Energy-Efficient Timely Truck Transportation for Geographically-Dispersed Tasks

Qingyu Liu, Haibo Zeng, Minghua Chen

***Abstract*—We consider a common truck operation 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 ones on single task. We first prove that our problem is NP-hard. Moreover, it is uniquely challenging to solve our problem, as we further show that optimizing task execution times 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 and efficient heuristic algorithm, called SPEED (Sub-gradient-based Price-driven Energy-Efficient Delivery). We characterize sufficient conditions under which SPEED generates an optimal solution, and derive an optimality gap for SPEED when the conditions are not satisfied. We evaluate the practical performances of our solutions using real-world traces over the US national highway. We observe that our solutions can save up to 22% fuel as compared to the fastest-/shortest- path baselines, and up to 10% fuel than a conceivable alternative generalized from the state-of-the-art single-task algorithm. The fuel saving is robust to the number of tasks to be fulfilled. Simulations also show that our algorithms always obtain close-to-optimal solutions and meet time window constraints for all feasible problem instances. In comparison, the conceivable alternative fails to meet time window constraints for up to 45% of the instances.**

***Index Terms*—Energy-efficient transportation, timely delivery, task execution times optimization, path planning, speed planning.**

I. Introduction

The US trucking industry is critical and drives the US national economy. In 2016, it hauls 70.6% (up to 10.42 billion tons) of all freight tonnage [2], and collects $676 billion in gross freight revenues, accounting for 79.8% of the freight bill [2]. This number would rank 19th 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 [3]. This alerting observation, together with that fuel consumption accounts for the largest truck operating cost factor (26%) [3], 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 long-haul truck drives across a national highway to fulfill multiple 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. Pickup and delivery time window constraints are common in the trucking industry. For instance, mobile applications like uShip[1] and Uber Freight[2] nowadays provide lots of freight transportation requests for truck operators, which are often associated with earliest and latest pickup and delivery time requirements. Our design space includes path planning, speed planning, and task execution times optimization. Path planning and speed planning are two well-recognized approaches to effectively save fuel[3], as studied in the single-task setting [4], [5].

In addition, we consider a new design space of task execution times optimization for saving fuel, which differentiates our study under the multi-task setting from existing ones under the single-task setting. In the single-task setting, the task execution time budget (also known as the deadline constraint) is a fixed input [4], [5]. However, in the multi-task setting it is necessary to optimally coordinate execution times for individual tasks by jointly considering their time window constraints. To be specific, for individual tasks, a longer execution time budget can save fuel due to a bigger design space of path planning and speed planning. However, we cannot allocate an arbitrarily long execution time budget for each task, because it is crucial to catch time window constraints to avoid substantial penalty of violation. Further, due to that different tasks can have temporally overlapped time window constraints, although increasing the execution time budget of one task can save its fuel, overall it may consumes more fuel to fulfill all tasks because of the decreasing execution time budgets of other tasks. In conclusion, task execution times optimization is a must for effectively saving fuel in the multi-task setting.

We remark that optimizing task execution times is challenging, as it is a non-convex puzzle as proven later in Sec. III-C. Hence existing single-task studies, e.g., [4], [5], cannot be adapted to solve our multi-task problem directly, due to the lack of an efficient task execution times optimization scheme.

Tab. I compares our work with existing studies on energy-efficient timely truck operations. We present details of related work later in Sec. II. In conclusion, we are the first to

Corresponding author: *Qingyu Liu*, *Haibo Zeng*, and *Minghua Chen*.

Part of this work has been presented at the ACM International Conference on Future Energy Systems, Karlsruhe, Germany, June 12 - 15, 2018 [1]. A major part of this work was done during Qingyu's visit to the Department of Information Engineering, The Chinese University of Hong Kong, in 2017.

Q. Liu and H. Zeng are with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, USA (e-mail: {qyliu14, hbzeng}@vt.edu). M. Chen is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: minghua@ie.cuhk.edu.hk).

[1] uShip, https://www.uship.com/

[2] Uber Freight, https://freight.uber.com/

[3] US Department of Energy, https://afdc.energy.gov/data/

TABLE I: Comparison of our work and existing energy-efficient timely truck transportation studies.

Studied problem		RSP [6]–[8]	PASO [4], [5]	[9], [10]	[11], [12]	Our work
Setting	Fulfill multiple tasks	✗	✗	✗	✗	✓
Design space	Path planning	✓	✓	✗	✓	✓
	Speed planning	✗	✓	✓	✗	✓
	Task execution times optimization	✗	✗	✗	✗	✓
Constraint	Time window constraints	✓	✓	✗	✗	✓

study the problem of minimizing the fuel consumption for a long-haul heavy truck to fulfill multiple transportation tasks subject to task pickup and delivery time window constraints. Solving our problem requires to simultaneously optimize task execution times, path planning, and speed planning. We make the following specific **contributions** in this paper.

▷ We prove that our problem is NP-hard. We show that optimizing task execution times is a new design space for saving fuel introduced by the multi-task setting, and it is a non-convex puzzle. Thus it is uniquely challenging to solve our problem under the multi-task setting, as compared to solving existing ones under the single-task setting.

▷ We design a Fully-Polynomial-Time Approximation Scheme (FPTAS). Theoretically it always achieves a $(1+\epsilon)$-approximate solution with a time complexity polynomial in problem inputs and $1/\epsilon$, for any user-defined $\epsilon > 0$. Practically it is suitable for solving small-scale problem instances.

▷ We develop an efficient heuristic SPEED (Sub-gradient-based Price-driven Energy-Efficient Delivery), based on dual sub-gradient updates of task execution times. We characterize conditions under which SPEED generates an optimal solution, and further derive a performance gap comparing the solution of SPEED with the optimal when the conditions are not satisfied. Practically SPEED can obtain close-to-optimal solutions quickly for large-scale problem instances.

▷ We evaluate our solutions using real-world traces over the US national highway system. Our solutions save up to 22% fuel as compared to the fastest-/shortest- path baselines, and up to 10% fuel as compared to a conceivable alternative generalized from the state-of-the-art single-task algorithm. Besides, our algorithms always obtain close-to-optimal solutions and meet the time window constraints for all feasible problem instances. In comparison, the conceivable alternative fails to meet one or more time window constraints for up to 45% of the instances. We also observe that the fuel saving of our solutions is independent to the number of tasks.

II. Related Work

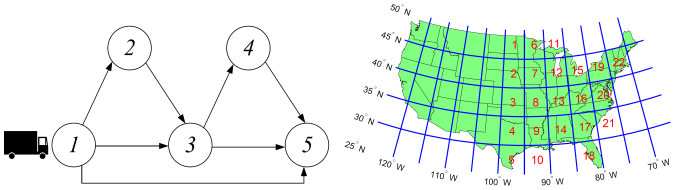
Restricted Shortest Path (RSP): RSP requires to find a path such that the path fuel consumption is minimized while the path travel time is within a deadline constraint. RSP is NP-hard [6] with heuristic algorithms [8] and FPTASes [6], [7] designed. RSP optimizes fuel consumption with only path planning involved, assuming fixed speeds. Moreover, RSP considers only single task, where the task execution time budget is given as a fixed deadline constraint, and thus no task execution times optimization is involved. Therefore, existing results on RSP cannot be directly applied to the multi-task setting where the challenging design space of task execution times optimization is a must for saving fuel.

