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

Lei Deng  
Dept. of IE, CUHK

Minghua Chen  
Dept. of IE, CUHK

Mohammad H. Hajiesmaili  
Dept. of IE, CUHK

Haibo Zeng  
Dept. of ECE, Virginia Tech

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=NP. We then design a fully polynomial time approximation scheme (FPTAS) that attains an approximation ratio of \(1 + \epsilon\) with a time complexity of \(O(mn^2/\epsilon^2)\), where \(m\) and \(n\) are the numbers of nodes and edges, respectively. 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 fast heuristic solution with \(O(m + n \log n)\) 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 the 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 solution achieves 17% (resp. 14%) fuel consumption reduction, as compared to a fastest path (resp. shortest path) algorithm adapted from common practice.

CCS Concepts

- Applied computing → Transportation; Mathematics of computing → Mixed discrete-continuous optimization;

1. INTRODUCTION

In the U.S., heavy-duty trucks haul more than 70% of all freight tonnage [11], and they consume 17.6% of energy in transportation sector [21, Tab. 2.8] and contribute to about 5% of the greenhouse gas emission [8]. Fuel cost is the largest operating cost (34%) of truck owners/operators [25], 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 [13, 27, 38], to better management of truck parts such as maintaining optimal tire pressures [4]. The second line is to operate heavy-duty trucks more economically. This explores several possibilities, e.g., reducing idling energy consumption [40], platooning more than one heavy-duty trucks [15, 32], route planning [23, 41, 43], and speed planning [3, 10, 29, 30]. 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 [43] 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: As a rule of thumb for truck operations on highway, every one mile per hour (mph) increase in speed incurs about 0.14 mile per gallon (mpg) penalty in fuel economy [3, 10].

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 [12, 35]. As estimated by the U.S. Federal Highway Administration (FHWA) in [35], 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 [18], 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 [6]. Finally, violating scheduled delay can introduce difficulties for global logistic decisions and even increase the uncertainty and inefficiency of supply chains [35]. Overall, it is crucial to ensure timely goods delivery.
livery for truck operators, and considering timely delivery in fuel cost minimization poses a unique challenge of which only partial results for special cases are recently available [29,30].

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 truck’s fuel consumption while satisfying a hard deadline constraint, under which we take both route planning and speed planning into account to exploit complete design space of reducing fuel consumption. Since heavy-duty trucks are mainly operated for long-haul delivery and most of time run on highways [21, Tab. 5.2 and Fig. 5.1], we focus our model on their operation in the highway transportation network system. We summarize our contributions in the following.

- We formulate an energy-efficient timely transportation problem of minimizing the fuel consumption subject to a hard deadline constraint for a heavy-duty truck running on a highway transportation network, with design spaces of both route planning and speed planning in Sec. 2. We show that our problem is NP-Complete.
- In Sec. 3, we design a fully polynomial time approximation scheme (FPTAS) for solving the energy-efficient timely transportation problem. The proposed FPTAS attains an approximation ratio of $1 + \epsilon$ with a network-size induced complexity of $O(mn^2/\epsilon^2)$, where $m$ and $n$ are the numbers of nodes and edges, respectively.
- 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. In Sec. 4, by leveraging elegant insights from studying the dual of the original problem, we design a fast heuristic solution with $O(m + n \log n)$ complexity. The proposed heuristic scheme 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 the practical instances in numerical experiments in Sec. 5, justifying the superior empirical performance of our heuristic.

- We carry out extensive numerical experiments using real-world truck data over the U.S. highway network in Sec. 5. 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 [2].

- For those who are familiar with Restricted Shortest Path (RSP) problem [26,28,31], our energy-efficient timely transportation problem is a generalized version of RSP, including an extra design space of speed planning. Therefore, from the theoretical perspective, we generalize the FPTAS design and the dual-based design of RSP to our problem.

2. MODEL AND PROBLEM FORMULATION

2.1 System Model

Consider a highway transportation network as exemplified in Fig. 1. We model it as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex/node set and $\mathcal{E}$ is the edge/road set. We define $n \triangleq |\mathcal{V}|$ as the number of nodes and $m \triangleq |\mathcal{E}|$ as the number of edges. For each edge $e \in \mathcal{E}$, we denote $D_e > 0$ as its distance (unit: mile), and $R_{lb}^e > 0$ (resp. $R_{ub}^e > 0$) as its minimal (resp. maximal) speed (unit: mph). (Governments usually set the maximal speed for all highways and the minimal 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 who aims to ship cargos from a source node $s \in \mathcal{V}$ to a destination node $d \in \mathcal{V}$. The goal is to minimize the energy/fuel\(^1\) consumption subject to a hard delay 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 [3,10]. 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 [22]). In this paper, we use the instantaneous fuel consumption model [14,22] 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. The instantaneous acceleration/deceleration reflects the speed variation. However, since we consider a highway model, the truck spends most of time to maintain a relatively constant cruise speed [17,36] such that the fuel consumption caused by acceleration/deceleration would be negligible. This motivates us to model the instantaneous fuel consumption as a function of the instantaneous speed.

