Transportation Research Part B 153 (2021) 91–110

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

Itinerary planning for cooperative truck platooning

Mojtaba Abdolmaleki, Mehrdad Shahabi, Yafeng Yin *, Neda Masoud

Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI 48109, United States

ARTICLE INFO

Keywords:
Truck platooning
Itinerary planning
Concave minimum cost flow
Outer approximation
Approximation algorithm

ABSTRACT

A cooperative truck platoon is a set of virtually linked trucks driving with a small intra-vehicle headway enabled by connected and automated vehicle technologies. One of the primary benefits of truck platooning is energy savings due to the reduction of aerodynamic drag on the platooned vehicles. The focus of this paper is on scheduling travel itineraries of a given set of trucks to facilitate the formation of platoons to maximize energy savings. By constructing a time-expanded network, we formulate the problem as a minimum concave-cost network flow problem, and devise a few solution methods to find the optimal or high-quality solutions. The solution methods include an outer approximation algorithm for solving a mixed-integer convex minimization reformulation of the problem, a dynamic-programming-based heuristic scalable to large-scale instances, and a fast approximation algorithm with guaranteed performance for a restrictive version of the problem. All the proposed algorithms are examined and benchmarked on medium to large networks under various test scenarios. The numerical results demonstrate the efficiency of the proposed methods and their applicability in real-world settings.

1. Introduction

Cooperative vehicle platooning has been introduced as a promising element of next generation transportation systems. Rendered by connected and automated vehicle technologies (van Wyk et al., 2019; Duret et al., 2020), a platoon is formed when a convoy of vehicles are virtually tethered together and travel on a road with small inter-vehicle gaps. Among different forms of platooning, truck platooning has attracted significant attention recently, as trucks are likely the early adopter of the technology due to their operational characteristics such as long distance and relatively fixed travel routes (Muratori et al., 2017). Upon successful implementation, truck platooning can yield a variety of benefits such as improved fuel economy, reduced labor cost and enhanced safety (Bonnet and Fritz, 2000; Browand et al., 2004; Davila, 2013; Shida et al., 2010; Duret et al., 2020). Although the technology is still at the stage of field experiments, decision makers in many countries have created a road map that consists of regulatory and technological steps needed to launch the technology to the market. For example, the European Union has recently announced a timeline anticipating multi-class truck platooning to hit European highways before 2025.

Given the limited market penetration of connected and automated trucks, systematic itinerary planning is necessary in order to place trucks in spatio-temporal proximity of each other to form platoons. The success of truck platooning, particularly at its early deployment, relies on scheduling trucks' itineraries so that trucks of different types, speeds, paths, and departure times are synchronized to form platoons at platoonable locations, e.g., less-congested freeway segments. An optimized itinerary makes certain that the energy savings from truck platooning are realized while respecting various operational restrictions such as on-time arrival or detour length. Such a scheduled platoon planning problem can typically be investigated from either an offline (pre-trip) or an online (real-time) perspective (Bhoopalam et al., 2018). The scheduled platoon planning problem with flexible routing decisions

* Corresponding author.
E-mail address: yafeng@umich.edu (Y. Yin).

https://doi.org/10.1016/j.trb.2021.08.016
Received 22 July 2020; Received in revised form 13 June 2021; Accepted 31 August 2021
Available online 25 September 2021

was initially proposed by Larson et al. (2013), which considered a piecewise concave energy consumption function for platoon members and proposed a fast heuristic that can handle large fleets. This work was then followed by a few studies that formulated the problem as an integer linear program and solved it with exact solution algorithms (e.g., Larsson et al., 2015; Larson et al., 2016; Nourmohammadzadeh and Hartmann, 2016; Sokolov et al., 2017). Recently, Boysen et al. (2018) considered a general concave cost function to schedule single-class trucks with identical paths to form platoons. They formulated the problem as finding the optimal clique decomposition of an interval graph, and proved that the general case is NP-hard. They further proposed a cubic algorithm in the number of trucks.

The existing literature falls short on two major aspects. First, the majority of the proposed models lack well-structured optimization properties, yielding weak performance when solving large problems (in terms of the number of trucks and the network size). Second, the non-compact nature of the adopted modeling frameworks in the literature has made it necessary to employ restrictive assumptions, such as having a single-class fleet, inflexibility of forming platoons at different links, simplified energy consumption function and network topology, or limited speed choices for trucks. As such, the scheduled platooning planning problem remains unsolvable for realistic instances that comprise of large fleets traveling in large-scale real-world networks.

