i is fulfilled using the path with minimal travel time from its source σ_i to its destination σ_{i+1}.
(2) A shortest-path-based approach without task execution times optimization: the travel speed r_e for each $e \in E$ is fixed as $r_e = r_e^u$, and each τ_i is fulfilled using the path with minimal travel distance from its source σ_i to its destination σ_{i+1}.
(3) A PASO-based approach with greedy task execution times allocation: we greedily allocate deadlines as large as possible for individual tasks from the first task to the last task iteratively one by one. To be specific in the ith iteration, we run the heuristic proposed by [4] to solve problem PASO for the single task τ_i, minimizing the fuel consumption from σ_i to σ_{i+1} with a deadline of $T_{i+1}^{\text{in}} - \max\{T_i^{\text{out}}, a_{i-1}\}$, where a_{i-1} is defined in formula (2), which is the truck arrival time at σ_i and can be achieved since we have solved the PASO problems for tasks $\{\tau_1, \tau_2, ..., \tau_{i-1}\}$ before the ith iteration.

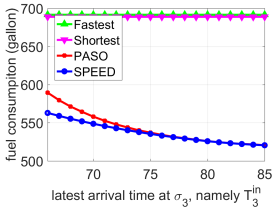
We consider two tasks with the first task from 1 to 9 and the second task from 9 to 22, assuming full load for both tasks and $T_2^{\text{in}} = 40, T_3^{\text{in}} = 65$. This TREK instance is denoted as a tuple of $(\sigma_1, \sigma_2, \sigma_3, T_2^{\text{in}}, T_3^{\text{in}}) = (1, 9, 22, 40, 65)$. We remark that the instance $(1, 9, 22, 40, 65)$ is representative because (i) both tasks are heavy-duty (both tasks are assumed with a full load requirement) and long-haul (both tasks require the truck to travel across US), and (ii) the latest arrival time constraints T_2^{in} and T_3^{in} are larger than the minimal task execution times, allowing a large design space for task execution times optimization for saving fuel.

Detailed simulation results are shown in Tab. 4. In the table we also present the increment (%) of the travel time, travel distance and fuel consumption for the four solutions compared with the respective optimums. As shown by Tab. 4, both the fastest solution and the shortest solution consumes ~30% more fuel than SPEED. The PASO solution saves fuel for the individual task τ_1 in the cost of a larger travel time compared with SPEED. However, its solution is far from optimal, due to its greedy and non-optimal task execution times allocation (~40 hours for τ_1 and ~25 hours for τ_2). The solution of SPEED is close-to-optimal and saves 10% fuel compared with the PASO solution, with a close-to-optimal execution time allocated for each task (~29 hours for τ_1 and ~36 hours for τ_2).

We already know our problem TREK requires a joint optimization on task execution times, path planning, and speed planning. Tab. 4 presents the impacts of the three optimization aspects on saving fuel, by comparing SPEED with the non-optimal baselines and the PASO alternative. From Tab. 4, the travel distance of the four algorithms are similar, in spite of the various travel time and fuel consumption. This highlights the importance of exploring speed planning and task execution times optimization, in addition to path planning, in reducing fuel consumption.



(a) $(1, 22, 9, T_2^{\text{in}}, 65)$: T_2^{in} is enumerated from 30 to 40 with a unit step.



(b) $(1, 22, 9, 40, T_3^{\text{in}})$: T_3^{in} is enumerated from 66 to 85 with a unit step.

Figure 2: Impact of the latest arrival time T_2^{in} and T_3^{in} on the total fuel consumption performance.

5.3 Impact of Time Windows on the Cost

For instances of two tasks each with a full load, we respectively estimate the effect of the two latest arrival time constraints T_2^{in} and T_3^{in} on the performance of different algorithms. For the instance of τ_1 from 1 to 22 and τ_2 from 22 to 9, we first fix T_3^{in} to be 65 and evaluate effect from T_2^{in}. Results are shown in Fig. 2(a). The fastest-path-based solution performs almost same as the the shortest-path-based solution, and is independent to T_2^{in}. For $T_2^{\text{in}} \leq 32$, both PASO and SPEED obtain near-optimal performances, because the greedy task execution times allocation (T_2^{in} hours for τ_1 and $65 - T_2^{\text{in}}$ hours for τ_3) is close-to-optimal. For $T_2^{\text{in}} \in (32, 40]$, SPEED guarantees a close-to-optimal solution and greatly outperforms PASO, since the greedy allocation in PASO is far from optimal (the optimal travel time for τ_1 is strictly smaller than T_2^{in}). Moreover, PASO fails to obtain a feasible solution for $T_2^{\text{in}} \geq 41$, since following the greedy allocation strategy, τ_2 is assigned a deadline of $65 - T_2^{\text{in}} < 24$ which is smaller than the minimal travel time (24.55) from σ_2 to σ_3. Meanwhile, SPEED always provides near-optimal feasible solutions.

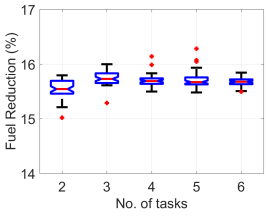
We then fix T_2^{in} to be 40 and evaluate the effect from T_3^{in} with the results presented in Fig. 2(b). For $T_3^{\text{in}} \in [65, 79]$, SPEED allocates execution times for tasks properly (a travel time strictly smaller than 40 hours for τ_1 and remaining for τ_2), while PASO cannot (it always assign ~40 hours to τ_1 and $T_3^{\text{in}} - 40$ hours to τ_2). For $T_3^{\text{in}} \geq 80$, both PASO and SPEED allocate execution times for tasks properly.

For the instance $(1, 22, 9, T_2^{\text{in}}, 65)$, since the minimal execution time of τ_1 (resp. τ_2) is 24.84 (resp. 24.55), clearly that the feasibility region of T_2^{in} where our TREK is solvable is $T_2^{\text{in}} \in [24.84, 65]$. Compared with SPEED which always gives near-optimal solutions for all solvable problem instances, PASO may fail to generate a feasible solution, because that the greedy task execution times allocation scheme fails to allocate a feasible task execution time for the task τ_2 when $T_2^{\text{in}} > 65 - 24.55$.

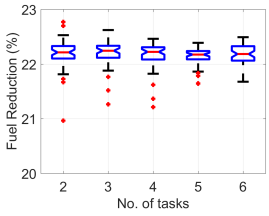
We further estimate the number of solvable problem instances for PASO with 100 random simulations. In each simulation, we randomly select σ_1, σ_2 and σ_3, assuming full load for both tasks. Suppose the minimal execution time for τ_1 is t_1^* (resp. t_2^* for τ_2), we randomly select an increment $x\% \in [10\%, 100\%]$ and fix T_3^{in} to be $T_3^{\text{in}} = (t_1^* + t_2^*) \cdot (1 + x\%)$. For this simulation characterized by $(\sigma_1, \sigma_2, \sigma_3, T_2^{\text{in}}, T_3^{\text{in}})$ with the feasibility region of $T_2^{\text{in}} \in [t_1^*, T_3^{\text{in}}]$, we run the PASO-based alternative to obtain the maximum time

Table 4: Comparison of SPEED and other alternatives for the instance of $(1, 9, 22, 40, 65)$, where a lower bound for the optimal fuel consumption is 478.73 according to Thm. 4.5. Unit: hour for travel time, mile for distance, and gallon for fuel consumption.