We thus define $f_e : [R_{lb}^e, R_{ub}^e] \to \mathbb{R}^+$ 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 $\tau$ (unit: hour) with the constant speed $r_e$ is $f_e(r_e) \cdot \tau$ (unit: gallon).

Since many existing models [14,16,17,19,39] 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 $[R_{lb}^e, R_{ub}^e]$. This assumption also holds in the physical interpretation of fuel-rate-speed function as shown in our technical report [24], and is further verified in our simulation using real-world data (see Fig. 5(a)).

\(^1\)We interchangeably use fuel and energy in this paper.

\(^2\)The strict convexity can be relaxed to convexity. For simplicity, we use the strict convexity in this paper.
2.2 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 \( e \in \mathcal{E} \),

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

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.

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. 2.1, 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 \( e \). We then define a fuel-time function \( c_e(\cdot) \) for each road \( e \),

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

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

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

\[
\text{PASO: } \min_{\mathbf{x} \in \mathcal{X}, \mathbf{t} \in \mathcal{T}} \sum_{e \in \mathcal{E}} x_e \cdot c_e(t_e) \tag{3}
\]

s.t. \[
\sum_{e \in \mathcal{E}} x_e t_e \leq T, \tag{4}
\]

In PASO, set \( \mathcal{X} \) restricts that one and only one \( s-d \) path is selected, defined as

\[
\mathcal{X} \triangleq \{ \mathbf{x} : x_e \in \{0, 1\}, \forall e \in \mathcal{E}, \text{ and } \sum_{e \in \text{out}(v)} x_e - \sum_{e \in \text{in}(v)} x_e = 1\{v=s\} - 1\{v=d\}, \forall v \in \mathcal{V}\},
\]

where \( 1\{\cdot\} \) is the indicator function, \( \text{in}(v) \triangleq \{(u, v) : (u, v) \in \mathcal{E}\} \) is the set of incoming edges of node \( v \in \mathcal{V} \), \( \text{out}(v) \triangleq \{(v, u) : (v, u) \in \mathcal{E}\} \) is the set of outgoing edges of node \( v \). Set \( \mathcal{T} \) captures the speed limits of all roads, defined as

\[
\mathcal{T} \triangleq \{ \mathbf{t} : t_e^l \leq t_e \leq t_e^u, \forall e \in \mathcal{E}\},
\]

where \( t_e^l \triangleq D_e / \hat{t}_e^u \) and \( t_e^u \triangleq D_e / \hat{t}_e^l \) are the minimal and maximal travel time due to the speed limits on edge \( e \), respectively.

Constraint (4) is to satisfy the hard delay requirement. Objective (3) is to minimize the total fuel consumption selected the path (3).

2.3 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 [26, 28]. 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 minimal-cost path subject to a hard path delay 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 [26], 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 \( R_e = R_e^u \) to an appropriate value for each edge \( e \) in PASO, and using the result that RSP is NP-Complete [26].

2.4 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 \( t_e^u \) to find the fastest path. If the travel time of the fastest path is larger than the delay requirement \( T \), PASO is infeasible. In the rest of this paper, we thus assume that the delay 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 \([t_e^l, t_e^u]\). Also, there exists a point \( t_e \in [t_e^l, t_e^u] \) such that \( c_e(t_e) \) is first strictly decreasing over \([t_e^l, t_e]\) and then strictly increasing over \([t_e, t_e^u]\).

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 \([t_e^l, t_e]\). Otherwise, we can decrease the travel time from \( t_e \) to \( t_e^l \) 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 travel time limit from \([t_e^l, t_e^u]\) to \([t_e^l, \hat{t}_e]\), which equivalently implies that we reset the speed limit from \([R_e^l, R_e^u]\) to \([R_e^l, \hat{R}_e]\). After such preprocessing, in the rest of the paper, \( c_e(t_e) \) can be assumed to be strictly convex and strictly decreasing over \([t_e^l, \hat{t}_e]\) 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 \( c(p, t_p) \triangleq \sum_{e \in p} c_e(t_e) \) as the fuel consumption of path \( p \) with travel time set \( t_p \), and OPT as the optimal value of PASO.

Next, we will propose a fully polynomial time approximation scheme (FPTAS) in Sec. 3 and a fast dual-based heuristic scheme in Sec. 4 to solve our problem PASO.