In contrast, this paper considers the scheduled platoon planning problem without these restrictions. Specifically, we simultaneously optimize the departure time, routing, and speed of trucks to form platoons to minimize their total fuel consumption without violating the travel window of individual trucks. We cast the platooning problem as a concave-cost minimum cost flow problem, and show that this problem under certain nonrestrictive conditions can be reformulated as an equivalent mixed integer nonlinear program. We propose a hybrid outer approximation/local-search solution algorithm to solve the problem. In addition, we prove several efficiency properties for our local search, and demonstrate the superior performance of our proposed approach as compared to existing methods in the literature (Larsson et al., 2015; Larson et al., 2016; Nourmohammadzadeh and Hartmann, 2016; Sokolov et al., 2017). Despite the superior performance of our proposed algorithm, its computational complexity grows exponentially in the size of the problem, limiting its applicability to small- to medium-sized instances. For large-scale problems, we propose a fast dynamic-programming-based heuristic, which significantly improves upon the solution quality of a heuristic proposed by Larson et al. (2013).

We then shift our attention to a special case of the problem considered by several previous studies (Larsson et al., 2015; Larson et al., 2016; Nourmohammadzadeh and Hartmann, 2016; Sokolov et al., 2017; Larson et al., 2013). In this special case, the energy consumption for a platoon of trucks follows a piece-wise linear concave function, which represents a higher consumption rate for the lead vehicle and a lower, uniform consumption rate for all following vehicles.; trucks traverse along paths with minimum energy consumption (hereinafter referred to as energy shortest paths) with pre-specified link speeds; they can wait at intermediate nodes as long as doing so does not compromise their travel time windows. We connect this special case with a well-known network design problem, namely the "Minimum Weight Pairwise Distance Preservers", and propose an efficient algorithm along with its corresponding approximation bounds.

Although we present our work in an offline, pre-trip setting, the proposed algorithms, particularly the dynamic-programming-based heuristic and the approximation algorithm, are efficient enough to support online, real-time decision making. The rest of the paper is organized as follows: Section 2 presents our model formulation, followed by the discussion of the outer approximation algorithm in Section 3. Section 4 introduces the dynamic-programming-based heuristic while Section 5 discusses the approximation algorithm. Section 6 demonstrate the performance of the proposed solution algorithms in different testing scenarios. Lastly, Section 7 concludes the paper.

2. Model

Consider a fleet of trucks comprised of different classes/brands traveling from their origins to destinations. It is assumed that each vehicle reports its origin and destination (OD) and travel time window (earliest departure and latest arrival times) to a central controller before its departure. Hence, trucks' trip schedules are known *a priori*. The controller, as the operator of the fleet, aims to determine the itinerary for each truck to facilitate platooning to minimize the total energy consumption. Each itinerary will specify departure time, route and speed choices at links along the route.

Mathematically, we consider there are C classes of trucks and define $\{K_c \subseteq K : c = 1, 2, \dots, C\}$ to represent the set of trucks in class c, where K is the set of all trucks. For each truck $k \in K$, we use T_k^{ED} and T_k^{LA} to denote its earliest departure and latest arrival times, respectively.

In order to incorporate the travel time windows into a physical network, we formulate the problem leveraging a time-expanded network. In particular, given a physical network, S represents the set of all physical nodes. To construct the time-expanded counterpart of the physical network, we discretize the time horizon into time intervals of length δ_t to form a set $T = \{t_1, t_2, \dots, t_r\}$, where $t_i \in T$ represents the ith ordered time interval. In the time-expanded network G, $N(G)$ and $E(G)$ represent the set of all nodes and links, respectively. We discretize the speed range into a set of discrete speed values V for trucks. A node $n_i \in N(G)$ is defined as a tuple (t_i, s_i), where t_i is the time interval a truck may be located at node $s_i \in S$. Subsequently, a link $l \in E(G)$ is defined as $(n_i, n_j) = (t_i, s_i, t_j, s_j)$, where t_i is the time interval one has to leave node s_i in order to arrive at node s_j during time interval t_j by following a certain speed from the set V on the physical link (s_i, s_j). See Fig. 1 for a demonstration of creating the time-expanded counterpart for a simple, linear physical network. As this figure demonstrates, a truck traversing the same link starting from the same time interval may have multiple links associated with it, depending on its choice of traveling speed.

The above procedure yields a time-expanded network that contains all feasible itineraries for each truck to traverse the network, where, once again, an itinerary specifies a truck's departure time, physical route, and speed choice at each physical link. A pre-processing procedure proposed by Masoud and Jayakrishnan (2017) can then be applied to refine the link set by eliminating a subset of infeasible links (i.e., links that are impossible for a truck to complete its trip within its travel time window). We denote G_k as the simplified spatio-temporal sub-network for truck k. A route in this network corresponds to a feasible itinerary for truck k.