PAth selection and Speed Optimization (PASO): PASO [4], [5] generalizes RSP with speed planning taken into account. Deng *et al.* [4], [5] develop both an FPTAS and a heuristic for PASO. Similar to RSP, PASO considers only single task. Therefore, existing results on PASO cannot be easily generalized to our scenario of multiple tasks, as solving the multi-task version of the problem involves addressing a uniquely challenging puzzle of task execution times optimization.

Pickup and Delivery Problem with Time Window (PDPTW): PDPTW is a generalization of the Traveling Salesman Problem (TSP) [13] and Vehicle Routing Problem (VRP) [14]. PDPTW [15]–[17] and its extensions [18], [19] minimize fuel consumption for a set of vehicles to timely fulfill multiple tasks. Note that PDPTW and our problem are fundamentally different: (i) we optimize path planning and speed planning for fulfilling each task, while PDPTW assumes fixed paths (e.g., shortest paths) and fixed road driving speeds for fulfilling each task; (ii) the main challenge of PDPTW is to optimize the execution order of tasks, while our setting assume the execution order of tasks is given; (iii) PDPTW is proven to be APX-hard [20] and thus no PTAS exists unless P = NP, while our problem admits an FPTAS (see our Sec. IV).

Other studies: Sahlholm *et al.* [21] present an approach to estimate road grade. Such grade information can be used by assistance systems to optimize driving speed to save fuel, as studied in [9], [10]. Studies [9], [10] both focus on fulfilling one task without time window constraint, assuming that the path is given and hence no path planning is involved. Boriboonsomsin *et al.* [11] present an eco-routing navigation system that determines the most fuel-economic path. Scora *et al.* [12] analyze the tradeoff between the amount of fuel savings and the added travel time relative to the fastest path. Both studies [11], [12] are under the single-task setting without time window constraint, assuming fixed road driving speeds and hence no speed planning is involved. Alam *et al.* [22] observe that improved fuel-efficiency can be obtained by maintaining the platoon of trucks throughout a hill, motivating subsequent studies, e.g., [23], [24], which focus on developing control strategies for truck platooning for saving fuel.

Overall to our best knowledge, we are the first to study the problem of minimizing fuel consumption for a long-haul heavy truck to fulfill multiple transportation tasks under task pickup and delivery time window constraints. Solving the problem requires us to simultaneously optimize task execution times, path planning, and speed planning. As compared to existing studies under the single-task setting, task execution times optimization is a new critical design space for saving fuel for our problem that is under the multi-task setting.



(a) System model: an example of a truck driving in a highway network. (b) Simulated US highway network that is partitioned into 22 regions.

Fig. 1: A truck timely fulfills multiple transportation tasks in a national highway system, with fuel consumption minimized.

III. Preliminary

A. System Model

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

We consider the scenario where one long-haul heavy-duty truck travels across G to fulfill K transportation tasks denoted by $\vec{\tau} = \{\tau_i, i = 1, 2, ..., K\}$, as illustrated in Fig. 1a. We make the following assumptions:

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

We model $\vec{\tau}$ by a sequence of nodes $\vec{\sigma} = \{\sigma_i \in V : i = 1, 2, ..., K+1\}$. For each task $\tau_i \in \vec{\tau}$, it requires the truck to pick up cargoes with a mass of $\rho_i > 0$ at the node $\sigma_i \in \vec{\sigma}$ within a pickup time window, and deliver them to the node $\sigma_{i+1} \in \vec{\sigma}$ within a delivery time window. The time window constraints are modeled by an earliest leaving time requirement $T_i^{\text{out}} \geq 0$ and a latest arrival time requirement $T_i^{\text{in}} > 0$ associated with each node $\sigma_i \in \vec{\sigma}$:

1) T_i^{out} defines the earliest time when the truck can leave σ_i. The truck cannot leave σ_i until the time T_i^{out} to finish both the delivery of the task τ_{i-1} and the pickup of the task τ_i;
2) T_i^{in} defines the latest time when the truck can arrive at σ_i. The truck must arrive at σ_i no later than the time T_i^{in} to finish both the delivery of the task τ_{i-1} and the pickup of the task τ_i.

The truck fuel consumption depends on many factors [25]. Like [4], [5], in this paper we ignore the fuel consumption incurred by acceleration and deceleration of a truck. This is because (i) when driving inside a road segment with homogeneous environmental conditions, as discussed in [26], [27] and proven by [5, Lem. 1], driving at a constant-speed is most fuel-economic; (ii) although a truck may accelerate or decelerate when it switches between different road segments, in our setting of long-haul truck transportation across a national highway network, the fuel consumption of acceleration/deceleration of switching can be fairly ignored as compared to that of traversing the road segment following a constant-speed (e.g., as shown in [28], a truck can accelerate from 0mph to 31mph in just 3% of the average length of road segments in our simulations over the US national highway. We empirically verify this observation in detail later in Sec. V-B by simulations using real-world traces). With the above justification, ignoring truck acceleration and deceleration allows us to simplify the formulation of our problem without incurring a significant optimality loss. Now given a road segment, it is reasonable to assume that the truck fuel consumption rate is a function of the cargo mass and the driving speed [4], [5].

We define $f_e(r_e, \rho) : [r_e^l, r_e^u] \times \mathbb{R}^+$ as the fuel consumption rate for the truck to pass a road segment $e \in E$ following a constant-speed of r_e with a load of ρ. Same to the assumption made by [4], [5] and verified by both physical laws and comprehensive simulations using real-world data, we assume $f_e(r_e, \rho)$ to be a strictly convex function with r_e over the interval $[r_e^l, r_e^u]$, given a truck load $\rho \geq 0$.

With the fuel consumption rate $f_e(r_e, \rho)$, we can define the following function $c_e(t_e, \rho)$ which calculates the truck fuel consumption of passing e with load ρ and travel time t_e

$$c_e(t_e, \rho) = t_e \cdot f_e(D_e/t_e, \rho). \tag{1}$$

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

B. Problem Formulation

In this paper there are two kinds of design variables: binary variable x_i^e defines 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 represents the specific travel time for the truck to pass edge e to fulfill the task τ_i.

By vectoring variables as $\vec{x}_i = \{x_i^e : e \in E\}$ and $\vec{t}_i = \{t_i^e : e \in E\}$, our problem Multi-task Energy-Efficient Trucking, denoted by MEET, can be formulated as

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

$$\text{s.t.} \quad a_i = \max\{a_{i-1}, T_i^{\text{out}}\} + \sum_{e \in E} x_i^e t_i^e \leq T_{i+1}^{\text{in}},$$
$$a_0 = 0, \ \forall i = 1, 2, ..., K, \tag{2b}$$

where the set $\mathcal{T}_i$ defines the possible road travel time, i.e.,

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

and $\mathcal{X}_i$ is the set of all paths from σ_i to σ_{i+1}, i.e.,

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

where $1_{\{\cdot\}}$ is the indicator function, $\text{in}(v) = \{(u,v) : (u,v) \in E\}$ is the set of incoming edges of node $v \in V$, $\text{out}(v) = \{(v,u) : (v,u) \in E\}$ is the set of outgoing edges of $v \in V$.