Algorithm	First task from 1 to 9			Second task from 9 to 22			Total performance					
	Time	Distance	Fuel	Time	Distance	Fuel	Time	Incre.	Distance	Incre.	Fuel	Incre.
Fastest	19.54	1306	276.5	24.48	1613	337.8	44.02	-	2919	0.07	614.3	28.32
Shortest	19.56	1306	276.18	24.58	1611	338.34	44.14	0.27	2917	-	614.52	28.36
PASO	39.93	1307	202.92	25.06	1613	329.94	64.99	47.64	2920	0.1	532.86	11.31
SPEED	29.03	1307	215.25	35.96	1616	264.52	64.99	47.64	2923	0.21	479.77	0.22

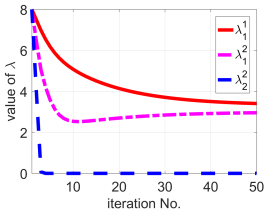


(a) Simulations for $x = 25$.

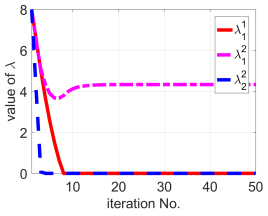


(b) Simulations for $x = 50$.

Figure 3: The fuel reduction of SPEED as compared to the fastest-path baseline, as a function of K. All latest arrival time constraints are set to be $t^* \cdot (1 + x\%)$ with t^* to be the minimal time to fulfill K randomly selected tasks.



(a) Instance (1, 22, 9, 30, 65).



(b) Instance (1, 22, 9, 40, 65).

Figure 4: Convergence of dual variables in SPEED.

$t \in [t_1^*, T_3^{\text{in}}]$ such that PASO can return a feasible solution for the instance $(\sigma_1, \sigma_2, \sigma_3, t, T_3^{\text{in}})$. After those 100 simulations, we observe that PASO can only solve 52.6% of theoretical solvable problem instances on average. In contrast, SPEED always outputs close-to-optimal solutions meeting all time window constraints.

5.4 Fuel Saving of SPEED is Robust to the Number of Tasks

In this section, we carry out extension simulations to evaluate the performance of SPEED when the number of tasks increases.

For each experimental case characterized by a specific $K \in \{2, 3, ..., 6\}$ and an $x \in \{25, 50\}$, we run 100 simulations each with K full-load tasks defined by random selected source/destination nodes $\{\sigma_1, ..., \sigma_{K+1}\}$. Latest arrival time constraints are same for any $\sigma_i, i = 2, ..., K+1 : T_i^{\text{in}} = t^* \cdot (1 + x\%)$, where t^* is the minimal time to fulfill those K tasks. Fig. 3 shows the fuel reduction results of SPEED compared with the fastest-path baseline. In all simulations, the performance of shortest-path-based baseline is almost same as the fastest-path baseline, and the PASO approach cannot give feasible solutions, because the greedy task execution times allocation cannot allocates a deadline that is large enough to timely fulfill the last task τ_K when the input latest arrival time constraints are the same as in our simulations. Fig. 3 suggests that the fuel saving of SPEED is robust to the number of tasks (~16% for $x = 25$ and ~22% for $x = 50$, both independent of K).

We also show the convergence of dual variables in each iteration in Fig. 4 for two instances. In Fig. 4(a) (resp. Fig. 4(b)), the absolute value of the sub-gradient of all three dual variables are within tol, which is an experimental parameter in Algorithm 3 to bound the number of iterations of SPEED, after 45 (resp. 16)iterations and thus SPEED terminates.

6 CONCLUSION

We consider the scenario where a truck drives across a national highway system to fulfill multi-tasks in a specific order. We formulate a problem TREK, whose objective is to minimize the total fuel consumption subject to the pickup and delivery time window constraints of individual tasks. Solving TREK is uniquely challenging due to the coupling among task execution times optimization, path planning, and speed planning. We first prove TREK is NP-hard, and argue that optimizing task execution times by itself is a non-convex puzzle. We then exploit the problem structure to develop (i) a Fully-Polynomial-Time Approximation Scheme (FPTAS), and (ii) a fast dual sub-gradient based heuristic SPEED. We provide performance guarantees of both algorithms. We further characterize a sufficient condition under which SPEED generates an optimal solution.

We evaluate the performance of our solutions using real-world traces over the US national highway system. Our solutions can save up to 22% fuel as compared to fastest-/shortest- path baselines, and up to 10% fuel than a conceivable alternative that directly generalizes the state-of-the-art single-task algorithm to the multi-task settings studied in this paper. The fuel saving is robust to the number of tasks to be fulfilled. A set of simulations also show that our algorithms always obtain close-to-optimal solutions with all time window constraints satisfied for any theoretically feasible instance, while the conceivable alternative fails to meet one or more time window constraints for 47% of the instances.

ACKNOWLEDGEMENT

This work was supported in part by the University Grants Committee of the Hong Kong Special Administrative Region, China (Theme-based Research Scheme Project No. T23-407/13-N and Collaborative Research Fund No. C7036-15G). We thank Dr. Lei Deng, an assistant professor in School of Electrical Engineering&Intelligentization, Dongguan University of Technology, for sharing the simulation code of PASO [4] and for his kind assistance on experiments.

REFERENCES

[1] A. T. Association et al. Ata american trucking trends, 2017.
[2] Secrets of better fuel economy. https://cumminsengines.com/uploads/docs/cummins_secrets_of_better_fuel_economy.pdf.
[3] E. Demir, T. Bekta, and G. Laporte. A comparative analysis of several vehicle emission models for road freight transportation. *Transportation Research Part D: Transport and Environment*, 2011.
[4] L. Deng, M. H. Hajiesmaili, M. Chen, and H. Zeng. Energy-efficient timely transportation of long-haul heavy-duty trucks. In *International Conference on Future Energy Systems (e-Energy)*. ACM, 2016.
[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1959.
[6] Fuel economy at various driving speeds. http://www.afdc.energy.gov/data/10312.
[7] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations research*, 1992.
[8] Traffic flow using corridor in HERE maps. https://developer.here.com/api-explorer/rest/traffic/flow-using-corridor.
[9] A. Hooper and D. Murray. An analysis of the operational costs of trucking: 2017 update. In *American Transportation Research Institute*, 2017.
[10] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó. Lagrange relaxation based method for the qos routing problem. In *International Conference on Computer Communications (INFOCOM)*. IEEE, 2001.
[11] Kenworth T800 vehicle. http://www.kenworth.com/trucks/t800.
[12] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 1992.
[13] H. C. Lau and Z. Liang. Pickup and delivery with time windows: Algorithms and test case generation. *International Journal on Artificial Intelligence Tools*, 2002.
[14] E. L. Lawler, J. K. Lenstra, A. R. Kan, D. B. Shmoys, et al. *The traveling salesman problem: a guided tour of combinatorial optimization*. Wiley New York, 1985.
[15] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology*, 2016.
[16] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 2001.
[17] T. Markel, A. Brooker, T. Hendricks, V. Johnson, K. Kelly, B. Kramer, M. O'Keefe, S. Sprik, and K. Wipke. ADVISOR: a systems analysis tool for advanced vehicle modeling. *Journal of Power Sources*, 2002.
[18] J. Monnot, V. T. Paschos, and S. Toulouse. Approximation algorithms for the traveling salesman problem. *Mathematical methods of operations research*, 2003.
[19] J. Nalepa and M. Blocho. Adaptive guided ejection search for pickup and delivery with time windows. *Journal of Intelligent & Fuzzy Systems*, 2017.
[20] A. I. Savran, E. Musaoglu, C. Yildiz, M. F. Yuce, and E. Yesil. Extended heuristic bubble algorithm for the pickup and delivery problem with time windows. In *International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2015.
[21] M. Stone. Tampa bay regional goods movement study-local issues. In *68th Annual Meeting of the Institute of Transportation Engineers*, 1998.
[22] J. D. Teresco. The Clinched Highway Mapping (CHM) Project. http://cmap.m-plex.com/.
[23] M. Tunnell. Estimating truck-related fuel consumption and emissions in maine: A comparative analysis for six-axle, 100,000 pound vehicle configuration. Technical report, 2011.
[24] Uber freight. https://freight.uber.com/.
[25] united press international. 55 mile-per-hour speed limit approved by house, 1973, retrieved 2008.
[26] USGS Elevation Point Query Service. http://nationalmap.gov/epqs/.
[27] uShip. https://www.uship.com/.

