Abstract—We consider a timely transportation problem where a heavy-duty truck travels between two locations across the national highway system, subject to a hard deadline constraint. Our objective is to minimize the total fuel consumption of the truck, by optimizing both route planning and speed planning. The problem is important for cost-effective and environment-friendly truck operation, and it is uniquely challenging due to its combinatorial nature as well as the need of considering hard deadline constraint. We first show that the problem is NP-complete; thus exact solution is computational prohibited unless \( P = \text{NP} \). We then design a fully polynomial time approximation scheme (FPTAS) to solve it. While achieving highly-preferred theoretical performance guarantee, the proposed FPTAS still suffers from long running time when applying to national-wide highway systems with tens of thousands of nodes and edges. Leveraging elegant insights from studying the dual of the original problem, we design a heuristic with much lower complexity. The proposed heuristic allows us to tackle the energy-efficient timely transportation problem on large-scale national highway systems. We further characterize a condition under which our heuristic generates an optimal solution. We observe that the condition holds in most of practical instances in numerical experiments, justifying the superior empirical performance of our heuristic. We carry out extensive numerical experiments using real-world truck data over the actual U.S. highway network. The results show that our proposed solutions achieve 17% (resp. 14%) fuel consumption reduction, as compared with a fastest path (resp. shortest path) algorithm adapted from common practice.

Index Terms—Energy-efficient transportation, timely delivery, route planning, speed planning.

I. INTRODUCTION

In the U.S., heavy-duty trucks haul more than 70% of all freight tonnage [2], and they consume 17.6% of energy in transportation sector [3, Table 2.8] and contribute to about 5% of the greenhouse gas emission [4]. Fuel cost is the largest operating cost (34%) of truck owners/operators [5], and reducing fuel consumption is critical for cost-effective and environment-friendly heavy-duty truck operations.

Currently there are mainly two lines of efforts to reduce fuel consumption of heavy-duty trucks. The first line is to operate with more fuel efficient trucks, from better designs for engines, drivetrains, aerodynamics, and tires [6]–[8], to better management of truck parts such as maintaining optimal tire pressures [9]. The second line is to operate heavy-duty trucks more economically. This explores several possibilities, e.g., reducing idling energy consumption [10], platooning more than one heavy-duty trucks [11], [12], route planning [13]–[15], and speed planning [16]–[19]. In this paper, we focus on route and speed planning. Different routes could have different mileages, levels of congestion, road grades, and surface types, etc., all of which would largely affect the fuel consumption. Real-world studies [15] show that choosing a more efficient route for a heavy-duty truck can improve its fuel economy by 21%. Speed planning is another well recognized approach to effectively reduce fuel consumption. Different running speed could lead to different fuel economy. For a vehicle with certain weight running on a road, normally there is a most fuel-efficient speed. When the running speed is below or above the most fuel-efficient speed, the fuel economy will be degraded. As a rule of thumb for truck operations on highway, every one mile per hour (mph) increase in speed (above the most fuel-efficient speed) incurs about 0.14 mile per gallon (mpg) decrease in fuel economy [18], [19].

However, operating at low speed may result in excessive travel time and the goods carried by the truck cannot be delivered on time. We remark that timely delivery is critical for truck operators [20], [21]. As estimated by the U.S. Federal Highway Administration (FHWA) in [20], unexpected delay can increase freight cost by 50% to 250%. Multiple reasons can explain the importance of timely delivery. First, some freight goods are perishable, such as food [22], which definitely require timely delivery. Second, to ensure customers’ satisfaction, some companies, e.g., Amazon, may have a service-level agreement (SLA) with users, under which the delivery delay is guaranteed [23]. Finally, violating scheduled delay can introduce difficulties for global logistic decisions and even increase the uncertainty and inefficiency of supply chains [20]. Overall, it is crucial to ensure timely goods delivery for truck operators, and considering timely delivery in fuel cost minimization poses a unique challenge.

Motivated by the above observations, in this paper, we study the problem of energy-efficient timely transportation for heavy-duty trucks. We aim to minimize the heavy duty

Energy-Efficient Timely Transportation of Long-Haul Heavy-Duty Trucks

Lei Deng, Mohammad H. Hajiesmaili, Minghua Chen, Senior Member, IEEE, and Haibo Zeng, Member, IEEE

Abstract—We consider a timely transportation problem where a heavy-duty truck travels between two locations across the national highway system, subject to a hard deadline constraint. Our objective is to minimize the total fuel consumption of the truck, by optimizing both route planning and speed planning. The problem is important for cost-effective and environment-friendly truck operation, and it is uniquely challenging due to its combinatorial nature as well as the need of considering hard deadline constraint. We first show that the problem is NP-complete; thus exact solution is computational prohibited unless \( P = \text{NP} \). We then design a fully polynomial time approximation scheme (FPTAS) to solve it. While achieving highly-preferred theoretical performance guarantee, the proposed FPTAS still suffers from long running time when applying to national-wide highway systems with tens of thousands of nodes and edges. Leveraging elegant insights from studying the dual of the original problem, we design a heuristic with much lower complexity. The proposed heuristic allows us to tackle the energy-efficient timely transportation problem on large-scale national highway systems. We further characterize a condition under which our heuristic generates an optimal solution. We observe that the condition holds in most of practical instances in numerical experiments, justifying the superior empirical performance of our heuristic. We carry out extensive numerical experiments using real-world truck data over the actual U.S. highway network. The results show that our proposed solutions achieve 17% (resp. 14%) fuel consumption reduction, as compared with a fastest path (resp. shortest path) algorithm adapted from common practice.

Index Terms—Energy-efficient transportation, timely delivery, route planning, speed planning.

I. INTRODUCTION

In the U.S., heavy-duty trucks haul more than 70% of all freight tonnage [2], and they consume 17.6% of energy in transportation sector [3, Table 2.8] and contribute to about 5% of the greenhouse gas emission [4]. Fuel cost is the largest operating cost (34%) of truck owners/operators [5], and reducing fuel consumption is critical for cost-effective and environment-friendly heavy-duty truck operations.

Currently there are mainly two lines of efforts to reduce fuel consumption of heavy-duty trucks. The first line is to operate with more fuel efficient trucks, from better designs for engines, drivetrains, aerodynamics, and tires [6]–[8], to better management of truck parts such as maintaining optimal tire pressures [9]. The second line is to operate heavy-duty trucks more economically. This explores several possibilities, e.g., reducing idling energy consumption [10], platooning more than one heavy-duty trucks [11], [12], route planning [13]–[15], and speed planning [16]–[19]. In this paper, we focus on route and speed planning. Different routes could have different mileages, levels of congestion, road grades, and surface types, etc., all of which would largely affect the fuel consumption. Real-world studies [15] show that choosing a more efficient route for a heavy-duty truck can improve its fuel economy by 21%. Speed planning is another well recognized approach to effectively reduce fuel consumption. Different running speed could lead to different fuel economy. For a vehicle with certain weight running on a road, normally there is a most fuel-efficient speed. When the running speed is below or above the most fuel-efficient speed, the fuel economy will be degraded. As a rule of thumb for truck operations on highway, every one mile per hour (mph) increase in speed (above the most fuel-efficient speed) incurs about 0.14 mile per gallon (mpg) decrease in fuel economy [18], [19].

However, operating at low speed may result in excessive travel time and the goods carried by the truck cannot be delivered on time. We remark that timely delivery is critical for truck operators [20], [21]. As estimated by the U.S. Federal Highway Administration (FHWA) in [20], unexpected delay can increase freight cost by 50% to 250%. Multiple reasons can explain the importance of timely delivery. First, some freight goods are perishable, such as food [22], which definitely require timely delivery. Second, to ensure customers’ satisfaction, some companies, e.g., Amazon, may have a service-level agreement (SLA) with users, under which the delivery delay is guaranteed [23]. Finally, violating scheduled delay can introduce difficulties for global logistic decisions and even increase the uncertainty and inefficiency of supply chains [20]. Overall, it is crucial to ensure timely goods delivery for truck operators, and considering timely delivery in fuel cost minimization poses a unique challenge.