In the formulation in (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 it arrives at σ_i before the earliest allowed leaving time T_i^{out}. Objective (2a) minimizes the total fuel consumption for fulfilling the task sequence $\vec{\tau}$.

In the paper we denote a solution to MEET as

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

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 with each edge $e \in p_i$ assigned a specific travel time $t_i^e \in [t_e^l, \hat{t}_e(\rho_i)]$.

We remark that speed planning of each road segment $e \in E$ in MEET is subject to a minimum speed limit (r_e^l) and a maximum speed limit (r_e^u) both of which are fixed constants. We focus on fixed limits instead of variable ones which can consider the dynamic traffic condition due to the following concerns: (i) we consider long-haul transportation, where trucks mostly run on interstate highways. The operation thus is less affected by the dynamic traffic condition in cities; (ii) we start with the fundamental problem MEET which has not been studied in the literature, as shown in Tab. I; (iii) the maximum traveling speed (r_e^u) in MEET gives a first-order modeling for the road congestion. We leave it as a future direction to study trucking problems considering dynamic traffic conditions, and we introduce our preliminary results in [29], [30].

An illustrative example of MEET is introduced in Tab. II with the highway network being Fig. 1a, where there are two transportation tasks τ_1 and τ_2. It is clear that there are two paths to fulfill τ_1: one is the path $\langle 1, 3\rangle$ with a travel time of 1 and a cost[4] of 3, while the other is the path $\langle 1, 2, 3\rangle$ with a travel time of 2 and a cost of 2. Similarly, there are two paths to fulfill τ_2: one is the path $\langle 3, 5\rangle$ with a travel time of 1 and a cost of 4, while the other is the path $\langle 3, 4, 5\rangle$ with a travel time of 2 and a cost of 2. The most fuel-economic solution meeting time window constraints is to follow the path $\langle 1, 3, 4, 5\rangle$, with a total fuel consumption of 5.

[4]We interchangeably use fuel consumption and cost in this paper.

TABLE II: An illustrative example of problem MEET based on Fig. 1a, assuming $t_e^l = t_e^u = 1$ and $c_e^l = c_e^u$ for all $e \in E$.

e	$(1,2)$	$(2,3)$	$(1,3)$	$(3,4)$	$(4,5)$	$(3,5)$	$(1,5)$
c_e	1	1	3	1	1	4	8
i	$\sigma_1 = 1$		$\sigma_2 = 3$		$\sigma_3 = 5$		
T_i^{out}	$T_1^{\text{out}} = 0$		$T_2^{\text{out}} = 0$		$T_3^{\text{out}} = 0$		
T_i^{in}	$T_1^{\text{in}} = 0$		$T_2^{\text{in}} = 2$		$T_3^{\text{in}} = 3$		

C. It is Challenging to Solve MEET

First, solving MEET is challenging since MEET is a NP-hard problem: under a single-task setting assuming fixed road driving speeds, MEET requires the truck to travel from a source to a destination to minimize the fuel consumption, with the travel time bounded above by a task execution time budget, which is exactly the problem RSP. Since RSP has been proven to be NP-hard [6], [7], MEET is NP-hard.

Theorem 1. *MEET is NP-hard.*

Proof: MEET is NP-hard because that RSP, which is a special case of MEET, is NP-hard [6], [7]. ∎

Second, solving MEET is challenging because task execution times optimization, which is the new design space for saving fuel and differentiates the multi-task problem MEET from existing single-task ones, is a non-convex puzzle: we denote $C_i(T_i)$ as the minimal fuel consumption to fulfill the single task τ_i with the travel time bounded above by a task execution time budget of T_i. Then MEET can also be formulated as follows with an execution time budget T_i allocated to τ_i for each $i = 1, ..., K$:

$$\text{obj:} \quad \min_{\vec{x}_i \in \mathcal{X}_i, \vec{t}_i \in \mathcal{T}_i} \sum_{i=1}^{K} C_i(T_i) \tag{3a}$$
$$\text{s.t.} \quad a_i = \max\{a_{i-1}, T_i^{\text{out}}\} + T_i \leq T_{i+1}^{\text{in}},$$
$$a_0 = 0, \ \forall i = 1, 2, ..., K, \tag{3b}$$
$$\sum_{e \in E} x_i^e t_i^e \leq T_i, \ \forall i = 1, 2, ..., K. \tag{3c}$$

If we can find the optimal task execution times allocation $\{T_i, i = 1, 2, ...K\}$, we can run existing single-task algorithm (e.g., the one from by [4]) K times independently and obtain a high-quality solution. But we argue that it is hard to optimize task execution times, because $C_i(T_i)$ is non-convex with T_i.

Take the network in Fig. 1a as an example and consider one task where source is node 1 and destination is node 5. Edge travel time and edge fuel consumption are the same as defined in Tab. II. It is clear that in Fig. 1a there are 5 paths from source to destination. By enumerating them, we have $C(T) = 4$ when $T = 4$ following the path $\langle 1, 2, 3, 4, 5\rangle$; $C(T) = 5$ when $T = 3$ following the path $\langle 1, 3, 4, 5\rangle$; $C(T) = 7$ when $T = 2$ following the path $\langle 1, 3, 5\rangle$; and $C(T) = 8$ when $T = 1$ following the path $\langle 1, 5\rangle$. Then it is clear that the minimal fuel consumption $C(T)$ is neither convex nor concave in T.

Note that as optimizing task execution times is a new design space introduced by the multi-task setting and is non-convex, results from existing single-task studies, e.g., [4], [5], cannot be adapted to solve our multi-task MEET directly, due to the lack of an efficient task execution times optimization scheme.

IV. ALGORITHMS FOR MEET

In this section we develop both an FPTAS and a heuristic for MEET. The FPTAS optimizes task execution times by intelligently enumerating possible results without incurring excessive complexity, and then selecting the best. It obtains an $(1+\epsilon)$-approximate solution for any user-defined $\epsilon > 0$. The heuristic, called SPEED, optimizes task execution times by iteratively allocating execution time budgets for individual tasks towards the optimal, by following the sub-gradient of the Lagrangian dual relaxation of MEET. We characterize conditions under which SPEED generates an optimal solution, and derive a performance gap comparing the solution of SPEED with the optimal when the conditions are not satisfied.

A. An FPTAS for MEET

We observe that MEET has optimal substructures, and hence can be solved by Dynamic Programming (DP). With the rounding and scaling technique which has been used to develop DP-based FPTASes for problems RSP [6]–[8] and PASO [4], [5], we can design an FPTAS for MEET.