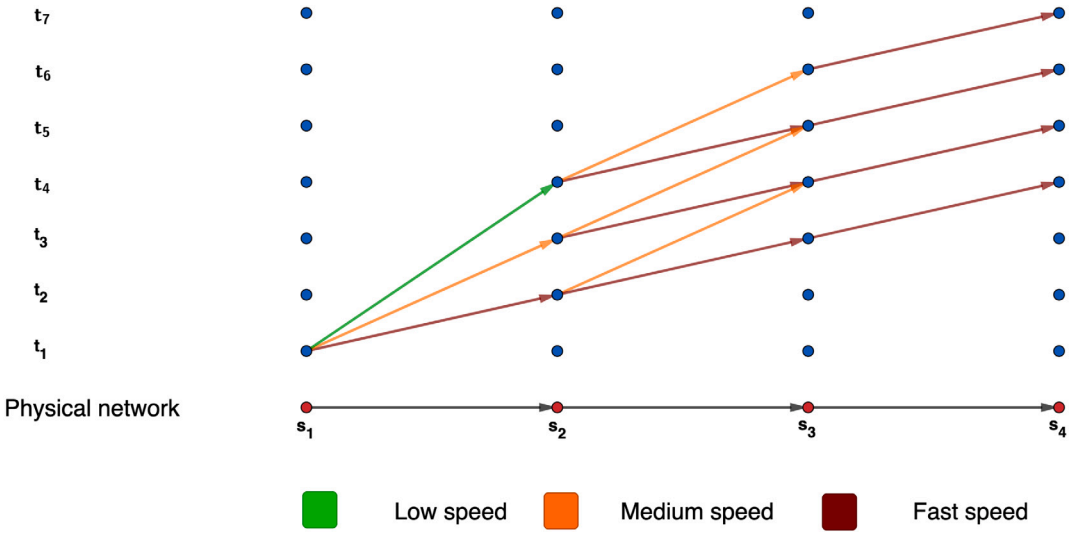

Fig. 1. Constructing time-expanded network.

2.1. Concave cost multicommodity flow formulation

Let us define the cost function $f_l(Y_l)$ to represent the energy consumption of link l, where $Y_l = (y_l^1, y_l^2, \dots, y_l^C)$ and y_l^c is the number of trucks in class c passing through link l. Function f_l is an increasing function with the number of trucks y_l^c. Considering the decreasing trend of marginal energy consumption with an increase in the platoon size, we assume the function f_l to be a jointly concave function. This concavity assumption is consistent with the consideration in previous studies or findings from field experiments (Sun and Yin, 2019; Boysen et al., 2018; Bonnet and Fritz, 2000; Zabat et al., 1995; Larson et al., 2013; Larsson et al., 2015; Larson et al., 2016; Nourmohammadzadeh and Hartmann, 2016; Sokolov et al., 2017). Finally, we assume that platoon formation on a network link is only possible for vehicles traversing the same physical link at the same time and with the same speed.

With the above problem set up, the platoon planning problem becomes equivalent to the problem of finding a route for each individual truck in its time-expanded network G_k such that the summation of f_l over all links $l \in E(G)$, i.e., the network energy consumption, is minimized. This problem is equivalent to a multicommodity flow problem with concave cost function. As such, we can formulate the following optimization problem (Model P1):

$$\text{Min} \quad z = \sum_{l \in L} f_l(Y_l) \tag{1a}$$

$$\text{s.t.} \quad \sum_{\substack{(t_i,s_i):\\ l=(t_i,s_i,t,s)\in E(G_k)}} x_l^k - \sum_{\substack{(t_j,s_j):\\ l=(t,s,t_j,s_j)\in E(G_k)}} x_l^k = d_{t,s}^k \qquad \forall k \in K, \forall (t,s) \in N(G_k) \tag{1b}$$

$$y_l^c = \sum_{k \in K_c} x_l^k \qquad \forall c \in \{1, \dots, C\}, \forall l \in E(G) \tag{1c}$$

$$x_l^k \in \{0,1\} \qquad \forall k \in K, \forall l \in E(G_k) \tag{1d}$$

where

$$d_{t,s}^k = \begin{cases} -1 & \text{If } s = O(k) \text{ and } t = T_k^{ED} \\ 1 & \text{If } s = D(k) \text{ and } t = T_k^{LA} \\ 0 & \text{Otherwise} \end{cases} \tag{1e}$$

The objective (1a) is to minimize the total energy consumption. Constraint (1b) ensures flow conservation for each truck k, where the binary decision variable x_l^k is defined to be 1 if truck k traverses the spatio-temporal link l and 0 otherwise. Constraint (1c) defines y_l^c as the sum of trucks of class c passing through link l. Finally, constraint (1d) specifies the decision variable x_l^k to be binary. Note that if a truck departs at any time interval t_1 later than T_k^{ED}, the first part of the path can be modeled as staying at the physical node $O(k)$ from time interval T_k^{ED} to t_1. The same applies to a truck that finishes its itinerary sooner than the time interval T_k^{LA}.