Motivated by the above observations, in this paper, we study the problem of energy-efficient timely transportation for heavy-duty trucks. We aim to minimize the heavy duty

IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS

Energy-Efficient Timely Transportation of Long-Haul Heavy-Duty Trucks

Lei Deng, Mohammad H. Hajiesmaili, Minghua Chen, Senior Member, IEEE, and Haibo Zeng, Member, IEEE
TABLE I
COMPARISONS OF OUR STUDY AND EXISTING WORKS ON PERFORMANCE OPTIMIZATION IN VARIOUS TRANSPORTATION SYSTEMS WITH DELAY TAKEN INTO CONSIDERATION. HERE RSP STANDS FOR RESTRICTED SHORTEST PATH PROBLEM, VRPTW STANDS FOR VEHICLE ROUTING PROBLEM WITH TIME WINDOWS, AND BSP STANDS FOR BI-OBJECTIVE SHORTEST PATH PROBLEM

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
<td>Cost minimization</td>
<td>Cost minimization</td>
<td>Cost minimization</td>
<td>Cost minimization or profit maximization</td>
<td>Bi-objective (cost and delay) minimization</td>
</tr>
<tr>
<td>Constraints</td>
<td>A hard deadline</td>
<td>A hard deadline</td>
<td>Time window*, Other constraints</td>
<td>Time window*, Other constraints</td>
<td>N/A</td>
</tr>
<tr>
<td>Design Spaces</td>
<td>Route planning, Speed planning</td>
<td>Route planning</td>
<td>Route planning, Speed planning</td>
<td>Route planning</td>
<td>Route planning</td>
</tr>
<tr>
<td>Results</td>
<td>Hardness</td>
<td>NP-Complete</td>
<td>NP-Complete [25]</td>
<td>NP-Complete [27]</td>
<td>N/A</td>
</tr>
<tr>
<td>Algorithms</td>
<td>Heuristic†</td>
<td>FPTAS, Heuristic†</td>
<td>FPTAS [24], Heuristic [26]</td>
<td>Heuristic [28]–[30]</td>
<td>Heuristic U [34], [35], Heuristic U [36]</td>
</tr>
</tbody>
</table>

* The time window constraint captures the hard deadline constraint in our problem and RSP as a special case.
† We further characterize a condition under which the heuristic outputs the optimal solution to our problem.

TABLE II
COMPARISONS OF OUR WORK AND EXISTING WORKS ON ENERGY-EFFICIENT HEAVY-DUTY TRUCK OPERATION

<table>
<thead>
<tr>
<th>Paper</th>
<th>Route Planning</th>
<th>Speed Planning</th>
<th>Hard Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>[37]</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>[16]</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>[17]</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>This work</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

We carry out extensive numerical experiments using real-world truck data over the U.S. highway network in Sec. V. The results show that our proposed solutions achieve 17% (resp. 14%) fuel consumption reduction, as compared to a fastest path (resp. shortest path) algorithm adapted from common practice. The amount of fuel consumption saving is enough to power up more than 90% of the entire transportation sector in New York State [38].

Comparison with existing works on energy-efficient heavy-duty truck operation. There are a large number of works focusing on energy-efficient heavy-duty truck operation, e.g., [16], [17], [37]. But to the best of our knowledge, our work is the first one that simultaneously considers route planning, speed planning, and hard deadline (see Tab. II).

Comparison with existing works on performance optimization in various transportation systems with delay taken into consideration. Theoretically, we also compare the problem studied in our work with other related problems studied in existing works in Tab. I. First, our energy-efficient timely transportation problem is a generalized version of Restricted Shortest Path problem (RSP) [24]–[26], with an extra design space of speed planning. Therefore, we generalize the FPTAS design and the dual-based design of RSP to our problem. Second, for the well-studied Vehicle Routing Problem with Time Windows (VRPTW) [27]–[30], if we only consider one vehicle and one customer with departure deadline, then it becomes the RSP problem, which is a special case of our problem without speed planning. Third, our problem can be regarded as a special case of the studied problems.
in [31] and [32] under different contexts from our focus on trucks, where both route planning and speed planning are considered. However, [31], [32] do not prove the hardness of the problem and only propose a heuristic algorithm without performance guarantee. The performance of these generic approaches can be quite unsatisfactory in some specific problems. For example, column generation approach suffers from slow convergence and thus terminating in certain iterations could produce a solution far away from the optimal one [39], [40], and multi-start local search could also get trapped in a bad local optimum [41].

II. Model and Problem Formulation

A. System Model

Consider a highway transportation network as exemplified in Fig. 1. We model it as a directed graph \( G = (V, E) \), where \( V \) is the vertex/node set and \( E \) is the edge/road set. We define \( n \triangleq |V| \) as the number of nodes and \( m \triangleq |E| \) as the number of edges. For each edge \( e \in E \), we denote \( D_e > 0 \) as its distance (unit: mile), and \( R_e^{lb} > 0 \) (resp. \( R_e^{ub} \geq R_e^{lb} \)) as its minimum (resp. maximum) speed (unit: mph). (Governments usually set the maximum speed for all highways and the minimum speed for some highways. For the sake of both safety and fuel efficiency, lower speed limits than passenger cars may be applied to large commercial vehicles like heavy-duty trucks and buses.) Now consider a long-haul heavy-duty truck at time 0 who aims to ship cargos from a source node \( s \in V \) to a destination node \( d \in V \). The goal is to minimize the energy/fuel consumption subject to a hard deadline requirement \( T > 0 \) (unit: hour).

Fuel consumption and travel delay are usually in conflict with each other, both of which are related to the speed profile of the truck. High travel speed obviously decreases the travel delay, but it can also increase the fuel consumption significantly [18], [19]. To analyze the performance tradeoff between energy and delay, we need to model the relationship between the fuel consumption and the travel speed. There are an intensive number of such models (see a survey in [42]). In this paper, we use the instantaneous fuel consumption model [42], [43] which generally depends on three factors: (i) static vehicle/road/environment properties, (ii) instantaneous acceleration/deceleration, and (iii) instantaneous speed. As we consider a specific vehicle running over a specific network, static vehicle/road/environment properties are fixed, and thus we model them as fixed parameters in our fuel consumption model. We further neglect the effects of instantaneous acceleration/deceleration based on the following two observations. First, as shown in [44] and [45] and our results in Lemma 1, running at a constant speed is most fuel-economic on a road segment with homogeneous grade and road/environment conditions. Thus, it is reasonable to ignore the effects of acceleration/deceleration inside any road segment (with homogeneous grade and road/environment conditions). Second, while a truck may involve acceleration/deceleration when switching from one road segment to another road segment, the acceleration/deceleration distance is negligible as compared to the length of road segments. For example, as shown in [46], the heavy-duty truck can accelerate from zero speed to 31 mph in just 500 feet, while the average length of highway road segments is 3.26 miles, according to our study of the U.S. national highway data (see Tab. IV). Thus, it is also reasonable to ignore the effects of acceleration/deceleration during road segment switch. With the above justification, in this paper, we assume that the instantaneous fuel consumption is a function of the instantaneous speed.

We thus define \( f_e : [R_e^{lb}, R_e^{ub}] \rightarrow \mathbb{R}^+ \) as the (instantaneous) fuel-rate-speed function of the truck running on edge \( e \); if the vehicle’s speed on edge \( e \) is \( r_e \) (unit: mph), the fuel consumption rate is \( f_e(r_e) \) (unit: gallons per hour (gph)), and then the total fuel consumption for driving time \( t \) (unit: hour) with the constant speed \( r_e = f_e(r_e) \cdot t \) (unit: gallon). Since many existing models [43], [47]–[50] use polynomial functions to model the fuel consumption which are also strictly convex in a reasonable speed limit region, in this paper, we assume that \( f_e(\cdot) \) is a polynomial function and is strictly convex\(^2\) over \([R_e^{lb}, R_e^{ub}]\). This assumption also holds in the physical interpretation of fuel-rate-speed function as shown in Appendix A in the supplementary materials.

---

1We interchangeably use fuel and energy in this paper.

2The strict convexity can be relaxed to convexity. For simplicity, we use the strict convexity in this paper.
and is further verified in our simulation using real-world data (see Fig. 5(a)).

B. Problem Formulation

We consider two design spaces: path selection (route planning) and speed optimization (speed planning). For path selection, we define a binary variable \( x_e \) for any \( e \in E \),

\[
    x_e = \begin{cases} 
    1, & \text{Edge } e \text{ is on the selected path}; \\
    0, & \text{otherwise}. 
    \end{cases}
\]  

(1)

For the speed optimization, the truck needs to determine a speed profile (speeds at all travel time) over any selected edge. This is a functional variable, but the convexity of fuel-rate-speed function can simplify the speed profile significantly based on the following lemma.

**Lemma 1.** For any edge \( e \), if the travel time \( t_e \) is given, i.e., the truck must pass edge \( e \) with exactly \( t_e \) hours, then the optimal speed profile to minimize the fuel consumption is to maintain constant speed \( D_e/t_e \) during the whole trip.

**Proof:** See Append. B in the supplementary materials.

**Lemma 1** shows that for any edge, any non-constant speed profile is dominated by another constant speed profile in terms of fuel consumption without sacrificing the delay performance. Therefore, without loss of optimality, the truck only needs to follow a constant speed for any edge. As explained in Sec. II-A, since we consider a long-haul highway scenario, we will ignore the speed transition period between two adjacent edges. Thus, for the speed optimization, we consider the travel time \( t_e > 0 \) over each edge \( e \) as the design variable, which equivalently implies a constant speed \( D_e/t_e \) over each edge. We then define a fuel-time function \( c_e(\cdot) \) for each road \( e \),

\[
    c_e(t_e) \triangleq t_e \cdot f_e(D_e/t_e),
\]  

(2)

which is the total fuel consumption for the truck traveling edge \( e \) with travel time \( t_e \).