3. AN FPTAS FOR PASO

Since PASO generalizes RSP, which is well-known to have an FPTAS [28, 34], it is natural to ask whether we can extend RSP’s FPTAS for our problem PASO. In this section, by carefully tackling the difference between PASO and RSP, we “reformulate” PASO such that we can adapt RSP’s FPTAS to construct an FPTAS for 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 + ϵ)-approximate solution in the sense that the solution is feasible and the corresponding fuel consumption is at most (1 + ϵ)OPT, and the time complexity is polynomial in both the problem size and 1/ϵ.

The essence of RSP’s FPTAS [28, 34] is a test procedure. For any input value V > 0 and any input accuracy parameter δ > 0, the test procedure can “approximately” compare V and the optimal value OPT in the sense that it can tell either OPT > V or OPT ≤ (1 + δ)V in polynomial time. Based on this test procedure, starting with some arbitrary lower bound LB and upper bound UB for OPT, a binary search scheme is designed [28, 34] to exponentially narrow down the bounding interval [LB, UB] and finally a (1 + ϵ)-approximate solution is outputted.

To solve our problem PASO, we adapt RSP’s FPTAS by designing our own test procedure. In RSP, [28] and [34] 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 δ. 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 RSP, in PASO each edge has a fuel-time function. Hence, instead of rounding a fixed cost in RSP, we quantize the continuous fuel-time function c_v(·) into another staircase fuel-time function ˜c_v(·) according to the input value V and the input accuracy parameter δ, 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. 3.1. Then we present our own test procedure (Algorithm 2) which invokes Algorithm 1 in Sec. 3.2. Finally, we describe the whole FPTAS (Algorithm 3) which invokes Algorithm 2 in Sec. 3.3.

### 3.1 Quantizing Fuel-Time Function

For any input value V > 0 and N ∈ Z^+, we quantize the edge-e fuel-time function c_v(t_e) to be

\[ ˜c_v(t_e) := \min \left\{ \frac{c_v(t_e)}{V} + 1, N \right\}, \quad \forall t_e \in [t^L_e, t^U_e]. \]  

(5)

Since we have assumed that c_v(t_e) is strictly decreasing in Sec. 2.4, ˜c_v(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 maximal number of stairs. Since c_v(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_v(t_e) by putting a ceiling V N. This truncation is sufficient for use in the test procedure (see Sec. 3.2). Clearly, ˜c_v(t_e) is a quantized and truncated version of c_v(t_e). An example is shown in Fig. 2. Here we set V = 20, N = 4. Thus, each stair spans 20 and c_v(t_e) is truncated by the ceiling V N = 80. The resulting curve ˜c_v(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 ˜c_v(t_e) is a staircase function and only takes integer values, we can use an N-dim vector τ_c to represent it without any information loss. We define it as τ_c := (τ_c^1, τ_c^2, · · · , τ_c^N) where τ_c^i is the minimal travel time over [t^L_e, t^U_e] such that ˜c_v(τ_c^i) = i and is defined as nan if ˜c_v(τ_c^i) = 0 has no solution. For the example in Fig. 2, we have τ_c = (τ_c^1, τ_c^2, τ_c^3, τ_c^4) = (nan, 2.5, 1.8, 1).

We call (τ_c^i, i) the i-th representative point of ˜c_v(·). Thus ˜c_v(·) is characterized by at most N representative points, which will play a key role in our test procedure in Sec. 3.2.

#### Time Complexity

The major complexity of Algorithm 1 comes from line 8, which needs to solve an equation. Since we have assumed that c_v(t_e) is a strictly decreasing function, we can use a binary search to solve this equation, which has time complexity O(log (t^U_e - t^L_e))^3. Hence, the total complexity of QUANTIZE(e, V, N) in Algorithm 1 is O(N log (t^U_e - t^L_e)).

If we define

\[ ξ := \max_{e \in E} \left( \frac{t^U_e - t^L_e}{V} \right) \]  

(6)

as the maximal range of travel time over all edges, for any t_e ∈ E, the complexity of QUANTIZE(e, V, N) is O(N log ξ).

### 3.2 The Test Procedure

Figure 2: An example for quantizing c_v(·).

Figure 3: Binary search quantizing c_v(·).