We divide MEET to two sub-problems: (i) in the sub-problem 1 for each single task τ_i, we enumerate its cost-bounded minimum-travel-time path for all possible cost of independently fulfilling τ_i; (ii) then in the sub-problem 2, we select exactly one solution per task, with the combined solution satisfying all time window constraints and minimizing total cost. We remark that existing single-task problems, e.g., RSP [6]–[8] and PASO [4], [5], only solve sub-problem 1; sub-problem 2 is unique for the multi-task problem MEET, and it requires to optimally coordinate execution times for tasks based on input time window constraints and results of the sub-problem 1. We note that both sub-problems have optimal substructures, and thus both can be solved by DP.

Although the DP approach can obtain high-quality solutions, it has an exponential time complexity. With the rounding and scaling technique that has been widely used by existing studies to develop DP-based FPTASes, e.g., [4]–[7], [31], [32], we can design an FPTAS to figure out $(1+\epsilon)$-approximate solutions to MEET in a polynomial time, for any $\epsilon > 0$. Considering that (i) the used rounding and scaling technique is standard, and (ii) our FPTAS suffers from an unacceptably large running time to obtain solutions practically (more than one hour even for a small instance in our simulations over real-world traces), we refer the details of our FPTAS to Part A of our supplementary materials, and focus on introducing an efficient heuristic to quickly obtain close-to-optimal solutions of MEET in the following.

B. The Lagrangian Dual Relaxation of MEET

We first give a new formulation of MEET in (4)

$$\text{obj:} \quad \min_{x_i \in X_i, t_i \in T_i} \sum_{i=1}^{N} \sum_{e \in E} x_i^e \ c_e(t_i^e, \rho_i) \tag{4a}$$

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

$$\forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r. \tag{4b}$$

Theorem 2. *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 Part B of supplementary materials. ■

The constraints in (4b) require that given any r and any $k \le r$, the total travel time from σ_k to σ_{r+1}, i.e., the aggregate execution times of tasks $\{\tau_k, \tau_{k+1}, ...\tau_r\}$, should be bounded above by $T_{r+1}^{\text{in}} - T_k^{\text{out}}$. Now we relax the constraints in (4b) to the objective function by introducing a non-negative Lagrangian dual variable $\lambda_k^r \ge 0$ for each (k, r) pair

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

where the equality in (a) holds due to the following reasons. The explanation on the reasons is spanning over the next a few sentences. It is clear that the specific dual variable λ_k^r in the aforementioned Lagrangian function is only associated with the travel time of tasks from τ_k to τ_r. To this end we observe that λ_k^r is associated with τ_i if and only if $k \le i \le r$. Therefore, the set of the dual variables associated with a specific task τ_i is $\{\lambda_k^r : \forall k \le i, \forall r \ge i\}$. Based on this observation, we have

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

since both sides are the sum of the dual variable times the task fulfilling time over all correlative tasks and dual variables.

By defining μ_i as

$$\mu_i \triangleq \sum_{r=i}^{K} \sum_{k=1}^{r} \lambda_k^r, \tag{5}$$

our Lagrangian function is

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

The dual problem of MEET is

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

Theorem 3.

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

where

$$w_i^e(\mu_i) \triangleq c_e(t_i^e(\mu_i), \rho_i) + \mu_i t_i^e(\mu_i), \tag{6}$$

with

$$t_i^e(\mu_i) \triangleq \arg \min_{t_e^l \le t_i^e \le t_e(\cdot)} (c_e(t_i^e, \rho_i) + \mu_i t_i^e). \tag{7}$$

Proof: Refer to Part C of supplementary materials. ■

In Thm. 3, $t_i^e(\mu_i)$ is the optimal travel time that minimizes the penalized edge cost with a price μ_i imposed on the edge travel time given specific μ_i, $w_i^e(\mu_i)$ 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. 3, given a set of non-negative dual variables $\vec{\lambda}$, we can obtain the value of $D(\vec{\lambda})$ by solving K shortest path problems independently. This critical observation motivates us to obtain close-to-optimal solutions to MEET, by iteratively updating dual variables $\vec{\lambda}$ to minimize duality gap.

C. An Efficient Heuristic SPEED for MEET

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

$$\delta(\mu_i) \triangleq \sum_{e \in p(\cdot_i)} t_i^e(\mu_i). \tag{8}$$

A key observation of $\delta(\mu_i)$ is presented as follows.

Theorem 4. *$\delta(\mu_i)$ is non-increasing in $\mu_i \in [0, +\infty)$ for any $i = 1, 2, ..., K$.*

Proof: Similar to [4, Thm.3] and is skipped. ■

In the next, we introduce a set of conditions under which we can construct an optimal solution to MEET.

Theorem 5. *Suppose the $\{\lambda_k^r \ge 0, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r\}$, the $\{\mu_i, \forall i = 1, 2, .., K\}$ computed in (5), and the $\{\delta(\mu_i), \forall i = 1, 2, ..., K\}$ computed in (8) satisfy that*

$$\left[T_k^{out} - \left(T_{r+1}^{in} + \sum_{j=k}^{r} \delta(\mu_j)\right)\right]_+ = 0,$$
$$\forall r = 1, 2, ..., K, \ \forall k = 1, 2, ..., r, \tag{9}$$

where the 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 the task τ_i with a travel time of $t_i^e(\mu_i)$ assigned for each edge $e \in p(\mu_i)$, and this solution must be the optimal solution to MEET.

Proof: Refer to Part D of supplementary materials. ■

Our proposed SPEED uses a sub-gradient based heuristic scheme in Algorithm 1 to iteratively update dual variables towards satisfying conditions in (9):

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

Algorithm 1 SPEED$(G, \vec{\sigma})$

1: **procedure**
2: ite=1, $p =$ NULL, $c(p) = +\infty$, $\lambda_k^r = \dot{\lambda}_k^r = \lambda_{\max}$
3: **while** $\exists\, r, k: \dot{\lambda}_k^r > \text{tol}$ and ite $\le$ ITE **do**
4: **for** $i = 1, 2, ..., K$ **do**
5: Set μ_i according to equality (5)
6: Obtain $t_i^e(\mu_i), \forall e \in E$ according to equality (7)
7: Set $w_i^e(\mu_i), \forall e \in E$ according to equality (6)
8: Get the shortest path $p(\mu_i)$ from σ_i to σ_{i+1}
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 (10)
11: Set $\lambda_k^r = \max\{\lambda_k^r + \dot{\lambda}_k^r, 0\}, \forall r, \forall k$
12: **if** $\dot{\lambda}_k^r = 0, \forall r = 1, 2, ..., K, \forall k = 1, 2, ..., r$ **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

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.