By vectorizing our decision variables as \( x \triangleq \{ x_e : e \in E \} \) and \( t \triangleq \{ t_e : e \in E \} \), now we are ready to formulate our Path selection and Speed Optimization (PASO) problem,

\[
\text{PASO:} \quad \min_{x \in X, t \in T} \sum_{e \in E} x_e \cdot c_e(t_e) \quad \text{s.t.} \quad \sum_{e \in E} x_e t_e \leq T, 
\]  

(3)

where \( t \leq T \) is the hard path deadline requirement. Clearly, our problem PASO generalizes RSP where we allow a varying edge cost and edge time because of the design space of speed optimization. Since RSP is NP-Complete [25], we can thus easily prove that our problem PASO is also NP-Complete.

**Theorem 1:** PASO is NP-Complete.

**Proof:** We can prove it by setting \( R_e^{lb} = R_e^{ub} \) to an appropriate value for each edge \( e \) in PASO, and using the result that RSP is NP-Complete [25].

**Theorem 1** shows that exact solution is computational prohibited unless \( P=NP \). In this paper, we thus seek approximate but efficient solutions to PASO.

C. Complexity Hardness

PASO has both integer variables and continuous variables. Thus it is worth understanding its hardness first. It turns out that a special case of PASO is the well-known Restricted Shortest Path (RSP) problem [24], [25]. In RSP, a directed graph is given where each edge has a fixed travel time and travel cost, and the goal is to find a minimum-cost path subject to a hard path deadline requirement. Clearly, our problem PASO generalizes RSP where we allow a varying edge cost and edge time because of the design space of speed optimization. Since RSP is NP-Complete [25], we can thus easily prove that our problem PASO is also NP-Complete.

**Theorem 1:** PASO is NP-Complete.

**Proof:** We can prove it by setting \( R_e^{lb} = R_e^{ub} \) to an appropriate value for each edge \( e \) in PASO, and using the result that RSP is NP-Complete [25].

**Theorem 1** shows that exact solution is computational prohibited unless \( P=NP \). In this paper, we thus seek approximate but efficient solutions to PASO.

D. Preprocessing and Some Notations

We first check the feasibility of our problem PASO. We can use the shortest path algorithm where each edge \( e \) has cost \( t_e^{lb} \) to find the fastest path. If the travel time of the fastest path is larger than the deadline requirement \( T \), PASO is infeasible. In the rest of this paper, we thus assume that the deadline constraint \( T \) is at least the travel time of the fastest path such that the problem is feasible.

We then analyze properties of the fuel-time function \( c_e(\cdot) \).

**Lemma 2:** \( c_e(t_e) \) is strictly convex over \([t_e^{lb}, t_e^{ub}]\). Also, there exists a point \( \hat{t}_e \in [t_e^{lb}, t_e^{ub}] \) such that \( c_e(t_e) \) is first strictly decreasing over \([t_e^{lb}, \hat{t}_e]\) and then strictly increasing over \([\hat{t}_e, t_e^{ub}]\).

**Proof:** See Append. C in the supplementary materials.

Based on Lemma 2, we can easily prove that the travel time over edge \( e \), i.e., \( t_e \), in any optimal solution of PASO must be in the region \([t_e^{lb}, \hat{t}_e]\). Otherwise, we can decrease the travel time from \( t_e \) to \( \hat{t}_e \) and at the same time decrease the fuel consumption, which violates the optimality of \( t_e \). Thus, without loss of optimality, we can reset the speed limit from \([t_e^{lb}, \hat{t}_e]\) to \([\hat{t}_e, \hat{t}_e]\), which equivalently implies that we reset the speed limit from \([t_e^{lb}, t_e^{ub}]\) to \([\hat{t}_e, t_e^{ub}]\). After such preprocessing, in the rest of the paper, \( c_e(t_e) \) can be assumed to be strictly convex and strictly decreasing over \([\hat{t}_e, t_e^{ub}]\) without loss of optimality.

In the rest of the paper, define an \( s-d \) path \( p \) as the set of all edges over \( p \) and \( t_p \triangleq \{ t_e : e \in p \} \) as the corresponding
travel time set. Moreover, we define \( c(p, t_p) = \sum_{e \in p} c_e(t_e) \) as the fuel consumption of path \( p \) with travel time set \( t_p \), and \( \text{OPT} \) as the optimal value of \( \text{PASO} \).

Next, we will propose a fully polynomial time approximation scheme (FPTAS) in Sec. III and a fast dual-based heuristic scheme in Sec. IV to solve our problem \( \text{PASO} \).

III. AN FPTAS FOR \( \text{PASO} \)

Since \( \text{PASO} \) generalizes \( \text{RSP} \), which is well-known to have an FPTAS [24], [51], it is natural to ask whether we can extend \( \text{RSP} \)’s FPTAS for our problem \( \text{PASO} \). In this section, by carefully tackling the difference between \( \text{PASO} \) and \( \text{RSP} \), we “reformulate” \( \text{PASO} \) such that we can adapt \( \text{RSP} \)’s FPTAS to construct an FPTAS for \( \text{PASO} \). More specifically, in this section, we propose an approximation scheme (Algorithm 3) such that for any given \( \epsilon \in (0, 1) \), it can find a \((1 + \epsilon)\)-approximate solution in the sense that the solution is feasible and the corresponding fuel consumption is at most \((1+\epsilon)\text{OPT}\), and the time complexity is polynomial in both the problem size and \( \epsilon \).

The essence of \( \text{RSP} \)’s FPTAS [24], [51] is a test procedure. For any input value \( S \) and any input accuracy parameter \( \delta > 0 \), the test procedure can “approximately” compare \( S \) and the optimal value \( \text{OPT} \) in the sense that it can tell either \( \text{OPT} > S \) or \( \text{OPT} \leq (1+\delta)S \) in polynomial time. Based on this test procedure, starting with some arbitrary lower bound \( \text{LB} \) and upper bound \( \text{UB} \) for \( \text{OPT} \), a binary search scheme is designed [24], [51] to exponentially narrow down the bounding interval \([\text{LB}, \text{UB}]\) and finally a \((1 + \epsilon)\)-approximate solution is outputted.

To solve our problem \( \text{PASO} \), we adapt \( \text{RSP} \)’s FPTAS by designing our own test procedure. In \( \text{RSP} \), [24] and [51] use the "rounding and scaling" technique, where each fixed edge cost is rounded into certain (polynomial) number of cost levels controlled by the accuracy parameter \( \delta \). As we only require an “approximate” comparison, rounding into certain number of cost levels is enough to perform such a task. However, as opposed to a fixed edge cost in \( \text{RSP} \), in \( \text{PASO} \) each edge has a fuel-time function. Hence, instead of rounding a fixed cost in \( \text{RSP} \), we quantize the continuous fuel-time function \( c_e(t_e) \) into another staircase fuel-time function \( \tilde{c}_e(t_e) \) according to the input value \( S \) and the input accuracy parameter \( \delta \), which can be further characterized by a polynomial number of representative points. We then prove that such quantization can perform the “approximate” comparison.

Later on we will describe our algorithms in a bottom-up fashion. We first describe the quantizing procedure (Algorithm 1) in Sec. III-A. Then we present our own test procedure (Algorithm 2) which invokes Algorithm 1 in Sec. III-B. Finally, we describe the whole FPTAS (Algorithm 3) which invokes Algorithm 2 in Sec. III-C.

A. Quantizing Fuel-Time Function

For any input value \( V > 0 \) and \( N \in \mathbb{Z}^+ \), we quantize the edge-\( e \) fuel-time function \( c_e(t_e) \) to be

\[
\tilde{c}_e(t_e) = \min \left\{ \frac{c_e(t_e)}{V} + 1, N \right\}, \quad \forall t_e \in [t_e^\text{lb}, t_e^\text{ub}].
\]

(5)

Since we have assumed that \( c_e(t_e) \) is strictly decreasing in Sec. II-D, \( \tilde{c}_e(t_e) \) thus becomes a staircase function with at most \( N \) stairs. During the quantization, parameter \( V \) is to control the accuracy, which is the vertical span of each stair. Larger \( V \) means rougher quantization and lower accuracy but smaller complexity. Parameter \( N \) is to control the maximum number of stairs. Since \( \tilde{c}_e(t_e) \) could take an arbitrarily large value, the number of stairs could be unbounded, which definitely incurs high complexity. To design a polynomial time test procedure where we only need to perform an “approximate” comparison, we truncate \( c_e(t_e) \) by putting a cell \( V, N \). This truncation is sufficient for use in the test procedure (see Sec. III-B). Clearly, \( \tilde{c}_e(t_e) \) is a quantized and truncated version of \( c_e(t_e) \). An example is shown in Fig. 2. Here we set \( V = 20, N = 4 \). Thus, each stair spans 20 and \( \tilde{c}_e(t_e) \) is truncated by the cell \( V, N = 80 \). The resulting curve \( \tilde{c}_e(t_e) \) is a non-increasing staircase function, which jumps from 4 to 3 at \( t_e = 1.8 \) and jumps from 3 to 2 at \( t_e = 2.8 \).

Moreover, since \( \tilde{c}_e(t_e) \) is a staircase function and only takes integer values, we can use an \( N \)-dim vector \( \tau_e \) to represent it without any information loss. We define it as \( \tau_e = (\tau_{e,1}, \tau_{e,2}, \ldots, \tau_{e,N}) \) where \( \tau_{e,i} \) is the minimum travel time over \([t_e^\text{lb}, t_e^\text{ub}]\) such that \( \tilde{c}_e(\tau_{e,i}) = i \) and is defined as \( \text{nan} \) if \( \tilde{c}_e(\tau_{e,i}) \neq i \) has no solution. For the example in Fig. 2, we have \( \tau_e = (1, 1, 1, 1) \).