7 APPENDIX

7.1 Summary of Key Notations

We summarize key notations in Tab. 5.

Table 5: Summary of key notations.

Notations	Definitions
$G \triangleq (V, E)$	The highway network G with connecting points V and road segments E
$f_e(\cdot)$	Fuel consumption rate function of $e \in E$
$c_e(\cdot)$	Fuel consumption function of $e \in E$
D_e	Distance of road segment $e \in E$
r_e^l (resp. r_e^u)	Minimum (resp. maximum) speed of e
t_e^l (resp. t_e^u)	Minimum (resp. maximum) travel time of e
c_e^l (resp. c_e^u)	Minimum (resp. maximum) fuel consumption of road segment e
$\hat{t}_e(\rho)$	The specific travel time $t \in [t_e^l, t_e^u]$ that minimizes $c_e(\cdot)$ given a truck load ρ
$\tau_i, i \in [1, K]$	The i^{th} task
s_i (resp. d_i)	Source node (resp. destination node) of the i^{th} task
$s\omega_i$ (resp. $d\omega_i$)	Source pickup window (resp. destination delivery window) of the i^{th} task
ρ_i	Weight of cargos (load) of the i^{th} task
$\alpha(s\omega_i)$ (resp. $\beta(s\omega_i)$)	Earliest (resp. latest) pickup time of the i^{th} task
$\alpha(d\omega_i)$ (resp. $\beta(d\omega_i)$)	Earliest (resp. latest) delivery time of the i^{th} task
$\sigma_i, i \in [1, K+1]$	Node s_i, or equivalently the node d_{i-1}
T_i^{out} (resp. T_i^{in})	Earliest leaving (resp. latest arrival) time of node σ_i considering $s\omega_i$ and $d\omega_{i-1}$

7.2 Dynamic Programming For TREK

In this section we give algorithmic details of using DP to solve our problem TREK which satisfies Bellman's principle of optimality. The DP approach has a pseudo-polynomial time complexity, and is the foundation of our FPTAS introduced in Sec. 3.

In TREK, clearly the path travel time from s_i to d_i to fulfill τ_i in the optimal solution must be upper bounded by $\beta(d\omega_i) - \alpha(s\omega_i)$. If we denote p_i^t as the path from s_i to d_i with the minimal cost and a travel time upper bounded by t, the path in the optimal solution for the truck to fulfill τ_i must belong to the set P_i defined by

$$P_i \triangleq \left\{p_i^t : \forall t \in (0, \beta(d\omega_i) - \alpha(s\omega_i)]\right\},$$

In order to solve TREK, we divide it to two sub-problems: (i) we first obtain P_i for each τ_i, and (ii) we then figure out the solution to TREK by selecting exactly one path p_i from P_i for each task τ_i with the combined solution $p = p_1 \cup p_2 ... \cup p_K$ meeting all time window constraints and minimizing the total cost. We observe both sub-problems meet the Bellman's principle of optimality and thus can be solved by DP.

7.2.1 Sub-Problem 1: Obtain P_i for Each τ_i. If we require each edge travel time to be integral, for the task τ_i, let us define $C_i(v, T)$ to be the minimal path cost of all paths from s_i to v with travel time bounded above by T for the truck with load ρ_i. This enables us to write the principle of optimality equation as

$$C_i(v, T) = \min\left\{C_i(v, T-1), \min_{\substack{\{u,t\}\in \\ \mathcal{M}_i(v,T)}} \left\{C_i(u, T-t) + c_{(u,v)}(t, \rho_i)\right\}\right\}, \tag{12}$$

with $\mathcal{M}_i(v, T)$ defined as

$$\mathcal{M}_i(v, T) \triangleq \{\{u, t\} : \quad \forall u \in V \text{ s.t. } \exists e = (u, v) \in E. \forall t \text{ s.t. } t \le T, \\ t \in \left[t_e^l, \hat{t}_e(\rho_i)\right], t \in \mathbb{Z}^+\},$$

and boundary conditions

$$C_i(s_i, T) = 0, \forall T = 0, 1, ..., \beta(d\omega_i) - \alpha(s\omega_i),$$

$$C_i(v,0) = +\infty, \forall v \in V \backslash \{s_i\}.$$

The Bellman's equation (12) states that for a single task τ_i, the minimal cost from s_i to a certain node v with travel time upper bounded by T is the smaller one comparing (i) the minimal cost from s_i to v with travel time bounded above by $T-1$, and (ii) the minimal cost from s_i to a node u with travel time upper bounded by $T-t$, plus the minimal cost to pass the edge (u,v) with travel time no larger than t. Clearly, $C_i(d_i,t)$ is the cost of the path p_i^t which can be obtained recursively following the Bellman's equation (12). Such a DP approach has a pseudo-polynomial time complexity because its size is polynomial with the $\beta(d\omega_i)$, but exponential with the bit length of the input, i.e. $\log(\beta(d\omega_i))$.

Instead of integer travel time, if only cost is required to be integral, similar Bellman's equation exits. We define $\mathcal{T}_i(v,C)$ to be the minimal path travel time of all paths from s_i to v with cost bounded above by C for the truck with load ρ_i. Then we have:

$$\mathcal{T}_i(v,C) = \min\left\{\mathcal{T}_i(v,C-1), \min_{\substack{\{u,c\}\in \\ \mathcal{N}_i(v,C)}} \left\{\mathcal{T}_i(u,C-c) + c^{-1}_{(u,v)}(c,\rho_i)\right\}\right\}, \tag{13}$$

with $\mathcal{N}_i(v,C)$ defined as

$$\mathcal{N}_i(v,C) \triangleq \{\{u,c\}: \quad \forall u \in E \text{ s.t. } \exists e=(u,v)\in E. \forall c \text{ s.t. } c \le C, \\ c \in \left[c_e(\hat{t}_e(\rho_i),\rho_i), c_e(t_e^l,\rho_i)\right], c \in \mathbb{Z}^+\},$$

and boundary conditions

$$\mathcal{T}_i(s_i,C) = 0, \forall C = 0,1,...,C_{\max},$$
$$\mathcal{T}_i(v,0) = +\infty, \forall v \in V\backslash\{s_i\},$$

where $C_{\max}$ is an upper bound for the cost in the optimal solution from s_i to d_i to fulfill τ_i, e.g. it can be $n \cdot c_{\max} : c_{\max} = \max_{e\in E} c_e^u$. For the Bellman's equation (13), the cost of p_i^t clearly is $\min_{0\le C\le C_{\max}}\{C : \mathcal{T}_i(d_i,C) \le t\}$.