**Algorithm 1 A Quantizing Procedure QUANTIZE(e, V, N)**

1: for i = 1, 2, · · · , N do
2: Set τ_c^i = nan
3: end for
4: Set n_{min} = ˜c_v(t^L_e^0) = \min\{\left\lfloor \frac{c_v(t^L_e^0)}{V} \right\rfloor + 1, N\} \quad \text{Step 1}
5: Set n_{max} = ˜c_v(t^U_e^0) = \min\{\left\lfloor \frac{c_v(t^U_e^0)}{V} \right\rfloor + 1, N\} \quad \text{Step 1}
6: Set τ_c^{n_{max}} = t^L_e^0 \quad \text{Step 2}
7: for i = n_{min}, n_{min} + 1, · · · , n_{max} - 1 do
8: Solve the equation c_v(t_e) = iV over t_e ∈ [t^L_e, t^U_e] \quad \text{Step 2} \text{.1}
9: if the equation has a solution t_e then
10: Set τ_c^i = t_e \quad \text{Step 2} \text{.2}
11: else if
12: end for
13: return τ_c = (τ_c^1, τ_c^2, · · · , τ_c^N) \quad \text{Step 2} \text{.3}

\[ \text{Before TEST(V,V,1) \quad Returns A Path} \]

\[ \text{Before TEST(V,V,1) \quad Returns FAIL} \]

\[ \text{Before TEST(V,V,1) Returns FAIL} \]

\[ \text{Before TEST(V,V,1) Returns A Path} \]

\[ \text{Before TEST(V,V,1) \quad Returns FAIL} \]
As introduced above, the test procedure should “approximately” compare \( V \) and the optimal value \( \text{OPT} \) such that it can answer either \( \text{OPT} > V \) or \( \text{OPT} \leq (1+\epsilon)V \) in polynomial time. Inspired by [34], which improves the FPTAS of RSP in [28], we adopt a more powerful test procedure, denoted by \( \text{TEST}(L, U, \delta) \). It can answer either \( \text{OPT} > U \) or \( \text{OPT} \leq U + \delta L \). Clearly, if we set \( L = U = V \), \( \text{TEST}(V, V, \delta) \) can answer either \( \text{OPT} > V \) or \( \text{OPT} \leq (1+\epsilon)V \), which exactly completes the “approximate” comparison. The reason to adopt a more powerful test procedure, similar to [34], is that we will also use it to finally output a \((1+\epsilon)\)-approximate solution. We will discuss it soon in Sec. 3.3.

The details of \( \text{TEST}(L, U, \delta) \) are shown in Algorithm 2. As we mentioned before, the major difference between our problem PASO and the existing problem RSP is that PASO has a continuous fuel-time function for each edge instead of a polynomial time complexity eventually, we put a ceil of \( N \) and \( \tau_p \). After such quantization, the fuel-time function is \( e \in \mathbb{Z} \). This is exactly an RSP problem. Therefore, the remaining steps follow the test procedure for RSP on the new graph \( \mathcal{G} \). Specifically, since each edge \( e \in \mathcal{E} \) in the original graph corresponds to at most \( N+1 \) edges in the new graph \( \mathcal{E} \). For each edge \( e \in \mathcal{E} \), the edge cost \( \bar{c}_e \) is a positive integer, as shown in (5). This is our own test procedure (Algorithm 2), we then follow the FPTAS for RSP in [34, Fig. 2] by replacing its test procedure with ours. For completeness, we present the FPTAS in Algorithm 2 and explain it with the following three steps.

**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 \( \text{OPT} \). Define that the minimal single-edge fuel cost is \( C_{\text{b}} \triangleq \min_{e \in \mathcal{E}} c_e(t_{e}^{\text{m}}) \) and the maximal single-edge fuel cost is \( C_{\text{b}} \triangleq \max_{e \in \mathcal{E}} c_e(t_{e}^{\text{m}}) \). Simply, we can use the minimal single-edge fuel consumption \( C_{\text{b}} \triangleq \) as the lower bound \( LB \) and use the maximal single-path fuel consumption \( nC_{\text{b}} \) as the

**Lemma 3.** If Algorithm 2 returns a path \( p \) and travel time set \( t_p \), then we have

\[
\text{OPT} \leq c(p, t_p) \leq U + L\delta. \tag{8}
\]

**Algorithm 2 A Test Procedure \( \text{TEST}(L, U, \delta) \)**

```
1: Set \( V = \frac{L}{U+1} \)
2: Set \( N = \left\lfloor \frac{V}{\delta} \right\rfloor + n + 1 \)
3: for \( e \in \mathcal{E} \) do
   4: \( \text{Get } \tau_e = \text{QUANTIZE}(e, V, N + 1) \)
5: \end for
6: Set \( g_e(c) = 0, \forall c = 0, 1, \ldots, N \)
7: Set \( g_e(0) = \infty, \forall c \neq s, v \in V \)
8: for \( e = 1, 2, \ldots, N \) do
   9: for \( v \in V \) do
      10: Set \( g_v(c) \) according to (7)
   11: \end for
12: if \( g_e(c) \leq T \) then
   13: return the corresponding path \( p \) and travel time set \( t_p = \{t_e : e \in p\} \)
14: \end if
15: \end for
16: return FAIL
```