We call \( (\tau_{i,e}, i) \) the \( i \)-th representative point of \( \tilde{c}_e(\cdot) \). Thus \( \tilde{c}_e(\cdot) \) is characterized by at most \( N \) representative points, which will play a key role in our test procedure in Sec. III-B.

We summarize the quantizing procedure \( \text{QUANTIZE}(\epsilon, V, N) \) in Algorithm 1. The basic idea is to first find the range of the stair levels, i.e., \([n_{\text{min}}, n_{\text{max}}]\) and then find \( \tau_{e,i} \) for any level \( i \) in this range by solving an equation \( c_e(t_e) = iV \).

1) Time Complexity: (i) When \( n_{\text{min}} = n_{\text{max}} \) (e.g., if \( t_e^\text{ub} = t_e^\text{lb} \)), the loop in lines 7-12 will not be executed. Thus, the total complexity of \( \text{QUANTIZE}(\epsilon, V, N) \) is \( O(N) \) due to the initial loop in lines 1-3. (ii) When \( n_{\text{min}} < n_{\text{max}} \), we need to solve an equation for each \( i \) in the range \([n_{\text{min}}, n_{\text{max}} - 1]\) as shown in line 8. Since we have assumed that \( c_e(t_e) \) is a strictly decreasing function, we can use a binary search to solve this equation, which has time complexity \( O \left( \log \frac{n_{\text{ub}} - n_{\text{lb}}}{\text{tol}} \right) \) where \( \text{tol} \) is the tolerance level for termination. The total complexity of \( \text{QUANTIZE} \)
As we mentioned before, the major difference between our problem \( \text{PASO} \) and the existing problem \( \text{RSP} \) is that \( \text{PASO} \) has a continuous fuel-time function for each edge instead of a fixed cost. Thus, different from the test procedure for \( \text{RSP} \) (see [51, Fig. 1]), we have a step to invoke the quantizing procedure (Algorithm 1) to quantize the fuel-time function, as shown in lines 3-5 in Algorithm 2. More importantly, since our test procedure \( \text{TEST}(L, U, \delta) \) aims to check either \( \text{OPT} > U \) or \( \text{OPT} \leq U + \delta L \), roughly speaking, we do not need to quantize the portion of each fuel-time function with high fuel cost, i.e., larger than \( U + \delta L \). Hence, to ensure polynomial time complexity eventually, we put a ceil \( V(N + 1) \) for \( c_e(t_e) \) as shown in line 4 of the algorithm, where \( V \) and \( N \) are appropriately set such that \( V(N + 1) \geq U + \delta L \).

After such quantization, the fuel-time function \( c_e(t_e) \) for each edge \( e \) consists of at most \( N + 1 \) representative points. Therefore, conceptually we can construct a new graph \( \hat{G} = (\hat{V}, \hat{E}) \). Each edge \( e \in \hat{E} \) in the original graph corresponds to at most \( N + 1 \) edges in the new graph \( \hat{E} \). For each edge \( e \in \hat{E} \), the edge cost \( \hat{c}_e \) is a positive integer, as shown in (5). This is exactly an \( \text{RSP} \) problem. Therefore, the remaining steps follow the test procedure for \( \text{RSP} \) on the new graph \( \hat{G} \). Specifically, since each edge \( e \in \hat{E} \) has at most \( N + 1 \) possible cost values all of which are positive integers (each edge \( e \) in the new graph \( \hat{E} \) has a positive integer cost), we can use dynamic programming to complete such test. Similar to [24] and [51], we define \( g_e(c) \) as the minimum path travel time among all \( s - v \) paths whose path cost is at most \( c \in \mathbb{Z}^+ \), and define \( g_e(c) = \infty \) if no such path. The optimality condition (or Bellman’s equation) becomes, for any \( c = 1, 2, \ldots \),

\[
\min_{u, i : e = (u, v) \in \hat{E}, e \neq \text{nan}} \{g_u(c - i) + \hat{c}_e\} \tag{7}
\]

which is shown in line 10 in Algorithm 2. Since we only need to answer either \( \text{OPT} > U \) or \( \text{OPT} \leq U + \delta L \), we do not have to process large \( c \). Instead, iterating \( c \) from 1 to \( N \) is enough for us to complete this task. This dynamic programming procedure is shown in lines 6-15 of Algorithm 2.

In \( \text{PASO} \), we should carefully design the quantizing and the dynamic programming procedures jointly to guarantee performance, as shown in the following lemmas, which are the counterparts to Lemma 2 and Lemma 3 for \( \text{RSP} \) in [51].
Proof: See Append. D in the supplementary materials. □

Lemma 4: If $U \geq OPT$, then Algorithm 2 must return a feasible path $p$ and travel time set $t_p$, which satisfy
\[
c(p,t_p) \leq OPT + L \delta. \quad (9)
\]

Proof: See Append. E in the supplementary materials. □

Lemma 5: If Algorithm 2 returns FAIL, then we have
\[
OPT > U. \quad (10)
\]

Proof: This directly follows Lemma 4. □

Our test procedure either returns a path $p$ and travel time set $t_p$ in line 13, which implies that $OPT \leq U + L \delta$ from Lemma 3, or returns FAIL in line 16, which implies $OPT > U$ from Lemma 5. Therefore, Lemma 3 and Lemma 5 justify that our test procedure (Algorithm 2) completes the “approximate” comparison, i.e., answers either $OPT > U$ or $OPT \leq U + L \delta$.

Thus, for the purpose of the test procedure, Lemma 3 and Lemma 5 are enough. However, we present Lemma 4, which is stronger than Lemma 5, to provide a sufficient condition such that our test procedure returns a path $p$ and travel time set $t_p$. We will use Lemma 4 shortly in Sec. III-C to finally output a $(1 + \epsilon)$-approximate solution.

1) Time Complexity: The quantizing procedures for all edges in lines 3-5 require $O(mn \log \xi)$. The dynamic programming procedure in lines 6-15 requires $O(mn^2)$. Since $N = \lceil \frac{U}{L} \rceil + n + 1 = \lceil \frac{UB}{LB} \rceil + n + 1 = O(\frac{UB}{LB} \cdot \frac{n}{\xi} + n)$, the total time complexity of Algorithm 2 is $O(mn \log \xi + mn^2) = O(m(\frac{UB}{LB} \cdot \frac{n}{\xi} + n) \log \xi + m(\frac{UB}{LB} \cdot \frac{n}{\xi} + n)^2)$.

C. The Proposed FPTAS

Based on our own test procedure (Algorithm 2), we then follow the FPTAS for RSP in [51, Fig. 2] by replacing its test procedure with ours. For completeness, we present the FPTAS in Algorithm 3 and explain it with the following three steps.

**Algorithm 3** An FPTAS

1: Get a lower bound $LB$ and upper bound $UB$ for $OPT$
2: Set $BL_1 = LB$
3: Set $BU_1 = UB$
4: while $\frac{BU_1}{BL_1} > 16$ do
5: $S = \sqrt{BL_1 \cdot BU_1}$
6: Call $TEST(S, S, 1)$
7: if $TEST(S, S, 1)$ returns FAIL then
8: Set $BL_1 = S$
9: else
10: Set $BU_1 = 2S$
11: end if
12: end while
13: Call $TEST(BL_1, BU_1, \epsilon)$

Step 1 (line 1): To initialize the bound interval, we need to first obtain a lower bound $LB$ and an upper bound $UB$ for the optimal value $OPT$. Define that the minimum single-edge fuel cost is $C^b \triangleq \min_{e \in E} c_e(t_{1e}^{UB})$ and the maximum single-edge fuel cost is $C^b \triangleq \max_{e \in E} c_e(t_{1e}^{LB})$. Simply, we can use the minimum single-edge fuel consumption $C^b$ as the lower bound $LB$ and use the maximum single-path fuel consumption $nC^b$ as the upper bound $UB$. Also, in Sec. IV, we will propose a heuristic scheme which can always output a set of $LB$ and $UB$.

Step 2 (lines 2-12): Using the initial lower bound $LB$ and upper bound $UB$, we design a binary search scheme, which repeatedly invokes our test procedure (Algorithm 2) to exponentially narrow down the bound interval $[BL_1, BU_1]$ until $BU_1/BL_1 \leq 16$. The binary search step is visualized in Fig. 3. Note that we always keep $BL_1$ as a lower bound and $BU_1$ as an upper bound for $OPT$. Whenever $BU_1/BL_1 > 16$, we input the geometric mean $S = \sqrt{BL_1 \cdot UB_1}$ and $\delta = 1$ to the test procedure, as shown in lines 5 and 6. If $TEST(S, S, 1)$ returns FAIL, then according to Lemma 4, we must have $S < OPT$. In this case, we reset the lower bound $BL_1$ to be $S$ in line 8. Otherwise, $TEST(S, S, 1)$ returns a feasible path $p$ and travel time set $t_p$. According to Lemma 3, we must have $OPT \leq S + \delta S = 2S$. We reset the upper bound to be $2S$ in line 10. It can be easily shown that this binary search returns a lower bound $BL_1$ and an upper bound $UB_1$ for $OPT$ such that $UB_1/BL_1 \leq 16$ in $O(\log \log \frac{UB_1}{BL_1})$ iterations.