As formulated, P1 belongs to the family of multi-commodity network flow problems with concave objective function. The formulation enjoys the unimodularity property and can be solved via a nonlinear programming solver. The following lemma formalizes this statement.

Lemma 1 (*Lozovanu, 1982; Ahuja et al., 1988*). *A solution to the continuous relaxation of problem P1 solves P1.*

Proof. First, note that we can substitute the auxiliary variable y_l^c with $\sum_{k\in c} x_l^k$, resulting in only decision variables x_l^k being subject to the assignment constraints (1b). Since the objective function is concave, the solution to the relaxed version of P1 is an extreme point of the feasible region. On the other hand, $d_{t,s}^k$ is integer and the assignment constraints satisfy unimodularity conditions, indicating that every extreme point of the feasible region is integer. □

2.2. Convex minimization reformulation

The concave minimization nature of Model P1 prevents a commercial nonlinear solver from solving large-scale instances of the problem, necessitating specialized algorithms, such as the piece-wise linear approximation by Magnanti and Stratila (2004). In this section, we reformulate P1 as a convex minimization problem that facilitates the development of an outer approximation algorithm in Section 3.

Making use of the fact that the decision variable x_l^k is binary (indicating that $x_l^k = \left(x_l^k\right)^n, \forall n \in \mathbb{R}^+$), and further assuming that the function f_l is twice differentiable and the diagonal elements of its Hessian matrix are finite, we can show that the following function g_l is strongly convex (Murray and Ng, 2010):

$$g_l(x_l^1, x_l^2, \dots, x_l^K) = f_l\Big(\sum_{k\in K_1} x_l^k, \sum_{k\in K_2} x_l^k, \dots, \sum_{k\in K_c} x_l^k\Big) + M \sum_{k\in K} \Phi(x_l^k) \tag{2}$$

where $\Phi : [0,1] \to \mathbb{R}$ is a strictly convex function with $\Phi(0) = \Phi(1) = 0$. In addition, M is a sufficiently large positive number. The basic idea is to augment the original objective with a strictly convex function to ensure convexity while making sure the objective function value is intact. We refer the readers to Murray and Ng (2010) for more details on finding suitable values for M and various forms of function Φ.

By further introducing a continuous variable Q_l, we reformulate P1 into the following convex mixed integer nonlinear program (MINLP) (Model P2):

$$\text{Min} \quad z = \sum_{l\in L} Q_l \tag{3a}$$

$$\text{s.t.} \quad \sum_{\substack{(t_i,s_i):\\ l=(t_i,s_i,t,s)\in E(G_k)}} x_l^k - \sum_{\substack{(t_j,s_j):\\ l=(t,s,t_j,s_j)\in E(G_k)}} x_l^k = d_{t,s}^k \qquad \forall k \in K, \forall (t,s) \in N(G_k) \tag{3b}$$

$$g_l(x_l^1, x_l^2, \dots, x_l^K) \le Q_l \qquad \forall l \in E(G) \tag{3c}$$

$$x_l^k \in \{0,1\} \qquad \forall k \in K, \forall l \in E(G_k) \tag{3d}$$

This reformulation enjoys desirable mathematical properties that make it amenable to decomposition algorithms, such as outer approximation.

3. Outer approximation

Decomposition techniques such as outer approximation (OA) have been widely employed to solve MINLPs. OA belongs to the family of decomposition and cutting-plane methods, which separate the original MINLP into a nonlinear program (NLP) subproblem and a relaxed version of the original problem known as the master problem (MP). In OA algorithm, the nonlinear subproblem is achieved by fixing the integer variables, and therefore provides an upper bound solution. The master problem on the other hand, is obtained by sequentially substituting the nonlinear convex function by its first-order linear, or in some cases second-order quadratic, approximations. MP, which is responsible for a lower bound solution, is therefore tightened sequentially by adding OA cuts for the convex nonlinearities at every iteration. The algorithm cycles between the subproblem and the master problem until the gap between the upper and lower bounds meets the convergence criterion, or the maximum number of iterations is reached. Given the convexity of the formulation, and the fact that the OA cuts are under-estimators of the convex function, OA converges to the global optimal solution of the original problem in a finite number of iterations. Below we will demonstrate that the application of OA to the MINLP model P2 is suitable, owing to its closed-form solution for the subproblem.