**Lemma 4.** If \( U \geq \text{OPT} \), then Algorithm 2 must return a feasible path \( p \) and travel time set \( t_p \), which satisfy

\[
c(p, t_p) \leq \text{OPT} + L\delta. \tag{9}
\]

**Lemma 5.** If Algorithm 2 returns FAIL, then we have

\[
\text{OPT} > U \tag{10}
\]

**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 \( \text{OPT} \leq U + L\delta \) from Lemma 3, or returns FAIL in line 16, which implies \( \text{OPT} > U \) from Lemma 5. Therefore, Lemma 3 and Lemma 5 justify that our test procedure (Algorithm 2) completes the “approximate” comparison, i.e., answers either \( \text{OPT} > U \) or \( \text{OPT} \leq U + L\delta \).

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. 3.3 to finally output a \((1+\epsilon)\)-approximate solution.

**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 \( O(m N^2) \). Since \( N = \left\lfloor \frac{V}{\delta} \right\rfloor + n + 1 = \left\lfloor \frac{V}{\delta} \right\rfloor + \frac{1}{2} + \frac{1}{\delta} + n + 1 = O\left(\frac{V}{\delta} \frac{1}{\delta} + \frac{n}{\delta} + n\right) \), the total time complexity of Algorithm 2 is \( O(m N \log \xi + m N^2) = O(mN \log \xi + m \log \xi + m^2 + n^2) \).

### 3.3 The Proposed FPTAS

Based on our own test procedure (Algorithm 2), we then follow the FPTAS for RSP in [34, 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.

**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 \( \text{OPT} \). Define that the minimal single-edge fuel cost is \( C_{\text{b}} \triangleq \min_{e \in \mathcal{E}} c_e(t_{e}^{\text{m}}) \) and the maximal single-edge fuel cost is \( C_{\text{b}} \triangleq \max_{e \in \mathcal{E}} c_e(t_{e}^{\text{m}}) \). Simply, we can use the minimal single-edge fuel consumption \( C_{\text{b}} \triangleq \) as the lower bound \( LB \) and use the maximal single-path fuel consumption \( nC_{\text{b}} \) as the
upper bound UB. Also, in Sec. 4, 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 [LB, UB] until \( \frac{LB}{UB} \leq \epsilon \). The binary search step is visualized in Fig. 3. Note that we always keep LB as a lower bound and UB as an upper bound for OPT. Whenever \( \frac{UB}{LB} > 16 \), we input the geometric mean \( V = \sqrt{LB \cdot UB} \) and \( \delta = 1 \) to the test procedure, as shown in lines 5 and 6. If TEST(\(V, V, 1\)) returns FAIL, then according to Lemma 4, we must have \( V < OPT \). In this case, we reset the lower bound \( LB \) to be \( V \) in line 8. Otherwise, TEST(\(V, V, 1\)) returns a feasible path \( p \) and travel time \( t_p \). According to Lemma 3, we must have OPT \( \geq V + 5V = 2V \). We reset the upper bound to be \( 2V \) in line 10. It can be easily shown that this binary search returns a lower bound \( LB \) and an upper bound \( UB \) for OPT such that \( \frac{UB}{LB} \leq 16 \) in \( O(\log \log \frac{UB}{LB}) \) iterations.

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

\[
c(p, t_p) \leq OPT + \epsilon LB \leq OPT + \epsilon OPT = (1 + \epsilon)OPT.
\]

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

**Time Complexity:** Step 1 requires \( O(m) \) to get an initial lower bound LB and upper bound UB. Step 2 invokes the test procedure \( O(\log \log \frac{UB}{LB}) \) times and each invocation takes \( O(mn \log \xi + mn^2) \) time by using \( L = U = V \) and \( \delta = 1 \). Thus Step 2 takes \( O(mn \log \xi + mn^2) \log \log \frac{UB}{LB} \). 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 \( O(\frac{UB}{LB}) = O(1) \) because \( \frac{UB}{LB} \leq 16 = O(1) \). Here we can also see why we need to use a binary search to obtain \( \frac{UB}{LB} \leq 16 \) in Step 2. This is because \( \frac{UB}{LB} = O(1) \) ensures polynomial time complexity in Step 3. Therefore, the total complexity is \( O((mn \log \xi + mn^2) \log \log \frac{UB}{LB} + \frac{mn \log \xi + mn^2}{\epsilon}) \).

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 \( O((mn \log \xi + mn^2) \log \log \frac{UB}{LB} + \frac{mn \log \xi + mn^2}{\epsilon}) \). In addition, when we use \( LB = C^b \) and \( UB = nC^b \) where \( C^b \triangleq \min_{e \in E} c_e(t_e^b) \) and \( C^b \triangleq \max_{e \in E} c_e(t_e^b) \), we have \( \log \log \frac{UB}{LB} = \max\{O(\log \log n), O(I_e)\} \) where \( I_e \) is the input size of all parameters of edge \( e \). Thus, Algorithm 3 has time complexity polynomial in the size of the problem PASO and therefore is an FPTAS.