Step 3 (line 13): When $\frac{BU_1}{BL_1} \leq 16$, we call our test procedure again but we use $L = BL_1$ and $U = BU_1$ and $\delta = \epsilon$. Since $BU_1 \geq OPT$, according to Lemma 4, $TEST(BL_1, BU_1, \epsilon)$ must return a feasible path $p$ and travel time $t_p$ such that
\[
c(p,t_p) \leq OPT + \epsilon BL_1 \leq OPT + \epsilon OPT = (1 + \epsilon)OPT.
\]

Therefore, we get a $(1 + \epsilon)$-approximate solution to PASO.

1) Time Complexity: $Step 1$ requires $O(m)$ to get an initial lower bound $LB$ and upper bound $UB$. $Step 2$ invokes the test procedure $O(\log \log \frac{UB_1}{BL_1})$ times and each invoke takes $O(m \log \xi + mn^2)$ time by using $L = U = S$ and $\delta = 1$. Thus $Step 2$ takes $O((mn \log \xi + mn^2) \log \log \frac{UB_1}{BL_1})$. $Step 3$ also invokes the test procedure, and it takes $O(\frac{mn \log \xi + mn^2}{\epsilon})$ time by using $\delta = \epsilon < 1$ and $O(\frac{UB_1}{LB_1}) = O(\frac{BU_1}{BL_1}) = O(1)$ because $\frac{BU_1}{BL_1} \leq 16 = O(1)$. Here we can also see why we need to use a binary search to obtain $\frac{BU_1}{BL_1} \leq 16$ in $Step 2$. This is because $\frac{BU_1}{BL_1} = O(1)$ ensures polynomial time complexity in $Step 3$. Therefore, the total complexity is $O((mn \log \xi + mn^2) \log \log \frac{UB_1}{LB_1} + \frac{mn \log \xi + mn^2}{\epsilon})$. \footnote{A simple path can have at most $n$ edges.}
We summarize our results for the approximate scheme in the following theorem.

**Theorem 2.** Algorithm 3 returns a \((1 + \epsilon)\)-approximate solution for PASO in time \(O((mn \log \lambda + n^2) \log \log \frac{UB}{\epsilon} + \frac{mn \log \xi + m^2}{A})\). In addition, under our assumption that any edge-\(e\) fuel-rate-speed function \(f_e(\cdot)\) is a polynomial function (see Sec. II-A), we use \(LB = C^Lb\) and \(UB = nC^ub\) where \(C^Lb \triangleq \min_{e \in E} c_e(t_e^{Lb})\) and \(C^ub \triangleq \max_{e \in E} c_e(t_e^{Lb}) = c_e(t_e^{Lb})\), we have \(\log \frac{UB}{\epsilon} = \max(O(\log \log n), O(I_e))\) where \(I_e\) is the input size of all parameters of edge \(e\). Thus, Algorithm 3 has time complexity polynomial in the input size of the problem PASO and \(1/\epsilon\) and therefore is an FPTAS.

**Proof:** See Append. F in the supplementary materials.

Although we generalize the FPTAS design from RSP to PASO, such an FPTAS (Algorithm 3) still has high complexity for a large-scale highway network with tens of thousands of nodes and edges. In the next section, we propose a heuristic scheme with substantially lower complexity.

### IV. A Fast Dual-Based Heuristic

In this section, we present a heuristic scheme for our problem PASO based on Lagrangian relaxation. Such a heuristic scheme, as we will show later in Sec. IV-C, runs much faster than the FPTAS (Algorithm 3). Also, it always outputs a lower bound \(LB\) and an upper bound \(UB\) on OPT, which implements Step 1 in Algorithm 3. Moreover, in most practical scenarios as shown in Sec. V, this heuristic scheme outputs an optimal (or at least near optimal) solution, i.e., \(LB = UB = OPT\) or at least \(LB \approx OPT \approx UB\).

#### A. Lagrangian Relaxation and Dual Problem

In our problem PASO, since the hard deadline constraint (4) couples path selection variable \(x\) with speed optimization variable \(t\), we relax it and introduce a Lagrangian dual variable \(\lambda \geq 0\), which can be interpreted as a (per-unit) delay price over the entire network.

Based on this relaxation, we can get the corresponding Lagrangian,

\[
L(x, t, \lambda) \triangleq \sum_{e \in E} x_e \cdot c_e(t_e) + \lambda \sum_{e \in E} x_e t_e - T
\]

\[
= \sum_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e) - \lambda T, \quad (11)
\]

and the corresponding dual function is defined as \(D(\lambda) \triangleq \min_{x \in X, t \in T} L(x, t, \lambda)\). Then the dual problem of PASO is formulated as

\[
(\text{PASO-Dual}) \quad \max_{\lambda \geq 0} D(\lambda)
\]

#### B. Obtain Dual Function

Before we solve the dual problem, let us first show how to obtain the dual function for a given \(\lambda\) as follows,

\[
D(\lambda) = \min_{x \in X, t \in T} L(x, t, \lambda)
\]

\[
= -\lambda T + \min_{x \in X, t \in T} \sum_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e)
\]

\[
= -\lambda T + \min_{x \in X} \sum_{t \in T} \left[ \sum_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e) \right]
\]

\[
= -\lambda T + \min_{x \in X} \sum_{t \in T} \left[ \min_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e) \right]
\]

\[
= -\lambda T + \min_{x \in X} \sum_{t \in T} \left[ \min_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e) \right]
\]

\[
= -\lambda T + \sum_{t \in T} \left[ \min_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e) \right]
\]

\[
= -\lambda T + \min_{x \in X} \sum_{t \in T} \left[ c_e(t_e^{L}(\lambda)) + \lambda t_e^{L}(\lambda) \right]
\]

\[
= -\lambda T + \min_{x \in X} \sum_{t \in T} \left[ c_e(t_e^{L}(\lambda)) + \lambda t_e^{L}(\lambda) \right]
\]

\[
= -\lambda T + \sum_{e \in T^*} w_e(\lambda).
\]

(12)

We explain \((E_1) - (E_5)\) in (12) one by one. Equality \((E_1)\) is because no coupled constraints exist for \(x\) and \(t\). Equality \((E_2)\) is because no coupled constraints exist for the travel time at different edges in \(T\).

In equality \((E_3)\), \(t_e^{L}(\lambda)\) is defined as

\[
t_e^{L}(\lambda) \triangleq \arg \min_{t_e \in [c_e(t_e^{L}(\lambda)) + \lambda t_e^{L}(\lambda)]} c_e(t_e) + \lambda t_e.
\]

Note that since we have assumed that \(c_e(t_e)\) is strictly convex and strictly decreasing over \([c_e(t_e^{L}(\lambda)) + \lambda t_e^{L}(\lambda)]\) in Sec. II-D, \(t_e^{L}(\lambda)\) is unique and thus \((13)\) is well defined. Specifically, \(t_e^{L}(\lambda)\) can be obtained analytically as follows.

**Lemma 6.** Define \(c_e^{-1}(\cdot)\) as the inverse function of \(c_e^{-1}(\cdot)\). Then we have

\[
t_e^{L}(\lambda) = \begin{cases} t_e^{L}(\lambda), & \text{if } 0 \leq \lambda < c_e^{-1}(t_e^{L}(\lambda)); \\ c_e^{-1}(\lambda), & \text{if } c_e^{-1}(t_e^{L}(\lambda)) \leq \lambda \leq c_e^{-1}(t_e^{L}(\lambda)); \\ \lambda, & \text{if } \lambda > c_e^{-1}(t_e^{L}(\lambda)). \end{cases}
\]

(14)

**Proof:** See Append. G in the supplementary materials.

Now let us consider the complexity of computing \(t_e^{L}(\lambda)\) based on Lemma 6. Since we assume that \(f_e(\cdot)\) is a polynomial function in Sec. II-A and we define \(c_e(t_e) = t_e \cdot f_e(T_e / t_e)\) in (2), then we can easily evaluate \(c_e^{-1}(\cdot)\) for any \(t_e\). Thus, we can first determine the region that \(\lambda\) belongs to. Then,

- If \(0 \leq \lambda < c_e^{-1}(t_e^{L}(\lambda))\), we obtain \(t_e^{L}(\lambda) = t_e^{L}(\lambda)\) with time complexity \(O(1)\).
- If \(\lambda > c_e^{-1}(t_e^{L}(\lambda))\), we obtain \(t_e^{L}(\lambda) = \lambda\) with time complexity \(O(1)\).
- If \(c_e^{-1}(t_e^{L}(\lambda)) \leq \lambda \leq c_e^{-1}(t_e^{L}(\lambda))\), however, we cannot directly get \(c_e^{-1}(\lambda)\) because the inverse function \(c_e^{-1}(\cdot)\) is not easy to evaluate. Instead of directly evaluating the inverse function, we find a \(t_e\) such that \(c_e(t_e) = \lambda\) and such a \(t_e\) becomes \(t_e^{L}(\lambda)\). Since \(c_e^{-1}(\cdot)\) is a strictly increasing function due to the strict convexity of \(c_e(\cdot)\), numerically we can design a binary search scheme to obtain \(t_e^{L}(\lambda)\), whose time complexity is \(O\left(\log \frac{UB - LB}{\epsilon} \right) = O(\log \xi)\).