Proposition 1. *In P2, the NLP subproblem for OA has a closed-form solution.*

Proof. The NLP subproblem is achieved by fixing the binary integer variables $\bar{x}_l^k$. Once the integer variables are fixed, the optimal value of the continuous variable Q_l is obtained via the following equation:

$$\bar{Q}_l = g_l(\bar{x}_l^1, \bar{x}_l^2, \dots, \bar{x}_l^K) \tag{4}$$

3.1. Master problem

The master problem in OA (OA-MP) is a mixed integer linear program achieved by sequentially approximating the nonlinear constraint (3c) with its sub-gradient inequalities at any given integer point $\bar{x}_l^{k^h}$. In particular, given a feasible integer solution at every iteration h, a linear OA cut in the form of Eq. (5c) is generated to augment the sets of previously generated cuts in order to linearly approximate the convex constraint (3c). We should point out that since sub-gradient-based OA cuts (Eq. (5c)) are the under-estimator of the convex functions, they will not cut-off the optimal point(s). These equations are merely responsible for cutting off those solutions that do not belong to the linear approximation of the convex feasible region at every iteration.

As mentioned earlier, the lower bound for the OA algorithm is achieved by solving the master problem. In order to further improve the overall efficiency of the OA algorithm, Fletcher and Leyffer (1994) showed that the OA-MP does not have to be solved to optimality, and the algorithm can continue as long as a new integer solution can be found by the MP at every iteration. In this case, Fletcher and Leyffer (1994) suggested adding Eq. (5d) to ensure that the MP solution remains at most less than the upper bound. They also proved that by adding such a constraint, no integer solution is replicated by OA-MP twice. Therefore, if at any iteration no new integer solution is generated by OA-MP, the master problem becomes infeasible, at which point an ϵ-optimal solution is reached and the algorithm terminates. We should note that ϵ is a desired level of accuracy and is supplied by the user. Below, the master problem or OA-MP is presented:

$$\text{Min} \quad z = \sum_{l \in L} Q_l \tag{5a}$$

$$\text{s.t.} \quad \sum_{\substack{(t_i,s_i):\\ l=(t_i,s_i,t,s)\in E(G_k)}} x_l^k - \sum_{\substack{(t_j,s_j):\\ l=(t,s,t_j,s_j)\in E(G_k)}} x_l^k = d_{t,s}^k \qquad \begin{matrix}\forall k \in K,\\ \forall (t,s) \in N(G_k)\end{matrix} \tag{5b}$$

$$g_l(\bar{x}_l^{1^h}, \bar{x}_l^{2^h}, \ldots, \bar{x}_l^{K^h}) + \sum_{k \in K} \frac{\partial g_l(\bar{x}_l^{1^h}, \bar{x}_l^{2^h}, \ldots, \bar{x}_l^{K^h})}{\partial x_l^k}(x_l^k - \bar{x}_l^{k^h}) \le Q_l \qquad \begin{matrix}\forall l \in E(G),\\ \forall h = 1,2,\ldots,H\end{matrix} \tag{5c}$$

$$\sum_{l \in L} Q_l \le UB - \epsilon \tag{5d}$$

$$x_l^k \in \{0,1\} \qquad \begin{matrix}\forall k \in K,\\ \forall l \in E(G_k)\end{matrix} \tag{5e}$$

3.2. Revising the master problem

The integer points provided by OA-MP are responsible for providing the OA lower bound and generating a new sets of cuts for the next iteration. Generally, the performance of the algorithm heavily depends on the quality of the integer solution provided by OA-MP. While the above formulation of OA-MP can be handled by off-the-shelf optimization solvers such as CPLEX, its continuous relaxation is still weak and can be further improved to enhance the quality of the integer solutions returned by OA-MP. This step is critical in the overall OA algorithm since achieving better bounds for MP at every step would potentially lead to fewer iterations for the overall algorithm. To this end, we reformulate constraint (5c) as follows:

$$g_l(\bar{x}_l^{1^h}, \bar{x}_l^{2^h}, \ldots, \bar{x}_l^{K^h}) + \frac{\partial g_l(\bar{x}_l^{1^h}, \bar{x}_l^{2^h}, \ldots, \bar{x}_l^{K^h})}{\partial x_l^k}(x_l^k - \bar{x}_l^{k^h}) \le Q_l, \ \forall l \in E(G), \forall h = 1,2,\ldots,H, \forall k \in K \tag{6}$$

Eq. (6) is the disaggregated version of constraint (5c), which would provide a tighter continuous relaxation for OA-MP. Note that replacing equation (5c) with (6) does not change the feasible integer space of the original OA-MP, but would help find better integer solutions in a reasonably shorter time. We should point out that the continuous variables generated through two formulations are not the same. However, according to Fletcher and Leyffer (1994), OA-MP is responsible for generating a new integer variable at every iteration and the continuous variables are optimized through the subproblem.