7.2.2 Sub-Problem 2: the Combined Solution. In sub-problem 1, for each τ_i, we have achieved P_i through obtaining either all the travel-time-bounded optimal cost $C_i(d_i,T)$, or all the cost-bounded optimal travel time $\mathcal{T}_i(d_i,C)$. Our sub-problem 2 requires to pick exactly one p_i from each P_i such that the combined solution $p = p_1 \cup p_2 ... \cup p_K$ can optimize the total cost with all time window constraints satisfied. We propose to use DP to solve sub-problem 2.

With integer travel time assumption, we define $C(\sigma_i,T)$ as minimal cost after the truck has fulfilled tasks from τ_1 to τ_{i-1} with all associated time window satisfied, given that the truck is now at σ_i when the time is T. Then the principle of optimality equation is

$$C(\sigma_i,T) = \begin{cases} \min\{C(\sigma_i,T-1), C^*(\sigma_i,T)\}, & T \le T_i^{\text{in}}; \\ C(\sigma_i,T_i^{\text{in}}), & \text{otherwise}, \end{cases} \tag{14}$$

where the definition of $C^*(\sigma_i,T)$ is

$$C^*(\sigma_i,T) = \min_{\substack{t\in\mathbb{Z}^+, t\le T, \\ T-t\ge T_{i-1}^{\text{out}}}} \{C(\sigma_{i-1},T-t) + C_{i-1}(d_{i-1},t)\},$$

where $C_{i-1}(d_{i-1},t)$ is the same defined as in the equation (12). Note that in the above definition constraint $T - t \ge T_{i-1}^{\text{out}}$ guarantees that the truck cannot leave σ_{i-1} before the allowed earliest leaving time. The boundary conditions are

$$C(\sigma_1,T) = 0, \forall T = 0,1,...,T_{K+1}^{\text{in}}$$
$$C(\sigma_i,0) = +\infty, \forall i = 2,3,...,K+1.$$

Clearly $C(\sigma_{K+1}, T_{K+1}^{\text{in}})$ gives optima to TREK. The feasibility is guaranteed in each step of the DP where both earliest leaving time and latest arrival time are satisfied in the equation (14). Similar to sub-problem 1, the time complexity of the DP targeting sub-problem 2 is pseudo-polynomial due to that it is exponential with the bit length of the input time window constraint, i.e. $\log\left(T_{K+1}^{\text{in}}\right)$.

If we instead only require the edge cost to be integral, similar Bellman's equation also exists in our sub-problem 2. We define $\mathcal{T}(\sigma_i,C)$ as the earliest arrival time at node σ_i with cost bounded above by C, in order to pick up the task τ_i after the truck has fulfilled the tasks from τ_1 to τ_{i-1} within associated time windows. Then the principle of optimality is

$$\mathcal{T}(\sigma_i,C) = \min\left\{\mathcal{T}(\sigma_i,C-1), \min_{c\in\mathbb{Z}^+, c\le C} \mathcal{T}^*(\sigma_i,c,C)\right\}, \tag{15}$$

where $\mathcal{T}^*(\sigma_i,c,C)$ is

$$\mathcal{T}^*(\sigma_i,c,C) = \begin{cases} T, & T \le T_i^{\text{in}}; \\ +\infty, & \text{otherwise}, \end{cases}$$
$$T = \max\left\{\mathcal{T}(\sigma_{i-1},C-c), T_{i-1}^{\text{out}}\right\} + \mathcal{T}_{i-1}(d_{i-1},c).$$

In equation (15), $\mathcal{T}^*(\sigma_i,c,C)$ is the earliest arrival time at σ_i with a cost bound C considering the case where the truck first arrives at σ_{i-1} with a cost upper bound $C-c$ and then drives from σ_{i-1} to σ_i fulfilling the task τ_{i-1} with the associated path cost bounded above by c. In the definition of $\mathcal{T}^*(\sigma_i,c,C)$, earliest leaving time is satisfied by comparing the $\mathcal{T}(\sigma_{i-1},C-c)$ with the constraint T_{i-1}^{out}, and latest arrival time is also satisfied since we set $\mathcal{T}^*(\sigma_i,c,C)$ to be infinity if the truck arrives at σ_i later than the allowed latest arrival time. The boundary conditions are

$$\mathcal{T}(\sigma_1,C) = 0, \forall C = 0,1,...,KC_{\max},$$
$$\mathcal{T}(\sigma_i,0) = +\infty, \forall i = 2,3,...,K+1,$$

where $C_{\max}$ is the same defined as in the equation (13) and hence $KC_{\max}$ is an upper bound for the optimal travel cost of TREK. Following Bellman's equation (15), the optimal solution of TREK with the requirement of integer edge cost is defined clearly by

$$\text{OPT} = \min_{0\le C\le KC_{\max}} \left\{C : \mathcal{T}(\sigma_{K+1},C) \le T_{K+1}^{\text{in}}\right\}.$$

Although proposed DP approach in this section requires either integer travel time or integer cost, with proper rounding and scaling technique, it can achieve solutions with performance guarantee for general TREK instances without integer time/cost assumptions, which is the key idea of our FPTAS designed in Sec. 3.

7.3 Proof to Thm. 2.1

We prove our problem TREK is NP-hard in this section.

Proof. Following the definition of RSP described in [7], given any RSP instance with input $\{G,s,d,T\}$ where G is the graph, s (resp. d) is the source (resp. destination) node, T is the travel time upper bound from s to d, each edge $e \in E$ has a positive integer

travel time t_e and a positive integer cost c_e, we can define a TREK problem instance to find the cost-optimal path to fulfill only one task τ_1 over the graph G with each edge has a fixed travel time $t_e^l = t_e^u = t_e$ and a fixed cost $c_e^l = c_e^u = c_e$. τ_1 is defined as

$$\tau_1 \triangleq \{s_1 = s, d_1 = d, s\omega_1 = [0, T], d\omega_1 = [0, T], \rho_1 = 0\}.$$

It is straightforward that the RSP instance and the constructed TREK instance are equivalently. Thus, TREK is NP-hard since RSP has been proved to be NP-hard [16]. □

7.4 Proof to Lem. 7.1

Lemma 7.1. *In Algorithm 1, given any specific $\hat{c} \in \{1, 2, ..., \hat{U}\}$, $e \in E, i \in \{1, 2, ..., K\}$, calculated $t_i^e(\hat{c})$ (line 5) is the minimal travel time $t \in [t_e^l, \hat{t}_e(\rho_i)]$ that meets $\hat{c} = \lceil c_e(t, \rho_i)/S \rceil$.*

Proof. Due to the strictly decreasing property of $c_e(t, \rho_i)$ with t given ρ_i, for the time t in the range of $t \in [t_e^l, \hat{t}_e(\rho_i)]$, we know the cost c will be in the range of $c \in [c_e(\hat{t}_e(\rho_i), \rho_i), c_e(t_e^l, \rho_i)]$. Considering the cost quantization approach of $\hat{c} = \lceil c/S \rceil$, then we know the range of $\hat{c}$ is $\hat{c} \in [\hat{c}_e^l, \hat{c}_e^u]$ with $\hat{c}_e^l = \lceil c_e(\hat{t}_e(\rho_i), \rho_i)/S \rceil$ and $\hat{c}_e^u = \lceil c_e(t_e^l, \rho_i)/S \rceil$.