Theorem 6. *For any theoretically-feasible MEET instance, Algorithm 1 must return a feasible solution p meeting all time window constraints. The optimality gap between the cost of p, namely $c(p)$, and the optimal cost, namely OPT, must be bounded above as follows:*

$$c(p) - OPT \le \begin{cases} 0, \\ K^2 \max_{\forall r,k} \dot{\lambda}_k^r \max_{\forall r,k} \bar{\lambda}_k^r / \phi(\bar{\lambda}_k^r), \end{cases}$$

where the first case of the zero gap is for p returned in Line 13, and the second case of the problem-dependent gap 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 Part E of supplementary materials. ■

According to Thm. 6, Algorithm 1 always obtains feasible solutions to MEET with performance guarantee. Moreover, the obtained solution must be optimal if it is returned in Line 13. We note that Deng *et al.* [4], [5] have designed an efficient heuristic for the single-task problem PASO which is a special case of our multi-task problem MEET. Although their heuristic also explores the Lagrangian dual relaxation, it is fundamentally different from our heuristic SPEED: there is only one deadline constraint in PASO while the number of deadline constraints in MEET is $O(K^2)$ (see (4b)). Hence only one dual variable exists in the dual problem to PASO while there is a set of dual variables in the dual problem to MEET. The heuristic of PASO uses binary search to find the optimal dual variable which minimizes the duality gap. It is not clear how to conduct binary search to obtain the optimal values of a set of dual variables if we extend their heuristic to the multi-task setting. Instead of using binary search, our SPEED iteratively updates a set of dual variables towards the optimal, by following the sub-gradient of the dual relaxation.

Finally, note that in SPEED we initialize dual variables by $\lambda_{\max}$. Obviously SPEED can converge faster with a better (smaller) $\lambda_{\max}$. Considering that the single-task heuristic of PASO [4], [5] is a special case of our multi-task SPEED and they both explore the Lagrangian dual relaxation, we can use the single-task heuristic to solve K PASO problems independently and obtain a good $\lambda_{\max}$ for SPEED: for each task, we set the deadline constraint to be its minimum fulfilling time, and run the single-task heuristic to obtain the dual variable corresponding to the close-to-optimal solution of PASO; after solving K PASO problems, we set $\lambda_{\max}$ to be the largest value of obtained K dual variables.

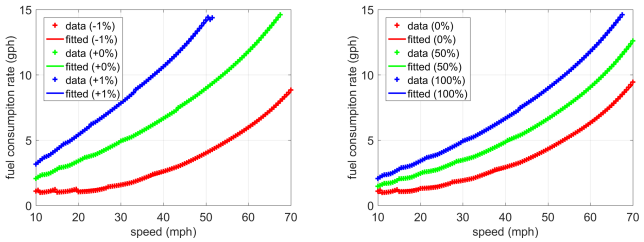
V. Performance Evaluation

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 algorithm proposed by [4], [5] from their single-task setting to our multi-task setting. We implement all algorithms using C++ and run them on a laptop with a 4-core Intel Core-i5-4200M (2.50GHz) processor and 16GB memory, running 64-bit Ubuntu 16.04 LTS. We use the SNAP graph structure [33] to construct the US national highway system from the clinched highway mapping project[5] 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[6] from the US 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[7] 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 [34] to collect fuel consumption rate data with driving speed for different road grade and truck load for a class-8 heavy truck of Kenworth T800[8]. Then we use the curve fitting toolbox in MATLAB to learn the fuel consumption rate function $f_e(r_e, \rho)$.

The highway network 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" part is divided into 22 regions (see Fig. 1b), where the node nearest to each region's center is used as the candidate for the task source node and the task destination node, same as the experimental setting in [4], [5]. 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, i.e., $T_i^{\text{out}} = 0, i = 1, 2, ..., K$. As for SPEED (i.e., Algorithm 1), we fix $\phi(\ )$ to be 0.1 for all dual variables, and fix the algorithm tolerance level tol to be 0.01. Besides, for the sake of convenience, we denote the minimal time of independently fulfilling the single task τ_i as t_i, i.e.,

$$t_i \ , \ \min_{x_i \in X_i} \ x_i^e \ t_e^l \ , \quad i = 1, 2, ..., K. \tag{11}$$

[5]Clinched highway mapping, http://cmap.m-plex.com/
[6]Elevation point query service, http://nationalmap.gov/epqs/
[7]HERE, https://www.here.com/
[8]Kenworth T800, https://www.kenworth.com/trucks/t800



(a) Results with different road grade. (b) Results with different truck load.

Fig. 2: Simulated fuel consumption rate with driving speed.

A. Modeling Fuel-Consumption-Rate Function

In our simulations, given a road grade and a truck load, we model the fuel consumption rate (gph) with speed (mph) for the T800 truck using the following function

$$f(r) \ = \ a \ \ r^3 + b \ \ r^2 + c \ \ r + d,$$

where r is the speed, $f(r)$ is the fuel consumption rate, and a, b, c, d are parameters learned by MATLAB based on the simulated fuel consumption rate data using ADVISOR.

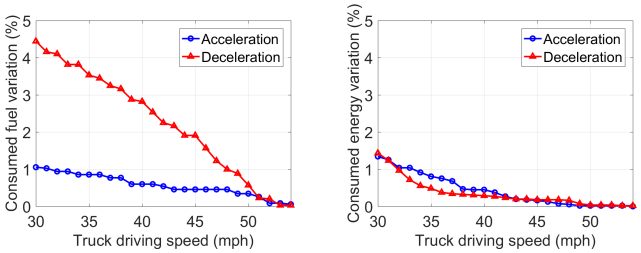
In order to obtain the fuel consumption rate data f given a driving speed r, a road grade θ, and a truck load ρ, we specify a driving cycle test of 4 hours with a constant-speed of r, a constant road grade of θ, and a constant truck load of ρ. We enumerate r from 10 to 70 with a step of 0.2, θ from 10.0 to 10.0 with a step of 0.1 , and we set ρ to be 0% (empty load), 50% (half load), or 100% (full load).

Fig. 2 presents a part of the fuel consumption rate results with different driving speed in our simulations. In Fig. 2a, we fix the truck to be full-load and assume it drives on a road with grade of 1 , 0 , or 1 , where we observe that high speed and large grade result in large fuel consumption rate, and verify that the fitted function $f(r)$ is strictly convex in reasonable speed regions. In Fig. 2b, we assume the truck has a load of 0%, 50%, or 100%, driving on a road with grade of 0 . Similar to Fig. 2a, Fig. 2b also verifies the convexity of the fuel consumption rate function. Besides, it is clear that the truck load greatly affect the truck fuel consumption rate, and a heavy load leads to a large fuel consumption.

B. Acceleration/Deceleration Fuel Consumption

As discussed in Sec. III-A, we ignore the fuel consumption incurred by acceleration/deceleration. As proven by [5, Lem. 1] that following a constant-speed is most fuel-economic when driving inside a road segment, we do not need to consider acceleration/deceleration for a truck to traverse each $e \ 2 \ E$; and in this section, simulations using ADVISOR further show that it is fair to neglect acceleration/deceleration for a truck to switch between different road segments, since the fuel consumption incurred by acceleration or deceleration during switching is way smaller as compared to that incurred by driving at a constant-speed to traverse a road segment.