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.

## 4. 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. 4.3, 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. 5, this heuristic scheme outputs an optimal (or at least near optimal) solution, i.e., \( LB = UB = OPT \) (or at least \( LB \approx OPT \approx UB \)).

### 4.1 Lagrangian Relaxation and Dual Problem

In our problem PASO, since the hard delay 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 such relaxation, we can get the corresponding Lagrangian,

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

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

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

### 4.2 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,

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

\[
\begin{aligned}
&= -\lambda T + \min_{x \in \mathcal{X}, t \in \mathcal{T}} \sum_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e) \\
&\text{(Eq1)} \quad \leq -\lambda T + \min_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} \sum_{e \in E} x_e \cdot (c_e(t_e) + \lambda t_e)
\end{aligned}
\]

\[
\begin{aligned}
&\leq -\lambda T + \min_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} \sum_{e \in E} x_e \cdot \min_{\nu \leq t_e \leq \nu + e} (c_e(\nu) + \lambda \nu)
\end{aligned}
\]

\[
\begin{aligned}
&\leq -\lambda T + \min_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} x_e \cdot \min_{\nu \leq t_e \leq \nu + e} (c_e(\nu) + \lambda \nu) + \lambda t_e
\end{aligned}
\]

\[
\begin{aligned}
&\leq -\lambda T + \min_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} x_e \cdot \min_{\nu \leq t_e \leq \nu + e} (c_e(t_e^\nu) + \lambda \nu)
\end{aligned}
\]

\[
\begin{aligned}
&\leq -\lambda T + \min_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} x_e \cdot \nu_e(\lambda)
\end{aligned}
\]

\[
\begin{aligned}
&= -\lambda T + \sum_{e \in E} \nu_e(\lambda)
\end{aligned}
\]

(12)
We explain \((E_5) - (E_9)\) in (12) one by one. Equality \((E_5)\) is because no coupled constraints exist for \(x\) and \(t\). Equality \((E_9)\) is because no coupled constraints exist for the travel time at different edges in \(T\).

In equality \((E_5)\), \(t^*_e(\lambda)\) is defined as

\[
t^*_e(\lambda) \triangleq \arg \min_{0 \leq t_e \leq t^{eb}_e} (c_e(t_e) + \lambda t_e). \tag{13}
\]

Note that since we have assumed that \(c_e(t_e)\) is strictly convex and strictly decreasing over \([t^{eb}_e, t^{ub}_e]\) in Sec. 2.4, \(t^*_e(\lambda)\) is unique and thus (13) is well defined. Specifically, \(t^*_e(\lambda)\) can be obtained analytically as follows.

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

\[
t^*_e(\lambda) = \begin{cases} 
t^{eb}_e, & \text{if } 0 \leq \lambda < -c'_e(t^{eb}_e); \\ c^{-1}_e(-\lambda), & \text{if } -c'_e(t^{eb}_e) \leq \lambda \leq -c'_e(t^{ub}_e); \\ t^{ub}_e, & \text{if } \lambda > -c'_e(t^{ub}_e). \end{cases} \tag{14}
\]

In addition, (13) has a nice economic interpretation. As we have relaxed the hard delay 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, \(t^*_e(\lambda)\) is the best response of edge \(e\) for a given delay price \(\lambda\).

In equality \((E_9)\), \(w_e(\lambda)\) is defined as

\[
w_e(\lambda) \triangleq c_e(t^*_e(\lambda)) + \lambda t^*_e(\lambda), \tag{15}
\]

which can be interpreted as the minimal 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 \((E_9)\), since \(X\) restricts that an \(s - d\) path is selected, \(\min_{e \in X} \sum_{e \in e} x_e \cdot w_e(\lambda)\) 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 \(\text{PASO}\).

### 4.3 The Heuristic Algorithm

Our heuristic scheme relies on one key observation. Define

\[
\delta(\lambda) \triangleq \sum_{e \in p^*(\lambda)} t^*_e(\lambda), \tag{16}
\]

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 \(\lambda \in [0, +\infty)\).