Overall, the time complexity to obtain \(t_e^{L}(\lambda)\) is \(O(\log \xi)\).

In addition, (13) has a nice economic interpretation. As we have relaxed the hard deadline constraint, we penalize each edge \(e\) with a delay cost, which is the product of the travel time \(t_e\) and the (per-unit) delay price \(\lambda\). Then for a given delay price \(\lambda\), each edge selects the optimal travel time to minimize its *generalized* cost, including both fuel cost \(c_e(t_e)\) and delay cost \(\lambda t_e\). Thus, \(t_e^{L}(\lambda)\) is the best response of edge \(e\) for a given delay price \(\lambda\).
In equality (E₄), \( w_e(\lambda) \) is defined as

\[
  w_e(\lambda) \triangleq c_e(t^*_e(\lambda)) + \lambda t^*_e(\lambda),
\]

which can be interpreted as the minimum generalized cost (including both fuel cost and delay cost) of edge \( e \) for a given delay price \( \lambda \). Obviously, \( w_e(\lambda) \) is the generalized cost under the best response \( t^*_e(\lambda) \).

In equality (E₅), since \( \lambda \) restricts that an \( s-d \) path is selected, \( \min_{x \in X} \sum_{e \in e \cdot w_e(\lambda)} \) is exactly a shortest path problem where each edge \( e \) has a generalized cost \( w_e(\lambda) \). We define \( p^*(\lambda) \) as the resulting shortest-generalized-cost path.

In summary, (12) shows that for any dual variable \( \lambda \), we only need to solve a shortest path problem to obtain the dual function value \( D(\lambda) \), which is much easier than PASO.

C. The Heuristic Algorithm

Our heuristic scheme relies on one key observation. Define

\[
  \delta(\lambda) \triangleq \sum_{e \in p^*(\lambda)} t^*_e(\lambda),
\]

which is the total travel time of the resulting shortest-generalized-cost path \( p^*(\lambda) \) for a given \( \lambda \). Our key observation is the following theorem (see an example in Fig. 6).

**Theorem 3:** \( \delta(\lambda) \) is non-increasing over \( \lambda \in [0, +\infty) \).

**Proof:** See Append. H in the supplementary materials.

Theorem 3 shows that increasing \( \lambda \) will decrease the total travel time of the selected path based on the best responses of all edges. Intuitively, since \( \lambda \) can be interpreted as a delay price, increasing \( \lambda \) will force all edges to select a shorter travel time and further force the resulting shortest-generalized-cost path to have a shorter travel time.

Based on Theorem 3, we can use a simple dual variable \( \lambda \) to coordinate the total travel time. For example, when \( \delta(\lambda) > T \), we can increase \( \lambda \) such that \( \delta(\lambda) \) can be decreased to finally satisfy the hard deadline requirement. On the other hand, when \( \delta(\lambda) < T \), it means that the truck travels very fast and there still exists some room to increase the travel time and thus decrease the fuel consumption. Then we decrease \( \lambda \) such that \( \delta(\lambda) \) can be increased to reach \( T \). This is called a coordination mechanism [52, Ch. 5.1.6]. Therefore, we aim to find a \( \lambda_0 \) such that \( \delta(\lambda_0) = T \). However, our problem PASO is not convex but has a combinatorial difficulty. Thus it is not guaranteed to find such a \( \lambda_0 \). We thus call our binary search for \( \lambda_0 \) (Algorithm 4) a heuristic scheme.

In Algorithm 4, we first set an initial lower bound \( \lambda_L = 0 \) and an initial upper bound \( \lambda_U = \lambda_{\text{max}} \) for the targeted \( \lambda_0 \). In practice, since we are considering the fuel consumption and \( \lambda \) can be interpreted as a delay price, \( \lambda_{\text{max}} \) can be reasonably set to be an upper bound of the fuel consumption per hour. In our simulation in Sec. V, we set \( \lambda_{\text{max}} = 100 \), which works for all settings. Then we do binary search in lines 3-19, where tol in line 3 is the tolerance level for termination which is close to zero. During the binary search, based on the non-increasing property of \( \delta(\lambda) \) (Theorem 3), we keep updating the lower bound \( \lambda_L \) and its corresponding solution \( (p^*(\lambda_L), \{t^*_e(\lambda_L) : e \in p^*(\lambda_L)\}) \), as well as the upper bound \( \lambda_U \) and its corresponding solution \( (p^*(\lambda_U), \{t^*_e(\lambda_U) : e \in p^*(\lambda_U)\}) \).

This algorithm has two possible results:

\( \triangleright \) **Case 1:** If it returns in line 9, then we have found a \( \lambda_0 \) such that \( \delta(\lambda_0) = T \). We prove that the returned solution is optimal for PASO in Theorem 4.

\( \triangleright \) **Case 2:** If it returns in line 20, then we have found a \( \lambda_0 \) such that \( \delta(\lambda_L) > T \) and \( \delta(\lambda_U) < T \). With a small enough tolerance level tol, \( \lambda_L = \lambda_0 - \text{tol}/2 \rightarrow \lambda_0^\ast \). Roughly speaking, this means that \( \delta(\lambda) \) is not continuous at \( \lambda = \lambda_0 \). Although this return does not guarantee optimality, we prove in Theorem 5 that the returned solutions \( (p^*(\lambda_L), \{t^*_e(\lambda_L) : e \in p^*(\lambda_L)\}) \) and \( (p^*(\lambda_U), \{t^*_e(\lambda_U) : e \in p^*(\lambda_U)\}) \) give a lower bound LB and an upper bound UB for OPT, respectively.

**Theorem 4:** If Algorithm 4 returns in line 9, then the returned solution \( (p^*(\lambda_0), \{t^*_e(\lambda_0) : e \in p^*(\lambda_0)\}) \) is an optimal solution of PASO.

**Proof:** See Append. I in the supplementary materials.

As a by-product, Theorem 4 also shows that the strong duality for the combinatorial problem PASO holds in this case. Also, \( \lambda_0 \) is the optimal solution to the dual problem PASO-Dual.

**Theorem 5:** If Algorithm 4 returns in line 20, and define \( \text{LB} \triangleq \sum_{e \in p^*(\lambda_L)} c_e(t^*_e(\lambda_L)) \) and \( \text{UB} \triangleq \sum_{e \in p^*(\lambda_U)} c_e(t^*_e(\lambda_U)) \), then we have \( \text{LB} \leq \text{OPT} \leq \text{UB} \).

**Proof:** See Append. J in the supplementary materials.

The LB and UB returned by Algorithm 4 in line 20 can be used for Step I of Algorithm 3. For the case that Algorithm 4 returns in line 9, we use the returned optimal solution as both a lower bound and an upper bound with LB = UB = OPT.

After such unification, Algorithm 4 always outputs a LB and UB for the optimal solution OPT.
1) Time Complexity: In line 7 in Algorithm 4, we use Dijkstra’s shortest-path algorithm with a min-priority queue, which is the fastest known algorithm for the single-source single-destination shortest path problem with time complexity $O(m + n \log n)$ [53]. Thus, in the while loop, each iteration requires $O(m \log \xi + m + n \log n)$ time. And since the total number of iterations is $O(\log \frac{\max \{\text{ub}, \text{lb}\}}{\lambda})$, Algorithm 4 has complexity $O(m \log \xi + m + n \log n) \log \frac{\max \{\text{ub}, \text{lb}\}}{\lambda}$, much faster than the FPTAS (Algorithm 3). We summarize the complexity result in the following theorem.

Theorem 6: The time complexity of Algorithm 4 is $O((m \log \xi + m + n \log n) \log \frac{\max \{\text{ub}, \text{lb}\}}{\lambda})$.

Remark: A similar dual-based heuristic approach for RSP is proposed in [26]. However, as mentioned in Sec. III, different from RSP, our problem PASO has an extra design space of speed optimization. Therefore, theoretically our contribution in this section is to generalize the dual-based heuristic design from RSP [26] to PASO.

V. PERFORMANCE EVALUATION

In this section, we use real-world data to evaluate the performance of our algorithms. Our objectives are three-fold: (i) collect realistic dataset and model the fuel-rate-speed function, (ii) evaluate and compare the performance of our FPTAS and heuristic, (iii) compare our algorithms with baseline algorithms, including both shortest path algorithm and fastest path algorithm adapted from common practice, and (iv) investigate the energy-deadline tradeoff of long-haul heavy-duty trucks by evaluating how much fuel can be saved by increasing the hard deadline.