Obviously it is true that if $\hat{c} = \hat{c}_e^u$, $t_i^e(\hat{c}_e^u) = t_e^l$ which is clearly the minimal required travel time. Now we prove that $t_i^e(\hat{c}) = c_e^{-1}(\hat{c}S, \rho_i)$ when $\hat{c}_e^l \le \hat{c} < \hat{c}_e^u$ is the minimal required travel time.

First, we show if $c_e^{-1}(\hat{c}S, \rho_i)$ exists and is a feasible travel time, i.e. $\hat{c}S \in [c_e(\hat{t}_e(\rho_i), \rho_i), c_e(t_e^l, \rho_i)]$ and hence $c_e^{-1}(\hat{c}S, \rho_i) \in [t_e^l, \hat{t}_e(\rho_i)]$, then it is the minimal one among travel times t in the range of $t \in [t_e^l, \hat{t}_e(\rho_i)]$ that meets $\hat{c} = \lceil c_e(t, \rho_i)/S \rceil$.

For any $t \in [t_e^l, c_e^{-1}(\hat{c}S, \rho_i))$, since $c_e(\cdot)$ is strictly decreasing and hence the inverse function $c_e^{-1}(\cdot)$ is also strictly decreasing, it holds that

$$c_e(t, \rho_i) > c_e(c_e^{-1}(\hat{c}S, \rho_i), \rho_i) = \hat{c}S,$$

implying that

$$c_e(t, \rho_i)/S > \hat{c}S/S = \hat{c}.$$

Since $\hat{c}$ is an integer and $\lceil \hat{c} \rceil = \hat{c}$, it clearly holds that

$$\lceil c_e(t, \rho_i)/S \rceil \ge \hat{c} + 1 > \hat{c},$$

proving that $c_e^{-1}(\hat{c}S, \rho_i)$ is the minimal required travel time.

Next, we show if $\hat{c}_e^l \le \hat{c} < \hat{c}_e^u$, the travel time $c_e^{-1}(\hat{c}S, \rho_i)$ must exist and be feasible. Considering $\hat{c} \le \hat{c}_e^u - 1 < \hat{c}_e^u$, we have

$$\hat{c} \le \hat{c}_e^u - 1 = \lceil c_e(t_e^l, \rho_i)/S \rceil - 1 < c_e(t_e^l, \rho_i)/S,$$

implying that $\hat{c}S < c_e(t_e^l, \rho_i)$. Considering $\hat{c} \ge \hat{c}_e^l$, we have

$$\hat{c} \ge \hat{c}_e^l = \lceil c_e(\hat{t}_e(\rho_i), \rho_i)/S \rceil \ge c_e(\hat{t}_e(\rho_i), \rho_i)/S,$$

implying that $\hat{c}S \ge c_e(\hat{t}_e(\rho_i), \rho_i)$. Thus, $\hat{c}S$ is in the range of

$$\hat{c}S \in \left[c_e(\hat{t}_e(\rho_i), \rho_i), c_e(t_e^l, \rho_i)\right).$$

Such a range must be non-empty because that the assumption of $\hat{c}_e^l \le \hat{c} < \hat{c}_e^u$, namely $\hat{c}_e^l < \hat{c}_e^u$, implies that $c_e(\hat{t}_e(\rho_i), \rho_i) < c_e(t_e^l, \rho_i)$. Since function $c_e(\cdot, \rho_i)$ is feasible in the range of $[t_e^l, \hat{t}_e(\rho_i)]$ and is strictly decreasing, the inverse function $c_e^{-1}(\cdot, \rho_i)$ must be strictly decreasing and exist in the range of $[c_e(\hat{t}_e(\rho_i), \rho_i), c_e(t_e^l, \rho_i)]$, and $c_e^{-1}(\hat{c}S, \rho_i)$ must be feasible that $c_e^{-1}(\hat{c}S, \rho_i) \in [t_e^l, \hat{t}_e(\rho_i)]$, which completes our proof. □

7.5 Proof to Lem. 7.2

Lemma 7.2. *With the quantization approach $\hat{c}_e(t, \rho_i) = \lceil c_e(t, \rho_i)/S \rceil$, for any solution $p = p_1 \cup p_2 \cup ... \cup p_K$ of our problem TREK, the cost of each path p_i must satisfy*

$$c(p_i) \le \hat{c}(p_i)S \le c(p) + nS, \forall i = 1, 2, ..., K,$$

where $c(\cdot)$ is the path cost before quantization and $\hat{c}(\cdot)$ is the path cost after quantization. Besides, the cost of path p must meet

$$c(p) \le \hat{c}(p)S \le c(p) + KnS.$$

Proof. Due to the quantization of $\hat{c}_e(t, \rho_i) = \lceil c_e(t, \rho_i)/S \rceil$, clearly we have

$$\frac{c_e(t, \rho_i)}{S} \le \hat{c}_e(t, \rho_i) < \frac{c_e(t, \rho_i)}{S} + 1,$$

namely

$$c_e(t, \rho_i) \le \hat{c}_e(t, \rho_i)S < c_e(t, \rho_i) + S.$$

Now considering a specific p_i, it holds that

$$\sum_{e \in p_i} c_e(t_i^e, \rho_i) \le \sum_{e \in p_i} \hat{c}_e(t_i^e, \rho_i)S < \sum_{e \in p_i} \left(c_e(t_i^e, \rho_i) + S\right),$$

implying that

$$c(p_i) \le \hat{c}(p_i)S \overset{(a)}{\le} c(p_i) + nS,$$

where inequality (a) holds since p_i is a simple path.

Moreover, due to that $p = p_1 \cup p_2 \cup ... \cup p_K$, obviously we have

$$c(p) \le \hat{c}(p)S \le c(p) + KnS.$$

□

7.6 Proof to Lem. 7.3

Lemma 7.3. *Any non-empty solution $p = p_1 \cup p_2 \cup ... \cup p_K$ returned by our test procedure (Algorithm 1) must satisfy*

$$OPT \le c(p) \le U + K(n+1)S = U + L\epsilon.$$

Proof. Since p is returned by the test procedure, it must be a feasible solution to TREK with all time window constraints satisfied (feasibility is guaranteed in each step of the DP approach). Then clearly that

$$\mathrm{OPT} \le c(p).$$

According to Lem. 7.1, for any edge $e \in p_i$, the quantization on the edge cost follows the following approach

$$\hat{c}_e(t_i^e, \rho_i) = \left\lceil \frac{c_e(t_i^e, \rho_i)}{S} \right\rceil.$$

Hence we can use Lem. 7.2 to obtain the inequality below

$$c(p) \le \hat{c}(p)S \overset{(a)}{\le} \hat{U}S \le \left(\frac{U}{S} + K(n+1)\right)S = U + K(n+1)S,$$

where inequality (a) holds because that p is returned by the test procedure where the loop in line 13 restricts that the quantized cost of p, i.e. $\hat{c}(p)$, is upper bounded by $\hat{U}$. □