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 delay 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 [20, Ch. 5.1.6]. Therefore, we aim to find a \(\lambda_0\) such that \(\delta(\lambda_0) = T\). However, our problem \(\text{PASO}\) 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) as 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. 5, we set \(\lambda_{\text{max}} = 100\), which works for all settings. Then we do binary search in lines 3-19, where \(\text{tol}\) 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:

- **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 \(\text{PASO}\) in Theorem 4.
- **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 \(\text{tol}\), \(\lambda_L = \lambda_0 - \text{tol}/2 \rightarrow \lambda_0\). Likewise, \(\lambda_U = \lambda_0 + \text{tol}/2 \rightarrow \lambda_0\). 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 \(\text{OPT}\), 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 \(\text{PASO}\).
As a by-product, Theorem 4 also shows that the strong duality for the combinatorial problem PASO holds in this case, and λ = 0 is the optimal dual solution to PASO-Dual.

Theorem 5. If Algorithm 4 returns in line 20, and define LB = \( \sum_{e \in E^*} (\xi_e (\lambda_e)) \) and UB = \( \sum_{e \in E^*} (\lambda_e) \), then we have LB \( \leq \) OPT \( \leq UB \).

The LB and UB returned by Algorithm 4 in line 20 can be used for Step 1 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.

Time Complexity: If we use Dijkstra’s shortest-path algorithm with a min-priority queue in line 7 in Algorithm 4, Algorithm 4 has complexity \( O((m + n \log n) \log \lambda_{max}) \), much faster than the FPTAS (Algorithm 3).

Remark: A similar dual-based heuristical approach for RSP is proposed in [31]. However, as mentioned in Sec. 3, 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 heuristical design from RSP [31] to PASO.

5. 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, and (iii) compare our algorithms with baseline algorithms, including both shortest path algorithm and fastest path algorithm adapted from common practice.

5.1 Dataset

Transportation Network: We construct the U.S. National Highway Systems (NHS) from the dataset of Clinched Highway Mapping (CHM) Project [42]. The whole highway network graph file is specified in [1], which consists of 84504 nodes (waypoints) and 89119 (one-direction) edges.

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 [9] provided by the U.S. Geological Survey (USGS) to query elevations of all nodes in the NHS graph.

Speed Limits: We use the historical average speed as the maximal speed limit \( R_{max}^{(e)} \) for each road \( e \). HERE map [7] 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 [7] for two weeks and use the average as \( R_{max}^{(e)} \) for each road \( e \) in the NHS graph. For the minimal speed limit \( R_{min}^{(e)} \), we manually set it to be \( R_{min}^{(e)} = \min(30, R_{max}^{(e)}) \).

Fuel Consumption Data: It is hard for us to get available real-world fuel consumption data. In this paper, we instead leverage the widely-used ADVISOR simulator [37] to collect fuel consumption data (see Sec. 5.2).

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 [5], 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. 1.

Preprocessing Highway Network: In the original NHS graph from CHM [1], we observe that: (i) most roads are in the “eastern” U.S., and (ii) many roads are very short (non-intersection roads). To create a network with more diverse paths, we first cut the whole NHS graph to the “eastern” part with longitude to the east of 100°W (see Fig. 4). We further merge the non-intersection roads with the same level of grades into a single road. Some network statistics after these two kinds of preprocessing are shown in Tab. 2. Note that since the average distance for each edge is 3.26 miles after preprocessing, it is reasonable to ignore the speed transition over two adjacent edges, which justifies the assumption in our fuel consumption model.

Moreover, to better visualize our results, we divide the major “eastern” U.S. into 22 regions (see Fig. 4). In each region \( i \in [1, 22] \), we find the node in the graph which is nearest to the region’s center. We also call it node \( i \). Next we will use such 22 nodes as the source and destination nodes.

5.2 Model Fuel-Rate-Speed Function

We model the fuel-rate-speed function as
\[
f_f(x) = a_0 x^3 + b_0 x^2 + c_0 x + d_0, \quad \forall e \in E
\]  
(17)
Here \( x \) is the speed (unit: mph) and \( f_f(x) \) 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 [14]. We collect fuel consumption data from ADVISOR and fit fuel-rate-speed functions \( f_f(x) \) by MATLAB’s fit tool. Due to the space limitation, the details are shown in our technical report [24].

Our results show that the fuel-rate-speed function \( f_f(x) \) is strictly convex in reasonable speed limit regions. More con-
cretely, we visualize the fuel-rate-speed function \( f_\epsilon(x) \) and fuel-time function \( c_\epsilon(t_e) \) 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 \( c_\epsilon(t_e) \) 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. 2.2.

![Fuel-rate-speed function](a) and Fuel-time function \( c_\epsilon(t_e) \) over a 100-mile road.

**Figure 5:** Fit curve v.s. data for grades 0%, ±1%.

### 5.3 Evaluate/Compare FPTAS and Heuristic

We implement our algorithms with C++ where we use the SNAP graph structure [33]. 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, \( S_1, S_2, S_3, \) and \( S_4 \), as shown in Tab. 3. Note that since we aim to compare them, we use \( LB = 1 \) and \( UB = 1000 \) in Step 1 of Algorithm 3.