We first consider an instance of MEET ($\sigma_1 = 1$, $\sigma_2 = 9$, $\sigma_3 = 22$, $T_2^{\text{in}} = 40$, $T_3^{\text{in}} = 65$, $\rho_1 = \rho_2 = 100\%$). After running SPEED, we obtain a close-to-optimal solution. Later



(a) Results of a conventional truck. (b) Results of an electric truck.

Fig. 3: Energy consumption variation results (%) caused by truck acceleration or deceleration. For the conventional truck, it consumes fuel; while for the electric truck, it consumes the energy stored in the battery.

we compare SPEED with other approaches using the same instance, with the results summarized in Tab IV. Here to evaluate the acceleration/deceleration, (i) we calculate the gap of the assigned speeds of two adjacent edges in the path obtained by this solution. After enumerating a total of 961 speed gap results, the mean is 3.61 mph, the minimum is 0 mph, the first quartile is 0.14 mph, the median is 0.40 mph, the third quartile is 3.09 mph, and the maximum is 34.4 mph; (ii) we further use ADVISOR to simulate our solution considering two scenarios. In scenario one, the truck follows the constant-speed assigned by the solution to traverse each edge. In scenario two when traversing each edge, the truck first accelerates (or decelerates) under a rate of 1.8 ft/s^2 (or 1.8 ft/s^2) due to the speed gap of switching, and then follow the assigned constant-speed. Note that 1.8 ft/s^2 is an appropriate acceleration rate for a heavy truck as suggested by the existing study [28]. We observe that the fuel consumption of scenario two differs from that of scenario one only by 0.34%. Overall in our solution, the fuel consumption of acceleration/deceleration when switching between edges is way smaller than that of driving by a constant-speed to traverse an edge.

Next we assume that a Kenworth T800 truck with an empty load is driving through a road segment with a grade of 0 and a length of 3.26 miles. Here 3.26 miles is the average length of all 82781 road segments in our simulated US national highway. We consider three different scenarios of traversing the road segment: in scenario one, the truck follows a constant-speed of 55 mph; in scenario two, the truck first accelerates from x mph to 55 mph under a rate of 1.8 ft/s^2, and then follows a constant-speed of 55 mph; in scenario three, the truck first follows a constant-speed of 55 mph, and then decelerates from 55 mph to x mph under a rate of 1.8 ft/s^2. We enumerate x from 30 to 54 with a step of 1. Suppose c_c (resp. c_a, c_d) is the fuel consumption of scenario one (resp. scenario two, scenario three). We consider the variation comparing c_a (resp. c_d) with c_c, i.e., $|c_c - c_a|/c_c$ (resp. $|c_c - c_d|/c_c$), which estimates the impact of acceleration (resp. deceleration) on fuel consumption. Simulation results are given in Fig. 3a, where we observe that the fuel consumption of acceleration/deceleration when switching between edges is way smaller than that of driving by a constant-speed to traverse an edge.

Overall, we observe that it is fair to ignore the fuel consumption incurred by acceleration/deceleration, which simplifies our problem without incurring a significant optimality loss.

TABLE III: Performance comparison of FPTAS (F) with $\epsilon = 0.1$ 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. 6.

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

C. Comparing FPTAS with SPEED

We consider the scenario where the truck fulfills two tasks both with full load. Results for two small-scale instances are shown in Tab. III. In both instances, the solutions of SPEED are close-to-optimal (consume less than 2%-more fuel than the optimal). FPTAS can only handle the smallest instance (first instance with $n = 1518$ and $m = 3274$) for $\epsilon = 0.1$. As compared to SPEED, FPTAS incurs much longer (100) running time for the first instance. For the second instance with $n = 6187$ and $m = 13708$, FPTAS fails to output a solution even after running it for 12 hours.

D. Comparing SPEED with Other Alternatives for Two Tasks

We compare SPEED with two baselines and a conceivable alternative approach, for large-scale instances with two tasks:

1) A fastest-path-based baseline without task execution times optimization: the travel speed r_e for each $e \in E$ is fixed as the maximum one, i.e., $r_e = r_e^u$, and each task τ_i is fulfilled using the path with the minimal travel time from its source σ_i to its destination σ_{i+1}.
2) A shortest-path-based baseline without task execution times optimization: the travel speed r_e for each $e \in E$ is fixed as the maximum one, i.e., $r_e = r_e^u$, and each τ_i is fulfilled using the path with the 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. Specifically, in the ith iteration, we run the heuristic proposed by [4], [5] 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 $T_{i+1}^{\text{in}} \quad \max\{T_i^{\text{out}}, a_{i\ 1}\}$, 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 MEET 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 the US); and (ii) the latest arrival

TABLE IV: Comparison of SPEED and other alternatives for the instance of $(1, 9, 22, 40, 65)$. A lower bound for the optimal fuel consumption is 478.73 according to Thm. 6. 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**

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.

We give the simulation results in Tab. IV. 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 optimal. As seen in Tab. IV, 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).

From Tab. IV we also observe that 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 the fuel consumption for truck operators under the multi-task setting.

E. Comparing SPEED with Alternatives for Three Tasks

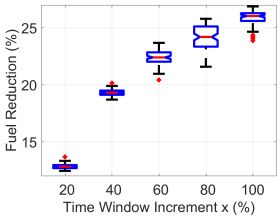
We then simulate MEET instances assuming three tasks. We assume full load for τ_1 and τ_3, but an empty load for τ_2, corresponding to a practical scenario where the truck is required to fulfill two freight transportation requests (τ_1 from σ_1 to σ_2, and τ_3 from σ_3 to σ_4), and have to travel with an empty load from the destination (σ_2) of the first request to the source (σ_3) of the second request. Such an MEET instance can be described as a tuple of $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, T_2^{\text{in}}, T_3^{\text{in}}, T_4^{\text{in}})$.

We simulate the instance $(18, 11, 4, 22, 45, 75, 105)$ and give the results in Tab. V, where SPEED obtains a close-to-optimal solution since a close-to-optimal execution time is allocated to each task (35.36 hours for τ_1, 31.45 hours for τ_2, and 38.19 hours for τ_3). In sharp contrast, the PASO-based alternative fails to minimize fuel consumption efficiently due to the non-optimal execution times allocation for tasks.

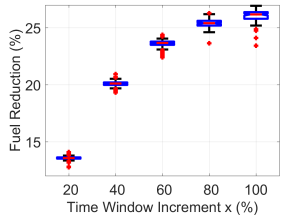
F. Impact of Time Windows on the Fuel Consumption

For MEET instances of two tasks each with a full load, in this section we estimate the effect of time window constraints on the achieved fuel consumptions of different algorithms.

First, we consider a setting of $T_2^{\text{in}} = t_1 \ \ (1 + x\%)$ and $T_3^{\text{in}} = (t_1 + t_2) \ \ (1 + x\%)$. We enumerate x from 20 to 100 with a step of 20. Given a specific x, we run 1000 simulations where in each simulation, we randomly select σ_1, σ_2 ~~6=~~ σ_1, and σ_3 ~~6=~~ σ_2. We give the fuel consumption reduction results comparing our SPEED to the fastest-path-based approach in Fig. 4a, according to which it is clear that larger time windows can lead to more fuel consumption reduction.