A. Dataset

1) Transportation Network: We construct the U.S. National Highway Systems (NHS) from the dataset of Clinched Highway Mapping (CHM) Project [54]. The whole highway network graph file is specified in [55], which consists of 84504 nodes (waypoints) and 89119 (one-direction) edges.

2) Elevation: In this paper, we consider the grade/slope effect when modeling the road-dependent fuel-rate-speed function. To obtain the road grades, we use the Elevation Point Query Service [56] provided by the U.S. Geological Survey (USGS) to query elevations of all nodes in the NHS graph.

3) Speed Limits: We use the historical average speed as the maximum speed $R^\text{ub}_{e}$ for each road $e$. HERE map [57] has put speed detectors over many countries including U.S., and it provides APIs to query location-based real-time speed information. We collect the real-time speed information from HERE map [57] for two weeks and use the average as $R^\text{ub}_{e}$ for each road $e$ in the NHS graph. For the minimum speed limit $R^\text{lb}_{e}$, we manually set it to be $R^\text{lb}_{e} = \min(30, R^\text{ub}_{e})$.

4) Fuel Consumption Data: It is hard for us to get suitable real-world fuel consumption data. In this paper, we instead leverage the widely-used ADVISOR simulator [58] to collect fuel consumption data (see Sec. V-B).

5) Heavy-Duty Truck: Fuel consumption highly depends on the truck type. Another benefit of using ADVISOR is that it also provides some heavy-duty truck configurations. In this simulation, we use the Kenworth T800 Vehicle [59], a Class 8 heavy-duty truck, with 36-ton full load. It is specified in files VEH_KENT800Trailer.m and HeavyTruck_in.m in ADVISOR with the following parameters in Tab. III.

In this simulation, we use 0.4% as the span of a grade level.

TABLE III

<table>
<thead>
<tr>
<th>Truck Parameters (Kenworth T800)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drag Coefficient $c_d$</td>
</tr>
<tr>
<td>Frontal area $A_f$ (m²)</td>
</tr>
<tr>
<td>Gider Mass $G_{id}$ kg</td>
</tr>
<tr>
<td>Cargo Mass $G_{ca}$ kg</td>
</tr>
</tbody>
</table>

Fig. 4. U.S. map and 22 regions.

TABLE IV

<table>
<thead>
<tr>
<th>NETWORK STATISTICS. “O” IS THE ORIGINAL NHS GRAPH, “E” IS THE “EASTERN” GRAPH (TO THE EAST OF 100°W), AND “M” IS THE MERGED ONE. θ IS THE GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>O</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>M</td>
</tr>
</tbody>
</table>

6We replace $\text{vinf}.\text{vehicle}.\text{name}$ by VEH_KENT800Trailer.m.

7In this simulation, we use 0.4% as the span of a grade level.
B. Model Fuel-Rate-Speed Function

We model the fuel-rate-speed function as

\[ f_e(x) = a_0 x^3 + b_0 x^2 + c_0 x + d_0, \forall x \in \mathcal{E} \]  

(17)

Here \( x \) is the speed (unit: mph) and \( f_e(x) \) is the fuel rate consumption (unit: gph (gallons per hour)). Although our model (17) can capture any road-dependent features/factors, e.g., grade, rolling resistance, and air density, etc., we only consider the road grade \( \theta \) in this simulation, which is the major factor for truck fuel consumption [43].

To learn the parameters \( a_0, b_0, c_0 \), and \( d_0 \) in (17) in terms of functions of \( \theta \), we use ADVISOR by invoking function adv_no_qui (action, input) where we specify action=drive_cycle to run a driving cycle test [60, Ch. 2.3]. As mentioned in Sec. V-A, we choose the default vehicle file HeavyTruck_in where we use a constant speed (say \( s \) (mph)) profile over a total of 4 hours and a constant grade (say \( \theta \)) over the whole speed profile. Then after running ADVISOR, we can get the total fuel consumption (say \( w \) (gallons)) over a 4-hour driving time with speed \( x \) over a grade-\( \theta \) road. Since almost all the time the truck runs with constant speed \( x \), we can get the corresponding fuel-rate consumption as \( w/4 \) (gph). By enumerating \( x \) from 10 mph to 70 mph with a step of 0.2 mph, and enumerating \( \theta \) from -10.0\% to 10.0\% with a step of 0.1\%, we collect many \((x, \theta, w/4)\) data points.

For each grade \( \theta \) from -10.0\% to 10.0\% with a step of 0.1\%, we use all \((x, w/4)\) points to fit the model (17) by invoking MATLAB’s fit function. We sample several grade points in Tab. V, where we put the strictly convex region for the fitted fuel-rate-speed function \( f_e(x) \). As we can see, all fuel-rate-speed functions \( f_e(\cdot) \) are strictly convex in reasonable speed limit regions. For example, when grade is 0\% (a flat road), \( f_e(\cdot) \) is strictly convex if the speed is larger than 14.22 mph, which holds generally in reality. This justifies our assumption that the fuel-rate-speed function is polynomial and strictly convex over the speed limit region.

More concretely, we visualize the fuel-rate-speed function \( f_e(x) \) and fuel-time function \( c_e(t_e) \) for three sampled grades, -1.0\%, 0.0\%, and 1.0\%, as shown in Fig. 5. We can see that both of them are strictly convex in reasonable regions. We also verify that \( c_e(t_e) \) will first strictly decrease and then strictly increasing and thus we only need to focus on the decreasing interval without loss of optimality, as discussed in Sec. II-B. From Fig. 5(b), we also observe that the fuel-time curve is not smooth but has some glitches. This is due to the gear switch of the truck.

C. Evaluate/Compare FPTAS and Heuristic

We implement our algorithms with C++ where we use the SNAP graph structure [61]. We evaluate on a server with an 8-core Intel Core-i7 3770 3.4 Ghz CPU and 16 GB memory, running CentOS 6.4. To evaluate and compare our FPTAS (Algorithm 3) and heuristic scheme (Algorithm 4), we consider 4 different settings, S1, S2, S3, and S4, as shown in Tab. VI. Note that since we aim to compare them, we use UB = 1 and UB = 1000 in Step 1 of Algorithm 3.

In terms of the minimized fuel cost of the algorithms, Tab. VI shows that the heuristic scheme always outputs the optimal solution (LB = UB, hence LB = UB = OPT), and the FPTAS also outputs a near-optimal solution (e.g., in S1, 74.812 is only a little bit larger than OPT = 74.811). This demonstrates that both FPTAS and the heuristic scheme have good performance. However, in terms of time/space complexity, the heuristic scheme is much better than FPTAS. As we can see, the FPTAS only works fine for the small-scale settings (S1 and S4), where the transportation network in regions 1 and 2 in Fig. 4 is considered, with only 1185 nodes and 2568 edges. When we use a little bit larger scale setting S2, it runs for nearly 1 hour and consumes 14.76 GB memory (out of 16 GB in total). Our server cannot run any other setting whose scale is larger than S2. We also note that the complexity of the FPTAS increases significantly as we decrease \( \epsilon \) from 0.1 to 0.05, as shown in settings S1&S4. Contrarily, our heuristic scheme can handle all 22 regions (setting S3) with 38213 nodes and 82781 edges easily with low time/space complexity.

Tab. VI verifies that the FPTAS is not necessarily scalable to practical large-scale highway networks, but our heuristic scheme works very well in terms of both performance and complexity. To see why the heuristic scheme performs well, we examine an example source-destination pair in the setting S3, \((s, d) = (4, 22)\), and plot its \( \delta(\lambda) \) function (the total travel time of the shortest-generalized-cost path, see (16)) in Fig. 6. We observe that function \( \delta(\lambda) \) is non-increasing, which verifies Theorem 3. Moreover, \( \delta(\lambda) \) has only a few small non-continuous jumps (e.g., a jump at point \( \lambda = 11.47 \) from 37.83 to 37.62). Whenever a (feasible) delay is not within such jump regions, we can always find a \( \lambda_0 \) such that \( \delta(\lambda_0) = T \). According to Theorem 4, the output solution must

\[ \text{TABLE V} \]

**Fitting Parameters, For the Convex Region, \( \leq x \) is the Interval \([0, x]\) and \( \geq x \) is the Interval \([x, \infty)\)**

<table>
<thead>
<tr>
<th>Grade (%)</th>
<th>( a_0 )</th>
<th>( b_0 )</th>
<th>( c_0 )</th>
<th>( d_0 )</th>
<th>Convex Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.0</td>
<td>5.5679e-06</td>
<td>-1.0839e-04</td>
<td>-0.0064</td>
<td>1.0655</td>
<td>( \geq 6.49 )</td>
</tr>
<tr>
<td>-1.0</td>
<td>1.0778e-05</td>
<td>1.2959e-03</td>
<td>-0.0495</td>
<td>1.2879</td>
<td>( \geq 0.00 )</td>
</tr>
<tr>
<td>0.0</td>
<td>3.3057e-05</td>
<td>-1.4102e-03</td>
<td>0.1476</td>
<td>0.5985</td>
<td>( \geq 14.22 )</td>
</tr>
<tr>
<td>1.0</td>
<td>4.5559e-05</td>
<td>-2.3563e-03</td>
<td>0.2583</td>
<td>0.6624</td>
<td>( \geq 15.85 )</td>
</tr>
<tr>
<td>2.0</td>
<td>5.9418e-05</td>
<td>-2.2194e-03</td>
<td>0.3404</td>
<td>0.8741</td>
<td>( \geq 12.45 )</td>
</tr>
</tbody>
</table>

Fig. 5. Fit curve v.s. data for grades 0\%, 1\%, and 2\%, as shown in Tab. V. We can see that all fuel-rate-speed functions \( f_e(\cdot) \) are strictly convex over the speed limit region.