In terms of the minimized fuel cost of the algorithms, Tab. 3 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 \( S_1 \), 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 (\( S_1 \) and \( S_4 \)), 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 \( S_2 \), 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 \( S_2 \). 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 \( S_1 \& S_4 \).

Contrarily, our heuristic scheme can handle all 22 regions \((s,d)\) pairs. As we can see, the FPTAS only works fine for the \( S_1 \) and \( S_2 \) settings. In later comparison, since the travel time of \( F \) is the minimal 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 \( Travel\ time \ of\ SOL = Travel\ time \ of\ F \times 1.10\). 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".

**Table 4:** Description of 6 solutions.

<table>
<thead>
<tr>
<th>Solution</th>
<th>Description</th>
<th>Benchmark</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>Sol. of fastest path with maximal speed</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>( F-SO )</td>
<td>Sol. of fastest path with optimal speed</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>( S )</td>
<td>Sol. of shortest path with maximal speed</td>
<td>Distance</td>
<td></td>
</tr>
<tr>
<td>( S-SO )</td>
<td>Sol. of shortest path with optimal speed</td>
<td>Distance</td>
<td></td>
</tr>
<tr>
<td>( OPT-LB)</td>
<td>Sol. of LB of our heuristic scheme</td>
<td>Fuel</td>
<td></td>
</tr>
<tr>
<td>( OPT-UB)</td>
<td>Sol. of UB of our heuristic scheme</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

### 5.4 Compare Performance with Baselines

In this section, we compare the performance of our heuristic scheme with the following 4 baseline algorithms: (i) fastest (time) path algorithm with maximal speed, (ii) fastest path algorithm with optimal speed, (iii) shortest (distance) path algorithm with maximal speed, and (iv) shortest path algorithm with optimal speed. 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. 4.

In later comparison, since the travel time of \( F \) is the minimal 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 \( Travel\ time \ of\ SOL = Travel\ time \ of\ F \times 1.10\). 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".

Now we input all 22 regions in Fig. 4 as the underlying highway network and use all permutations of the 22 nodes (the nearest points to each individual region) as \((s,d)\) pairs. For each \((s,d)\) pair, we use ten different delays, from \( \lceil T^0 \rceil \) to \( \lceil T^d \rceil + 9 \) where \( T^d \) is the fastest travel time from \( s \) to \( d \).

**A Single Instance:** We first consider one instance \((s,d,T) = (9,22,40)\). Tab. 5 compares the 6 solutions As we can see, our heuristic scheme again outputs the optimal solution. It consumes 300.1 gallons of fuel, runs 10.76% slower than...
Table 3: Comparisons of FPTAS and Heuristic. Here an instance is the tuple (source, destination, delay), i.e., (s, d, T). For example, in S1, (1, 2, 8) means that the source (resp. destination) node is 1 (resp. 2), which is the nearest node to the center of region 1 (resp. region 2) in Fig. 4, and the total delay is 8 hours.

<table>
<thead>
<tr>
<th>No.</th>
<th>Network</th>
<th>Input</th>
<th>Performance (gallon)</th>
<th>Time (second)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reg. n</td>
<td>m</td>
<td>Instance ε</td>
<td>Heuri. LB/UB</td>
<td>FPTAS</td>
</tr>
<tr>
<td>S1</td>
<td>1 &amp; 2</td>
<td>1185</td>
<td>2568 (1, 2, 8) 0.1</td>
<td>74.811/74.811</td>
<td>74.812</td>
</tr>
<tr>
<td>S2</td>
<td>17 &amp; 18</td>
<td>3274</td>
<td>7465 (18, 17, 10) 0.1</td>
<td>60.2795/60.2795</td>
<td>60.2798</td>
</tr>
<tr>
<td>S3</td>
<td>1-22</td>
<td>8274</td>
<td>8276 (4, 22, 40) 0.1</td>
<td>290.744/290.744</td>
<td>365</td>
</tr>
<tr>
<td>S4</td>
<td>1 &amp; 2</td>
<td>1185</td>
<td>2568 (1, 2, 8) 0.05</td>
<td>74.811/74.811</td>
<td>74.812</td>
</tr>
</tbody>
</table>

Tab. 6 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. 6 shows that the fastest path (resp. shortest path) algorithm without speed optimization consumes 20.14% (resp. 16.40%) of more fuels than our solution. 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.

6. 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.

Acknowledgment

The work presented in this paper was supported in part by National Basic Research Program of China (Project No. 2013CB336700) and 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).
7. REFERENCES

[38] E. K. Nam and R. Giannelli. Fuel consumption
modeling of conventional and advanced technology vehicles in the physical emission rate estimator (PERE). *U.S. Environmental Protection Agency*, 2005.