(a) We set $T_2^{\text{in}} = t_1^* \cdot (1 + x\%)$ and $T_3^{\text{in}} = (t_1^* + t_2^*) \cdot (1 + x\%)$.



(b) We set $T_2^{\text{in}} = T_3^{\text{in}} = (t_1^* + t_2^*) \cdot (1 + x\%)$.

Fig. 4: Fuel saving achieved by SPEED with the time window constraints, as compared to the fastest-path-based baseline. Here t_i is defined in the equation in (11).

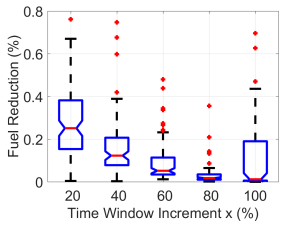


Fig. 5: Fuel saving of SPEED as compared to the PASO-based approach. We set $T_2^{\text{in}} = t_1 \ \ (1 + x\%)$ and $T_3^{\text{in}} = (t_1 + t_2) \ \ (1 + x\%)$, where t_i is defined in the equation in (11).

We compare SPEED with the fastest-path baseline in Fig. 4a. The shortest-path baseline performs almost same as the fastest-path baseline in all problem instances. We present the fuel consumption reduction results comparing SPEED with the PASO-based alternative in Fig. 5. We observe that the fuel consumption performance of SPEED is just slightly better than that of the PASO-based alternative (SPEED saves less than 1% fuel for most instances). This is because under the setting of $T_2^{\text{in}} = t_1 \ \ (1+x\%)$ and $T_3^{\text{in}} = (t_1+t_2) \ \ (1+x\%)$, the greedy task execution times allocation of the PASO-based approach is close-to-optimal.

Next we consider a different setting of $T_2^{\text{in}} = T_3^{\text{in}} = (t_1 + t_2) \ \ (1+x\%)$. We give the fuel consumption reduction results comparing our SPEED to the fastest-path baseline in Fig. 4b. Similar to Fig. 4a, from Fig. 4b we observe that larger time window constraints lead to more fuel consumption reduction of our SPEED. Moreover, we observe that the fuel saving of Fig. 4b is more than that of Fig. 4a. This is because that we set

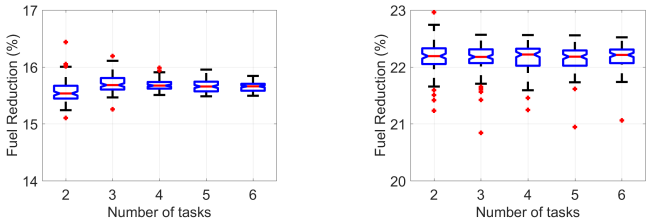
TABLE V: Comparison of SPEED and other alternatives for the instance of $(18, 11, 4, 22, 45, 75, 105)$, where a lower bound for the optimal total fuel consumption is 664.77 according to Thm. 6.

Algorithm	First task from 18 to 11,			Second task from 11 to 4			Third task from 4 to 22			Total performance			
	Time	Distance	Fuel	Time	Distance	Fuel	Time	Distance	Fuel	Time	Distance	Fuel	Fuel-Incre.
Fastest	24.5	1616	341.38	20.02	1401	183.57	26.52	1744	366.21	71.04	4761	891.16	**34.06**
Shortest	24.5	1616	341.39	20.02	1401	183.57	26.59	1743	366.25	71.11	4760	891.21	**34.06**
PASO	**44.97**	1616	254.18	**30.02**	1401	112.15	**29.79**	1746	330.72	104.78	4763	697.05	**4.86**
SPEED	**35.36**	1616	269.11	**31.45**	1401	107.71	**38.19**	1747	288.62	105	4764	665.44	**0.1**

TABLE VI: Fuel consumption reduction of SPEED as compared to the PASO-based alternative. We set $T_2^{\text{in}} = T_3^{\text{in}} = (t_1 + t_2)\ (1 + x\%)$, where t_i is defined in the equation (11).

Time window constraints increment $x\%$	20%	40%	60%	80%	100%
Ratio of solvable instances of the PASO alternative	1%	13%	68%	92%	100%
Fuel saving of SPEED on average?	9.2%	8.6%	5.9%	2.8%	0.3%

?: Because the PASO-based alternative only solves part of the theoretically feasible instances, here the fuel saving of SPEED is calculated only for those solvable instances instead of all the 1000 instances.



(a) Simulations for $x = 25$. (b) Simulations for $x = 50$.

Fig. 6: 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 $(1 + x\%) \sum_{i=1}^{K} t_i$.

$T_2^{\text{in}} = (t_1 + t_2)\ (1 + x\%)$ in Fig. 4b, but set $T_2^{\text{in}} = t_1\ (1 + x\%)$ in Fig. 4a. Therefore, MEET instances in Fig. 4b admit larger design spaces for minimizing the total fuel consumption.

Recall that the PASO alternative obtains close-to-optimal solutions meeting time window constraints (see Fig. 5) in the first setting. In sharp contrast, in the second setting where $T_2^{\text{in}} = T_3^{\text{in}} = (t_1 + t_2)\ (1 + x\%)$, the PASO-based approach cannot even obtain feasible solutions for many instances (see Tab. VI). According to Tab. VI, overall the PASO alternative fails to output solutions that can meet time window constraints for 45% of the instances. This is due to the non-optimal greedy task execution times allocation scheme: (i) the achieved task execution time of the first task τ_1 is way too large in order to minimize the fuel consumption of fulfilling τ_1, after solving the problem PASO for τ_1 subject to a deadline of $T_2^{\text{in}} = (t_1 + t_2)\ (1 + x\%)$; (ii) then the remaining deadline for fulfilling the task τ_2 is smaller than t_2 in many instances, and thus no feasible solutions can be obtained. Note that our SPEED always achieves close-to-optimal solutions meeting time window constraints in all instances, by jointly allocating task execution times for the two tasks towards the optimal.

G. Fuel Consumption of SPEED with the Number of Tasks

Finally, we carry out simulations with randomly generated inputs to evaluate SPEED with different number of tasks.

For each experimental case characterized by a specific $K \in \{2, 3, ..., 6\}$ and an $x \in \{25, 50\}$, we run 1000 simulations each with K full-load tasks defined by randomly selected source/destination nodes $\{\sigma_1, ..., \sigma_{K+1}\}$. Latest arrival time constraints are same for any $\sigma_i, i = 2, ..., K+1 : T_i^{\text{in}} = t\ (1 + x\%)$, where $t = \sum_{i=1}^{K} t_i$ is the minimal time to fulfill those K tasks. Fig. 6 shows the fuel reduction results of SPEED compared with the fastest-path baseline, which suggests that the fuel saving of SPEED is robust to the number of tasks (16% for $x = 25$ and 22% for $x = 50$).

TABLE VII: Ratio of solvable MEET instances of the PASO-based alternative, with the number of tasks (i.e., with K). All latest arrival time constraints are set to be $(1 + x\%) \sum_{i=1}^{K} t_i$.