![Image](image_url)
be optimal. For example, when $T = 40$, we can find $\lambda_0 = 4.48$ such that $\delta(\lambda_0) = 40$, as shown in Fig. 6. The optimal solution can be derived as $\{p^*(\lambda_0), (\lambda_0^*)\}$. Even when $T$ is within one of such jump regions (e.g., $T \in (37.62, 37.83)$), since the length of the deadline region (e.g., $37.62, 37.83$ has a length of 0.21 hours) is often negligible as compared to a nearly 40-hour travel, the output LB and UB would be very close. Hence, our heuristic scheme outputs an optimal (at least near-optimal) solution for any input $T$. We will further justify this observation with more instances in Sec. V-D.

D. Compare Performance with Baselines

In this section, we compare the performance of our heuristic scheme with four baseline algorithms. Let us first define the fastest path and the shortest path. The fastest path is the output path of any shortest path algorithm to the graph with edge-cost $f_e^{lb}(= \frac{D_e}{R_e})$ and the shortest path is the output path of any shortest path algorithm to the graph with edge-cost $D_e$. Then the four baseline algorithms are as follows:

(i) Fastest path algorithm with maximum speed: the path is the fastest path and the speed in each edge is the maximum speed.

(ii) Fastest path algorithm with optimal speed: the path is the fastest path and we further do speed optimization over the fastest path.

(iii) Shortest path algorithm with maximum speed: the path is the shortest path and the speed in each edge is the maximum speed.

(iv) Shortest path algorithm with optimal speed: the path is the shortest path and we further do speed optimization over the shortest path.

<table>
<thead>
<tr>
<th>Na</th>
<th>Network</th>
<th>Input</th>
<th>Performance (gallon)</th>
<th>Time (second)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1 &amp; 2</td>
<td>1185</td>
<td>2565</td>
<td>74.811/74.811</td>
<td>74.812</td>
</tr>
<tr>
<td>S2</td>
<td>1 &amp; 2</td>
<td>3274</td>
<td>7465</td>
<td>60.279/60.279</td>
<td>60.279</td>
</tr>
<tr>
<td>S3</td>
<td>1 &amp; 2</td>
<td>38213</td>
<td>82781</td>
<td>290.744/290.744</td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>1 &amp; 2</td>
<td>1185</td>
<td>2565</td>
<td>74.811/74.811</td>
<td>74.812</td>
</tr>
</tbody>
</table>

Each of them outputs one solution for PASO. Since our heuristic scheme outputs two solutions respectively corresponding to the LB and UB, we have 6 solutions in total, as summarized in Tab. VII.

In later comparison, since the travel time of $F$ is the minimum time for any feasible solution of PASO, we will use it as the time benchmark. For example, a solution SOL (e.g., SOL could be OPT-UB) with time increment 10% means that Travel time of SOL $- $Travel time of $F = 10\%$. Similarly, we use the travel distance of S/S-So as the distance benchmark, and use the fuel consumption of OPT-LB as the fuel benchmark.

In our simulation, we evaluate in total 2704 different $(s, d, T)$ tuples. The source node $s$ and the destination node $d$ could range from 1 to 22 (see the 22 regions in Fig. 4). For any $(s, d)$ pair, the deadlines $T$ could range from $[T^f]$ to $[T^f] + 9$ where $T^f$ is the smallest travel time from $s$ to $d$, i.e., the travel time computed by baseline algorithm (i).

1) A Single Instance: We first consider one instance $(s, d, T) = (9, 22, 40)$. Tab. VIII compares the 6 solutions. As we can see, our heuristic scheme again outputs the optimal solution. It consumes 301.0 gallons of fuel, runs 10.76\% slower than the time benchmark (F), and 0.3\% longer than the distance benchmark (S/S-So). Also, without speed optimization, the fastest path (F) consumes 32 more gallons (10.67\%) and the shortest path (S) consumers 18 more gallons (5.99\%).
But with speed optimization, both fastest path and shortest path have near-optimal performance.

For \((s, d) = (9, 22)\), we also evaluate the effect of input deadline \(T\) as shown in Fig. 7. Considering speed optimization, when the input deadline \(T \in [36.11, 38.58]\), the shortest path is infeasible, which shows that fastest path outperforms shortest path. The shortest path becomes feasible when \(T \geq 38.58\), and it outperforms the fastest path when \(T > 39\). This figure thus shows that the shortest path becomes better and better as the deadline constraint increases. Intuitively, when the hard deadline constraint can be satisfied, the travel distance would be critical for the total fuel consumption.

The OPT-LB curve in Fig. 7 is the energy-deadline tradeoff of \((s, d) = (9, 22)\). We see that increasing deadline can save fuel consumption, and the saving has a "diminishing return" property. For example, the truck can save 6.6 gallons of fuel if it increases its deadline from 37 to 38 hours, but the saving reduces to 1.46 gallons if its deadline is relaxed from 45 to 46 hours. A more comprehensive study on energy-deadline tradeoff is shown in Sec. V-E.

2) All Instances: Similar to Tab. VIII, we can get the time, distance, and fuel of the 6 solutions for all source-sink pairs. We evaluate the average performance of all running instances in terms of time/distance/fuel increments compared to the benchmark numbers, as summarized in Tab. IX. Note that in 4.84% of instances, shortest path is infeasible. Tab. IX only has the average performance over the instances where the shortest path is feasible.

Tab. IX shows that on average OPT-UB only consumes 0.02% of more fuels than the fuel benchmark (OPT-LB). This again shows that our heuristic scheme outputs a near-optimal solution in all instances.

For the baseline algorithms, Tab. IX shows that the fastest path (resp. shortest path) algorithm without speed optimization consumes 20.14% (resp. 16.4%) of more fuels than our solution. In other words, our heuristic solution achieves 16.76% (resp. 14.09%) fuel consumption reduction, as compared to the fastest path (resp. shortest path) algorithm. Our heuristic solution also improves the 36-ton-truck’s fuel economy from 5.05 for the fastest path and 5.13 for the shortest path to 5.96. Considering its significant portion of energy consumption, our solution can indeed save much fuel cost for the long-haul heavy-duty trucks.

When we allow speed optimization for the fastest path and the shortest path, we find that on average both of them are close to the optimal solution. More specifically, F-SO consumes 2.00% of more fuels and S-SO only consumes 0.31% of more fuels than OPT-LB. This apparently suggests that in the U.S., it is good enough to first choose the shortest or fastest path and then do speed optimization. However, in our simulation, the shortest path is infeasible among 4.84% of all instances, and the fastest path with speed optimization can consume 21.32% of more fuels in the worst instance. As opposed to them, our PASO solution is robust in the sense that it always output a solution that is both feasible and near-optimal. We also leave it as a future work to understand under which conditions the fastest/shortest path with speed optimization is close to the optimal solution.

E. Energy-Deadline Tradeoff

In this subsection, we evaluate all \((s, d)\) pairs where \(s\) and \(d\) range from 1 to 22. For each \((s, d)\) pair, we first get the smallest travel time \(T^f\), i.e., the travel time computed by baseline algorithm (i) in Sec. V-D, and get the corresponding fuel consumption \(C^f\). Now we increase the deadline by \(x\%\) and evaluate the fuel consumption \(C(x\%)\) when \(T = (1 + x\%)T^f\), and get the fuel consumption reduction \(\frac{C^f - C(x\%)}{C^f}\). By changing the percentage of delay increase, i.e., \(x\), we get different percentages of fuel consumption reduction. The energy-deadline tradeoff performance among all \((s, d)\) pairs is shown in Fig. 8.
As we can see, the fuel consumption reduction has a “diminishing return” property. As compared to the fastest travel time, if we increase the hard deadline by 10%, we can reduce the fuel consumption by about 10% on average. If we increase the hard deadline by 50%, we can reduce the fuel consumption by about 20% on average. If we further increase the hard deadline after 70%, there is little extra benefit. This is because the optimal running speed over most edges becomes the minimum speed and there is little room to do further speed optimization if we increase the deadline more than 70%.

VI. CONCLUSION AND FUTURE WORK

Provisioning both energy-efficient and timely delivery is of great importance for logistic operators. This paper presents a first step to study the energy-efficient timely transportation problem with an emphasis for long-haul heavy-duty trucks. We propose two algorithms: the first one is an FPTAS and the second one is a heuristic with lower complexity and near-optimal empirical performance. Our real-world data-driven simulations show that our solution guarantees timely delivery and can save up to 17% of fuel consumption as compared to a fastest/shortest path algorithm adapted from common practice. An interesting and important future direction is to generalize our results beyond the highway setting to cover more sophisticated local driving scenarios.

REFERENCES