3.3. Local search

Reformulating the problem P(1) as a convex optimization problem P2 comes at a cost of losing some of the structural properties of the original problem, such as the unimodularity of the constraint set as described in Lemma 1. In this section, we utilize the concavity of the original problem to further improve the quality of the OA-MP integer solutions by solving a sequence of linear programs that are computationally inexpensive. We should first point out that both P(1) and P2 have the same objective function values at the extreme points of the feasible region. Additionally, due to the fact that the original problem P(1) is a concave minimization problem, the first-order Taylor expansion of the objective function at any point, $\bar{x}$, is an over-estimator of the objective function. Therefore, any solution x to the linear program (7) has an objective value $z(x)$ satisfying $z(x) \le z(\bar{x})$, a reduction in the objective function value.

$$\min z = \sum_{l \in L}(f_l(\bar{x}_l^{1^h}, \bar{x}_l^{2^h}, \ldots, \bar{x}_l^{K^h}) + \sum_{k \in K} \frac{\partial f_l(\bar{x}_l^{1^h}, \bar{x}_l^{2^h}, \ldots, \bar{x}_l^{K^h})}{\partial x_l^k}(x_l^k - \bar{x}_l^{k^h})) \tag{7a}$$

$$\text{s.t.} \quad \sum_{\substack{t_i,s_i:\\ l=(t_i,s_i,t,s)\in L}} x_l^k - \sum_{\substack{t_j,s_j:\\ l=(t,s,t_j,s_j)\in L}} x_l^k = d_{t,s}^k \quad \forall k \in K, \forall (t,s) \in N(G) \tag{7b}$$

$$0 \le x_l^k \le 1 \qquad \forall k \in K, \forall l \in E(G_k) \tag{7c}$$

In the above formulation, the objective, (7a), is the first-order Taylor approximation of the function at point x around the point $\bar{x}$. Constraints (7b) reflect the flow balance constraints as in the original problem.

Note that the solution x^* of the optimization problem (7) is not necessarily a local optimal of P1. However, after solving a series of optimization problems (7) where x^* is replaced with $\bar{x}$, we can guarantee that we are sufficiently close to a local optimal. Therefore, following this procedure, we can iteratively improve our solution locally until we reach a locally optimal solution. In practice, we only need a few iterations, even in large-scale networks. Theorem 1 provides a polynomial upper bound on the number of times we have to solve problem (7) to obtain a locally optimal solution. Denote by $|P|$ the maximum number of vehicles that can pass through a link in the time-expanded network without violating their travel time windows constraints. Let $|E|$ be the number of edges in the time-expanded network.

Theorem 1. *For the multi-class platooning problem, if the functions $f_l(.)$ are all strongly concave, we obtain a locally optimal solution after at most $O(|E \parallel P|)$ times solving the problem (7). Moreover, for the single-class platooning problem, if the functions $f_l(.)$ are all concave, we obtain a locally optimal solution after at most $O(|P|^{5+\epsilon}|E|^3)$ times solving the problem (7), for any $\epsilon > 0$.*

Proof. See Appendix B. □

Algorithm 1 Outer Approximation

1: **Procedure OA(ϵ)**
2: **Set the maximum number of iterations as *H***
3: **Initialize: Set UB=$+\infty$**
4: **Find an initial assignment $\bar{x}_l^{k^1}$ by finding a shortest path for every truck k**
5: **while** $h \le H$ *and OA-MP is feasible* **do**
6: **1: OA Subproblem** :
7: *1.1: Calculate the subproblem objective function by Eq. (4)*
8: *1.2: Update the OA upper bound if $\bar{Q}_l \le UB$.*
9: **2: OA Master Problem**:
10: **if** *OA-MP objective (Eq. (5d))*$\ge$ *UB-ϵ* **then**
11: *OA-MP is infeasible, terminate the OA algorithm*;
12: **end if**
13: *2.1: Add the OA inequalities (Eq. (6))*;
14: *2.2: Find a feasible solution of OA-MP (Eqs. (5a), (6), (5b), (5e))*;
15: *2.3: Locally improve the solution of OA-MP by solving linear programs (7)*;
16: *2.4: Increment h*;
17: **end while**

4. Dynamic-programming-based heuristic

We have proposed the OA algorithm as an exact algorithm for the scheduled platoon planning problem. However, as the size of the problem (in terms of network and fleet size) grows, the number of variables and constraints in Model P1 increases exponentially, making it challenging for the proposed exact algorithm to solve those large-scale instances effectively. In this section, we devise a dynamic-programming-based heuristic to find high-quality solutions in a considerably lower amount of time, which is particularly useful for solving realistic problems in real time.