7.7 Proof to Lem. 7.4

Lemma 7.4. *If $U \geq OPT$ and our problem TREK is solvable, then the test procedure (Algorithm 1) must return a non-empty solution $p = p_1 \cup p_2 \cup ... \cup p_K$. Moreover, it holds for p that*

$$c(p) \leq OPT + L\epsilon.$$

Proof. We denote the optimal solution to TREK by $p^* = p_1^* \cup p_2^* \cup ... \cup p_K^*$ with edge travel time $t_i^e(*)$ for $e \in p_i^*$. Define

$$\bar{t}_i^e \triangleq \min_{t \in [t_e^l, t_i^e(*)]} t : \hat{c}_e(t, \rho_i) = \hat{c}\left(t_i^e(*), \rho_i\right) = \left\lceil \frac{c(t_i^e(*), \rho_i)}{S} \right\rceil,$$

which is the minimal travel time that shares the same quantized cost compared to $t_i^e(*)$, or equivalently the calculated $t_i^e(\hat{c})$ in line 5 of Algorithm 1 when $\hat{c} = \lceil c(t_i^e(*), \rho_i)/S \rceil$. Then we can define a feasible solution $\bar{p}$ which has the same topology as p^* but assigns a travel time of $\bar{t}_i^e$ instead of $t_i^e(*)$ to the edge $e \in p_i, \forall i = 1, 2, ..., K$. The feasibility of $\bar{p}$ comes from that no edge travel time is increased compared to the optimal solution p^*. And obviously it holds that

$$\hat{c}(\bar{p}) = \hat{c}(p^*), \hat{c}(\bar{p}_i) = \hat{c}(p_i^*), \forall i = 1, 2, ..., K.$$

Considering the assumption of $U \geq \mathrm{OPT} = c(p^*)$, obviously that $U \geq c(p_i^*), \forall i = 1, 2, ..., K$ due to the non-negative property of edge cost. Further according to Lem. 7.2, for any $i \in 1, 2, ..., K$, we have

$$\hat{c}(\bar{p}_i) = \hat{c}(p_i^*) \leq \frac{c(p_i^*)}{S} + n \leq \left\lfloor \frac{c(p_i^*)}{S} \right\rfloor + n + 1 \leq \hat{U}.$$

Similarly we can prove that $\hat{c}(\bar{p}) = \hat{c}(p^*) \leq \hat{U}$. Therefore, $\bar{p}_i$ must be examined in the loop (line 8) of Algorithm 1, and hence $\bar{p}$ must be examined in the loop (line 13) of Algorithm 1. Due to that $\bar{p}$ is a feasible solution to TREK, the test procedure must return a non-empty solution p which may be different from $\bar{p}$.

Since p is the optimal solution to the test procedure, it must hold that $\hat{c}(p) \leq \hat{c}(\bar{p})$. According to Lem. 7.2, we have

$$c(p) \leq \hat{c}(p)S \leq \hat{c}(\bar{p})S = \hat{c}(p^*)S \leq c(p^*) + KnS \leq \mathrm{OPT} + L\epsilon.$$

□

7.8 Proof to Lem. 7.5

Lemma 7.5. *Algorithm 1 has a time complexity of $O(Knm(Kn + \frac{UKn}{L\epsilon})^2)$, assuming it takes a constant time to calculate $\hat{t}_e(\rho_i), c_e(t, \rho_i)$ and $c_e^{-1}(c, \rho_i)$ given an edge $e \in E$ and a truck load ρ_i.*

Proof. There are three loops in Algorithm 1. First loop of line 3 has a time complexity of $O(Km\hat{U})$. Second loop of line 8 has a time complexity of $O(Knm\hat{U}^2)$. Third loop of line 13 has a time complexity of $O(K\hat{U}^2)$. Overall, our algorithm 1 has a time complexity of $O(Knm\hat{U}^2) = O(Knm(Kn + \frac{UKn}{L\epsilon})^2)$. □

7.9 Proof to Lem. 7.6

Lemma 7.6. *If Test$(G, \vec{\sigma}, B, B, 1)$ returns NULL, OPT $\geq B$. Otherwise, if Test$(G, \vec{\sigma}, B, B, 1)$ returns a non-empty solution, OPT $\leq 2B$.*

Proof. According to Lem. 7.4, if $\mathrm{OPT} < B$, Test$(G, \vec{\sigma}, B, B, 1)$ must return a non-empty solution. Thus if the test returns NULL, it holds that $\mathrm{OPT} \geq B$.

According to Lem. 7.3, for the returned non-empty solution p of Test$(G, \vec{\sigma}, B, B, 1)$, we have

$$\mathrm{OPT} \leq U + L\epsilon = 2B.$$

□

7.10 Proof to Thm. 3.1

We prove our FPTAS in Algorithm 2 provides a $(1+\epsilon)$-approximation ratio in this section.

Proof. According to Lem. 7.6, in the binary search scheme of line 3, our Algorithm 2 always maintain a valid lower bound B_L and a valid upper bound $2B_U$ for OPT till the gap $B_U/B_L \leq 2$.

Therefore, in line 9, since $2B_U$ is a valid upper bound of OPT, according to Lem. 7.4, Test$(G, \vec{\sigma}, B_L, 2B_U, \epsilon)$ must return a non-empty solution p_{fptas} that satisfies

$$c(p_{\mathrm{fptas}}) \leq \mathrm{OPT} + B_L\epsilon \leq (1 + \epsilon)\mathrm{OPT}.$$

□

7.11 Proof to Thm. 4.1

In this section, we prove the equivalence of two formulations of problem TREK: formulation (2) and formulation (4).

Proof. Since both formulations share the same objective function, we only need to prove that a solution $\vec{x}_i, \vec{t}_i, i = 1, 2, ..., K$ is feasible in terms of formulation (2) if and only if it is feasible in terms of formulation (4),

If part. Suppose $\vec{x}_i, \vec{t}_i, i = 1, 2, ..., K$ satisfies the constraint (4b), we prove it must satisfy the constraint (2b).

For such a solution $\vec{x}_i, \vec{t}_i, i = 1, 2, ..., K$, given a specific $i \in 1, 2, ...K$, if $a_{i-1} \leq T_i^{\mathrm{out}}$, then we have

$$\max\left\{a_{i-1}, T_i^{\mathrm{out}}\right\} + \sum_{e \in E} x_i^e t_i^e = T_i^{\mathrm{out}} + \sum_{e \in E} x_i^e t_i^e.$$

Based on the constraint (4b) when $r = k = i$, clearly it holds that

$$\max\left\{a_{i-1}, T_i^{\mathrm{out}}\right\} + \sum_{e \in E} x_i^e t_i^e \leq T_i^{\mathrm{out}} + T_{i+1}^{\mathrm{in}} - T_i^{\mathrm{out}} = T_{i+1}^{\mathrm{in}},$$

which is the constraint (2b).