	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$
$x = 25$	1%	0%	0%	0%	0%
$x = 50$	27%	2%	0%	0%	0%

We give the number of solvable simulated instances of the PASO-based alternative in Tab. VII. We remark again that in contrast to the results of PASO shown in Tab. VII, our SPEED always obtains close-to-optimal solutions meeting time window constraints for all simulated instances.

VI. Discussions and Future Directions

A. Generalizing SPEED to Optimize Resting Times

In order to guarantee a safe driving, truck drivers in the US are subject to hours-of-service regulations[9] which regulate the minimum amount of resting time. SPEED can figure out resting time for fulfilling each task, and leave the decision of when and where to rest to be made by drivers. We observe that the amount of resting time t_r can be modeled as a linear function $t_r = h\ \ t$, where $h \quad 0$ is a constant and t is the driving time. In this case, the time window constraints in (2b) of MEET shall include the resting time requirement as follows

$$a_i = \max\ \ a_{i\ 1}, T_i^{\text{out}}\ \ + (1 + h_i) \sum_{e2E} x_i^e t_i^e \qquad T_{i+1}^{\text{in}}. \quad (12)$$

As compared to the constraints in (2b), the constraints in (12) does not change the structure of MEET. Our SPEED thus can solve the corresponding problem. It is clear that if T_i is the execution time of τ_i allocated by the solution of SPEED, $h_i\ \ T_i$ will be the resting time when the truck fulfills τ_i following the solution. Note that SPEED only outputs a

[9]Wikipedia, https://en.wikipedia.org/wiki/Hour

minimum amount of resting time, but cannot optimize the selection of when/where to rest. We leave it as an important future direction beyond the study in this paper.

B. Generalizing SPEED to Problems with Electric Trucks

Now let us consider a new trucking scenario, where an electric truck timely delivers ρ freight from a source s to a destination d over a highway network $G(V, E)$. The battery capacity is small and hence cannot support the truck to travel from s to d without charging. We assume that the truck can charge at an intermediate node (a charging station) u with a charging rate of R watt and a charging expense of P dollars per hour. The problem requires to find a solution of path planning, speed planning, and charging time assignment, for the electric truck to minimize the charging expense while successfully delivering freight under time window constraints.

According to [35], it is reasonable to assume that the energy consumption for an electric truck to pass an edge $e \in E$ is a strictly convex function $c_e(t_e, \rho)$ of the edge travel time t_e. Now we expand the charging node u to two nodes u_1 and u_2, and add a directed link $\hat{e} = (u_1, u_2)$ with a cost function of $c_{\hat{e}}(t_{\hat{e}}, \rho) = R \cdot t_{\hat{e}}$. Consider three tasks with the first task from s to u_1, the second task from u_1 to u_2, and the third task from u_2 to d. Then the problem requires the truck to timely fulfill the three tasks to minimize the charging expense $P \cdot t_{\hat{e}}$.

Here we remark that although the deceleration of an electric truck is very different from that of a gasoline truck due to regenerative braking, it is fair to ignore the energy consumption incurred by acceleration/deceleration of an electric truck, due to the same motivation introduced in Sec. III-A. In Fig. 3b, we also use ADVISOR to empirically verify that the energy consumption of acceleration/deceleration of switching between edges is way smaller than that of traversing an edge by a constant-speed, from the perspective of an electric truck.

The aforementioned problem is similar to our MEET, but with extra capacity constraints which limit the cost (i.e., energy consumption) of fulfilling each task due to the truck battery capacity concern. We remark that we can follow the same approach used to develop SPEED to design an efficient heuristic to solve the transportation problem of electric trucks: we relax deadline constraints as well as capacity constraints to the objective function by introducing Lagrangian dual variables and obtain the dual problem; then we can update dual variables iteratively by following the dual sub-gradient to minimize the duality gap and hence obtain close-to-optimal solutions. Note that we do not consider problems which optimize the selection of charging stations to save fuel. We leave it as a possible future direction beyond the study in this paper.

VII. Conclusion

We consider a truck driving across a national highway system to fulfill multiple tasks in a specific order. We formulate a problem MEET, which minimizes the total fuel consumption subject to the pickup and delivery time window constraints of individual tasks. We observe that optimizing task execution times is a new challenging design space for saving fuel introduced by the multi-task setting, and it differentiates our multi-task study from existing ones that are under the single-task setting. We show that MEET is NP-hard, and optimizing task execution times is a non-convex puzzle. We then exploit the problem structure to develop an FPTAS and an efficient heuristic SPEED. We characterize sufficient conditions under which SPEED generates an optimal solution, and derive a performance gap comparing the solution of SPEED with the optimal when the conditions are not satisfied. We evaluate 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 generalized from the state-of-the-art single-task algorithm. Moreover, our algorithms always obtain close-to-optimal solutions meeting all time window constraints for all feasible instances, while the conceivable alternative fails to meet time window constraints for up to 45% of the instances in our simulations. In addition, the fuel saving of our solutions is robust to the number of tasks to be fulfilled.

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, currently an Assistant Professor in College of Electronic and Information Engineering, Shenzhen University, China, for sharing the simulation code of PASO [4] and for his kind assistance on experiments.

References

[1] Q. Liu, H. Zeng, and M. Chen, "Energy-efficient timely truck transportation for geographically-dispersed tasks," in *ACM International Conference on Future Energy Systems*, 2018.

[2] American trucking association and others, "ata american trucking trends," 2017.

[3] A. Hooper and D. Murray, "An analysis of the operational costs of trucking: 2017 update," 2017.

[4] L. Deng, M. H. Hajiesmaili, M. Chen, and H. Zeng, "Energy-efficient timely transportation of long-haul heavy-duty trucks," in *ACM International Conference on Future Energy Systems*, 2016.

[5] ——, "Energy-efficient timely transportation of long-haul heavy-duty trucks," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–15, 2017.

[6] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations research*, vol. 17, no. 1, pp. 36–42, 1992.

[7] D. H. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest path problem," *Operations Research Letters*, vol. 28, no. 5, pp. 213–219, 2001.

[8] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the qos routing problem," in *IEEE International Conference on Computer Communications*, 2001.

[9] E. Hellström, "Look-ahead control of heavy trucks utilizing road topography," Ph.D. dissertation, Institutionen för systemteknik, 2007.

[10] E. Hellström, M. Ivarsson, J. Åslund, and L. Nielsen, "Look-ahead control for heavy trucks to minimize trip time and fuel consumption," *Control Engineering Practice*, vol. 17, no. 2, pp. 245–254, 2009.

[11] K. Boriboonsomsin, M. J. Barth, W. Zhu, and A. Vu, "Eco-routing navigation system based on multisource historical and real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1694–1704, 2012.

[12] G. Scora, K. Boriboonsomsin, and M. Barth, "Value of eco-friendly route choice for heavy-duty trucks," *Research in Transportation Economics*, vol. 52, pp. 3–14, 2015.

[13] 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.