The core idea of our heuristic is to use a dynamic programming (DP) algorithm to iteratively find a hypothetical path in the time-expanded network that yields the maximum platoon savings along its links. Specifically, we aim to find a path p and force each truck k to pass through the overlap of its restricted subgraph G_k and path p. Once truck k joins path p, the platoons on the overlapping links will be updated, rewarding truck k with energy savings from joining a platoon. Note that path p does not necessarily contain the origins and destinations for all trucks, and platoon savings can be still achieved if a vehicle subgraph only partially overlaps with path p. In this section, we devise a DP algorithm that approximately finds path p that results in maximum platoon savings. Before introducing the algorithm, we should note that since each truck can have a different set of allowable speed choices for each link, there is no guarantee that the overlap of path p with subgraph G_k is a connected path. It may be a union of subpaths of p, as demonstrated in Fig. 2. In this figure, the red dashed track is the hypothetical path p while the green one is the restricted subgraph of a truck k who has to start its trip at time $t = 1$ and finish at time $t = 7$, and its allowable set of speed choices does not allow it to follow the hypothetical path p over its entire path.

Let us define a state in our DP algorithm as a spatio-temporal node (t_i, s_i). Furthermore, let $V(t_i, s_i)$ be the maximum savings for the system that can be achieved by following a path from an arbitrary node to spatio-temporal node (t_i, s_i). The goal is to find the node (s^*, t^*) such that $(t^*, s^*) = \text{argmax}_{(t,s)\in N(G)} V(t, s)$.

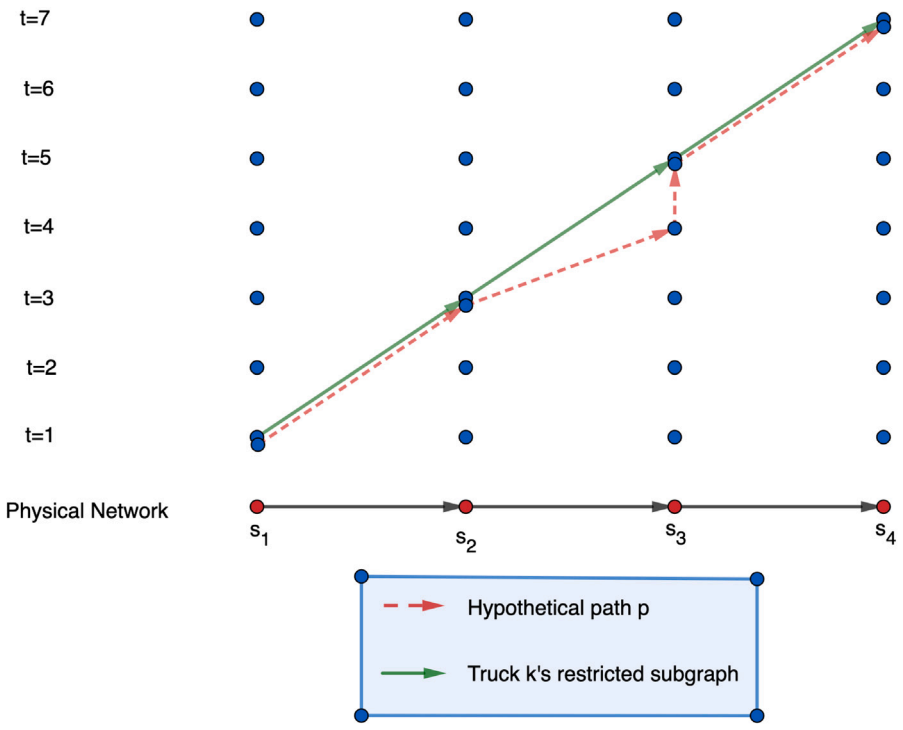

Fig. 2. The discontinuity of the overlap of path p and the restricted subgraph of a truck G_k.

We define two tensors, M_1 and M_2, to ensure the feasibility of the optimal path and facilitate the retrieval of the maximum-savings path after its cost is computed. Each element of the tensor $M_1(t,s)$ is a state (t',s') that precedes state (t,s) in the optimal path leading to the state (t,s). Also, each element of the tensor $M_2(t,s,i)$ denotes the vector of trucks that have contributed to energy savings along the optimal path ending at state (t,s) at a link that lies i steps prior to the link $(M_1(t,s),t,s)$ on the optimal path.

Let $w(t',s',t,s)$ denote the net energy savings for link (t',s',t,s). We compute $w(t',s',t,s)$ as follows. The first step is to compute the platoon energy savings for the set of trucks U, passing through link $l=(t',s',t,s)$. This can be achieved by subtracting the energy consumption for trucks in U passing through link l, $f_l(U)$, from the total energy consumption for set U of trucks passing through l while no platooning is considered. The second step is to compute the excess energy consumption incurred by the detour that each truck $k \in U$ has to take to pass through l, $u_d^l(k)$. We subtract the energy consumption of the energy-minimizing path for the truck k that includes the node (t',s') (in the absence of any platoon savings) from the energy consumption of the alternative energy-minimizing path that places k on the segment l along its destination. The excess energy consumption imposed by detours for the set U can then be obtained by summing up $u_d^l(k)$ over all the vehicles in the set U.