If $a_{i-1} > T_i^{\mathrm{out}}$, without loss of generality, we assume that $a_j > T_{j+1}^{\mathrm{out}}$ for $j = l, l+1, ..., i-1$ where $l \geq 1$ and $a_{l-1} \leq T_l^{\mathrm{out}}$. Then it holds that

$$\begin{aligned}
\max\{a_{i-1}, T_i^{\mathrm{out}}\} + \sum_{e \in E} x_i^e t_i^e &= a_{i-1} + \sum_{e \in E} x_i^e t_i^e \\
&= a_{i-2} + \sum_{e \in E} x_{i-1}^e t_{i-1}^e + \sum_{e \in E} x_i^e t_i^e \\
&= ... \\
&= a_l + \sum_{j=l+1}^{i} \sum_{e \in E} x_j^e t_j^e \\
&= T_l^{\mathrm{out}} + \sum_{j=l}^{i} \sum_{e \in E} x_j^e t_j^e.
\end{aligned}$$

Based on the constraint (4b) when $r = i, k = l$, namely the constraint of $\sum_{j=l}^{i}\sum_{e\in E} x_j^e t_j^e \le T_{i+1}^{\text{in}} - T_l^{\text{out}}$, obviously it holds that

$$\max\left\{a_{i-1}, T_i^{\text{out}}\right\} + \sum_{e\in E} x_i^e t_i^e \le T_l^{\text{out}} + T_{i+1}^{\text{in}} - T_l^{\text{out}} = T_{i+1}^{\text{in}},$$

which is the constraint (2b).

Only if part. Assume $\vec{x}_i, \vec{t}_i, i = 1, 2, ..., K$ satisfies the constraint (2b), we prove it must satisfy the constraint (4b).

Due to our assumption that $a_i = \max\{a_{i-1}, T_i^{\text{out}}\} + \sum_{e\in E} x_i^e t_i^e$ and $a_i \le T_{i+1}^{\text{in}}$, considering the two inequality $\max\{a_{i-1}, T_i^{\text{out}}\} \ge a_{i-1}$ and $\max\{a_{i-1}, T_i^{\text{out}}\} \ge T_i^{\text{out}}$, for any $r = 1, 2, ..., K$ and any $k = 1, 2, ..., r$, we have

$$\begin{aligned}
\sum_{e\in E} x_r^e t_r^e &\le T_{r+1}^{\text{in}} - \max\left\{a_{r-1}, T_r^{\text{out}}\right\},\\
&\le T_{r+1}^{\text{in}} - a_{r-1}\\
&= T_{r+1}^{\text{in}} - \max\left\{a_{r-2}, T_{r-1}^{\text{out}}\right\} - \sum_{e\in E} x_{r-1}^e t_{r-1}^e\\
&\le T_{r+1}^{\text{in}} - \sum_{e\in E} x_{r-1}^e t_{r-1}^e - a_{r-2}\\
&= T_{r+1}^{\text{in}} - \sum_{j=r-2}^{r-1}\sum_{e\in E} x_j^e t_j^e - \max\left\{a_{r-3}, T_{r-2}^{\text{out}}\right\}\\
&\le \ ...\\
&= T_{r+1}^{\text{in}} - \sum_{j=k}^{r-1}\sum_{e\in E} x_j^e t_j^e - \max\left\{a_{k-1}, T_k^{\text{out}}\right\}\\
&\le T_{r+1}^{\text{in}} - \sum_{j=k}^{r-1}\sum_{e\in E} x_j^e t_j^e - T_k^{\text{out}},
\end{aligned}$$

namely it holds that

$$\sum_{j=k}^{r}\sum_{e\in E} x_j^e t_j^e \le T_{r+1}^{\text{in}} - T_k^{\text{out}},$$

which is the constraint (4b). □

7.12 Proof to Thm. 4.2

We prove our Thm. 4.2 in this section, namely

$$D(\vec{\lambda}) = -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r(T_{r+1}^{\text{in}} - T_k^{\text{out}}) + \sum_{i=1}^{K}\sum_{e\in p(\mu_i)} w_i^e(\mu_i).$$

Proof.

$$\begin{aligned}
D(\vec{\lambda}) &= \min_{\vec{x},\vec{t}} L(\vec{x},\vec{t},\vec{\lambda})\\
&= -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r(T_{r+1}^{\text{in}} - T_k^{\text{out}})\\
&\quad + \min_{\vec{x},\vec{t}}\sum_{i=1}^{K}\sum_{e\in E} x_i^e\cdot\left[c_e(t_i^e,\rho_i) + \mu_i t_i^e\right]\\
&\overset{(a)}{=} -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r(T_{r+1}^{\text{in}} - T_k^{\text{out}})\\
&\quad + \sum_{i=1}^{K}\min_{\vec{x}_i}\min_{\vec{t}_i}\sum_{e\in E} x_i^e\cdot\left[c_e(t_i^e,\rho_i) + \mu_i t_i^e\right]\\
&\overset{(b)}{=} -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r(T_{r+1}^{\text{in}} - T_k^{\text{out}})\\
&\quad + \sum_{i=1}^{K}\min_{\vec{x}_i}\sum_{e\in E} x_i^e\cdot\min_{t_e^l\le t_i^e\le \hat{t}_e(\rho_i)}\left[c_e(t_i^e,\rho_i) + \mu_i t_i^e\right]\\
&\overset{(c)}{=} -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r(T_{r+1}^{\text{in}} - T_k^{\text{out}})\\
&\quad + \sum_{i=1}^{K}\min_{\vec{x}_i}\sum_{e\in E} x_i^e\cdot\left[c_e(t_i^e(\mu_i),\rho_i) + \mu_i t_i^e(\mu_i)\right]\\
&\overset{(d)}{=} -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r(T_{r+1}^{\text{in}} - T_k^{\text{out}}) + \sum_{i=1}^{K}\min_{\vec{x}_i}\sum_{e\in E} x_i^e\cdot w_i^e(\mu_i)\\
&\overset{(e)}{=} -\sum_{r=1}^{K}\sum_{k=1}^{r}\lambda_k^r(T_{r+1}^{\text{in}} - T_k^{\text{out}}) + \sum_{i=1}^{K}\sum_{e\in p(\mu_i)} w_i^e(\mu_i),
\end{aligned}$$

where equality (a) holds because no coupled constraints exist for different tasks and further no coupled constraints exist between $\vec{x}_i$ and $\vec{t}_i$ given any $i = 1, 2, ..., K$. Equality (b) is true because no coupled constraints exist for the travel time on different edges given any $i = 1, 2, ..., K$. Equality (c) holds due to our definition of $t_i^e(\mu_i)$. Similarly, equality (d) is true due to our definition of $w_i^e(\mu_i)$, and equality (e) holds due to the definition of $p(\mu_i)$. □

7.13 Proof to Thm. 4.4

Proof. First we prove the solution $\{p(\mu_i), i = 1, 2, ..., K\}$ that meets condition (10) must be a feasible solution to TREK.

Obviously for each task $\tau_i, i = 1, 2, ..., K$, such a solution defines a path from σ_i to σ_{i+1} to fulfill τ_i. Hence in order to prove its feasibility, we only need to prove it satisfies all the time window constraint (4b). For any $r = 1, 2, ..., K$ and $k = 1, 2, ..., r$, if the corresponding λ_k^r is strictly positive, clearly based on the condition (10) we have

$$T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r}\delta(\mu_j) = 0,$$

implying that the solution $\{p(\mu_i), i = 1, 2, ..., K\}$ meets the constraint (4b). For any $r = 1, 2, ..., K$ and $k = 1, 2, ..., r$, if the corresponding $\lambda_k^r = 0$, based on the condition (10) we have

$$\max\left\{T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r}\delta(\mu_j), 0\right\} = 0,$$

namely it holds that

$$T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r}\delta(\mu_j) \le 0,$$

also proving that the solution $\{p(\mu_i), i = 1, 2, ..., K\}$ meets the constraint (4b). Overall, the solution $\{p(\mu_i), i = 1, 2, ..., K\}$ must be feasible, and hence provides an upper bound for the optimal cost

OPT of problem TREK, i.e.

$$\mathcal{P}(\vec{\lambda}) = \sum_{i=1}^{K} \sum_{e \in p(\mu_i)} c_e(t_i^e(\mu_i), \rho_i) \geq \text{OPT}. \tag{16}$$

Next we look at the dual function value of the solution $\{p(\mu_i), i = 1, 2, ..., K\}$.

$$\begin{aligned} D(\vec{\lambda}) &= -\sum_{r=1}^{K}\sum_{k=1}^{r} \lambda_k^r (T_{r+1}^{\text{in}} - T_k^{\text{out}}) \\ &+ \sum_{i=1}^{K} \sum_{e \in p(\mu_i)} \left[c_e(t_i^e(\mu_i), \rho_i) + \mu_i t_i^e(\mu_i) \right] \\ &= -\sum_{r=1}^{K}\sum_{k=1}^{r} \lambda_k^r (T_{r+1}^{\text{in}} - T_k^{\text{out}}) \\ &+ \sum_{i=1}^{K} \mu_i \delta(\mu_i) + \sum_{i=1}^{K} \sum_{e \in p(\mu_i)} c_e(t_i^e(\mu_i), \rho_i) \\ &= \sum_{r=1}^{K}\sum_{k=1}^{r} \lambda_k^r \cdot \left(T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\mu_j) \right) \\ &+ \sum_{i=1}^{K} \sum_{e \in p(\mu_i)} c_e\left(t_i^e(\mu_i), \rho_i\right). \end{aligned}$$

Since the condition (10) implies that

$$\lambda_k^r \cdot \left(T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\mu_j) \right) = 0,$$
$$\forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r,$$

the dual function is equal to

$$D(\vec{\lambda}) = \sum_{i=1}^{K} \sum_{e \in p(\mu_i)} c_e(t_i^e(\mu_i), \rho_i).$$

According to the weak duality, any dual function value is a lower bound of the optimal cost OPT, namely it holds that

$$D(\vec{\lambda}) = \sum_{i=1}^{K} \sum_{e \in p(\mu_i)} c_e(t_i^e(\mu_i), \rho_i) \leq \text{OPT}. \tag{17}$$

Comparing the inequality (16) and the inequality (17), obviously that the solution $\{p(\mu_i), i = 1, 2, ..., K\}$ must be optimal to problem TREK. □

7.14 Proof to Thm. 4.5

We prove our Thm. 4.5 in this section, namely prove the performance bound of SPEED in Algorithm 3.

Proof. If TREK is solvable, the solution where each task is fulfilled by the fastest path must be feasible, which must be examined by SPEED due to that initially we set all dual variables to be a large enough $\lambda_{\max}$. Therefore, SPEED must return a feasible solution.

Result for case 1 holds directly due to Thm. 4.4.

For p returned in case 2, let us denote the dual variables defining p as $\bar{\lambda}_k^r, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r$, and denote $\max_{\forall r,k} |\dot{\bar{\lambda}}_k^r|$ as $\dot{\bar{\lambda}}$, namely $|\dot{\bar{\lambda}}_k^r| \leq \dot{\bar{\lambda}}, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r$.

Because p is feasible, we have

$$\mathcal{P}(\vec{\bar{\lambda}}) = \sum_{i=1}^{K} \sum_{e \in p(\bar{\mu}_i)} c_e\left(t_i^e(\bar{\mu}_i), \rho_i\right) \geq \text{OPT}. \tag{18}$$

Next we look at the dual function value of p.

$$\begin{aligned} D(\vec{\bar{\lambda}}) &= -\sum_{r=1}^{K}\sum_{k=1}^{r} \bar{\lambda}_k^r (T_{r+1}^{\text{in}} - T_k^{\text{out}}) \\ &+ \sum_{i=1}^{K} \bar{\mu}_i \delta(\bar{\mu}_i) + \sum_{i=1}^{K} \sum_{e \in p(\bar{\mu}_i)} c_e\left(t_i^e(\bar{\mu}_i), \rho_i\right) \\ &= \sum_{r=1}^{K}\sum_{k=1}^{r} \bar{\lambda}_k^r \cdot \left(T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\bar{\mu}_j) \right) \\ &+ \sum_{i=1}^{K} \sum_{e \in p(\bar{\mu}_i)} c_e\left(t_i^e(\bar{\mu}_i), \rho_i\right). \end{aligned}$$

Since p is feasible meeting all constraints in our formulation (4):

$$\sum_{j=k}^{r} \delta(\bar{\mu}_j) \leq T_{r+1}^{\text{in}} - T_k^{\text{out}}, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r,$$

it holds that

$$\left[T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\bar{\mu}_j) \right]_{\bar{\lambda}_k^r}^{+} \leq 0.$$

Further due to $\phi(\bar{\lambda}_k^r) > 0$, it holds that $\dot{\bar{\lambda}}_k^r \leq 0$, implying that $-\dot{\bar{\lambda}} \leq \dot{\bar{\lambda}}_k^r \leq 0$, and we have

$$\left[T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\bar{\mu}_j) \right]_{\bar{\lambda}_k^r}^{+} \geq -\dot{\bar{\lambda}}/\phi(\bar{\lambda}_k^r).$$

Considering definition of $[T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\bar{\mu}_j)]_{\bar{\lambda}_k^r}^{+}$, we have

$$\begin{aligned} \bar{\lambda}_k^r \left(T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\bar{\mu}_j) \right) &= \bar{\lambda}_k^r \left[T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\bar{\mu}_j) \right]_{\bar{\lambda}_k^r}^{+} \\ &\geq -\dot{\bar{\lambda}} \cdot \bar{\lambda}_k^r / \phi(\bar{\lambda}_k^r). \end{aligned}$$

According to the weak duality, it holds that

$$D(\vec{\bar{\lambda}}) \leq \text{OPT}. \tag{19}$$

Comparing the inequality (18) and the inequality (19), we have:

$$\begin{aligned} \mathcal{P}(\vec{\bar{\lambda}}) - \text{OPT} &\leq \mathcal{P}(\vec{\bar{\lambda}}) - D(\vec{\bar{\lambda}}) \\ &= -\sum_{r=1}^{K}\sum_{k=1}^{r} \bar{\lambda}_k^r \cdot \left(T_k^{\text{out}} - T_{r+1}^{\text{in}} + \sum_{j=k}^{r} \delta(\bar{\mu}_j) \right) \\ &\leq K^2 \cdot \dot{\bar{\lambda}} \cdot \max_{\forall r,k} \left\{ \bar{\lambda}_k^r / \phi(\bar{\lambda}_k^r) \right\} \\ &= K^2 \cdot \max_{\forall r,k} \left| \dot{\bar{\lambda}}_k^r \right| \cdot \max_{\forall r,k} \left\{ \bar{\lambda}_k^r / \phi(\bar{\lambda}_k^r) \right\}. \end{aligned}$$

□