Note that $w(t',s',t,s)$ equals the maximum net energy savings for link $l=(t',s',t,s)$ among all possible subsets U of the trucks. To determine if a truck k can pass through l we check tensors M_1 and M_2 to find the latest state truck k appeared on the optimal path to state (t,s). Then we check if it is feasible for k to pass through l on its trip to its destination without violating the time constraint. Lemma 2 shows how we can efficiently compute the value of $w(t',s',t,s)$.

Lemma 2. *The maximum net energy savings $w(t',s',t,s)$ for link (t',s',t,s) is calculated with $\prod_{i=1}^{C} r_i$ operations for a concave energy consumption function, where r_i is the number of trucks in class i that can pass through link (t',s',t,s).*

Proof. For each class $c \in \{1,2,\dots,C\}$, we sort the trucks $k \in K_c$ based on their corresponding excess energy consumption imposed by detours as:

$$K_c^l = \{k_1, k_2, \dots, k_r\}$$

where k_1 has the minimum excess energy consumption among trucks in K_c^l. For any subset of trucks $U_1 \subseteq K_c^l$ where we have $k_i \notin U_1, k_j \in U_1, u_d^l(k_i) < u_d^l(k_i)$ we can substitute the trucks k_i and k_j to get a new combination with a higher net energy savings, U_2, because the platooning energy savings for the sets U_1 and U_2 are the same while the excess energy consumption imposed by detour has been decreased by $u_d^l(k_j) - u_d^l(k_j)$. Keeping this observation in mind, we can conclude that the overlap of the optimal subset U^* with any set K_c^l is a set of consecutive trucks $U^* \bigcap K_c^l = \{k_1, k_2, \dots, k_{r^*}\}$, which always have the truck k_1 if it is not the empty set. That being said, the number of all possible such sets can be computed as the product $\prod_{i=1}^{C} r_i$. Computing the net energy savings for all of these special sets results in finding the subset U^* with maximum net energy savings on link l. □

Note that in practice, the number of classes may not exceed three or four types, and for the single-class fleet case we can find the $w(t', s', t, s)$ in linear time.

To initialize the DP process, we set the value of $V(t, s)$ with states $\{(t, s) : t = 0\}$ to be zero. Next, we traverse the states in the topological order of nodes to find the value function for each state. The Bellman equation displayed in (8) describes how we update the value function at each step. The set $N_d(t, s)$ in this equation contains all nodes in the time-expanded subgraph that directly are connected to node (t, s).

$$V(t_j, s_j) = \max_{(t_i, s_i) \in N_d(t_j, s_j) : t' \leq t} (w(t_i, s_i, t_j, s_j) + V(t_i, s_i)) \tag{8}$$

Note that after enforcing a trip to pass through subpaths of p, we can partially capture the remaining flexibility of the trip by introducing new trips that connect the remaining portions of the original trip. As such, we first introduce the sets A_k as the union of the two nodes s_k^o and s_k^d in addition to the end points of subpaths on the optimal path p, p_k^i, that truck k passes through. Next, we sort these fixed nodes that lie on truck k's path based on their time component, and for any consecutive pair of these nodes that are not readily connected through one of the subpaths p_k^i, we introduce a new (dummy) truck with those end points as its origin and destination. We iteratively run the DP process until the optimal value for the process equals 0, i.e., we cannot achieve any additional saving from platooning. Algorithm 2 presents the pseudo code for the DP-based heuristic described above:

Algorithm 2 Heuristic Algorithm

```
Stop = 0;
while Stop ≠ 1 do
    Stop = 1
    for k ∈ the set of all the trips in the system do
        Generate the restricted subnetwork for trip k following the procedure in 2.1;
    end for
    Run the DP process introduced above
    let f be the optimal value of the DP process.
    if f > 0 then
        Stop = 0
    end if
    for k ∈ set of all the trucks in the system do
        Identify the set A_k and the consecutive endpoints of A_k that are not connected by one of the subpaths p_k^i. For each one of
        them, define a new truck with those endpoints as its OD pair.
        Delete k from the set of trucks.
    end for
end while
```

5. Approximation algorithm

In this section, we propose a linear-programming-based algorithm for a special case of the scheduled platoon planning problem. Compared with the general case discussed above, we herein assume the energy consumption function for a platoon of trucks traveling through a link is a piecewise linear concave function with two pieces on the number of trucks in the platoon. More specifically, it postulates that the leading truck in a platoon does not save any energy while the following trucks, irrespective of their position in the platoon, experience an energy saving factor of $0 \leq \alpha \leq 1$, e.g., $\alpha = 0.1$, compared to their original energy consumption (Larson et al., 2013). This function can be viewed as an approximation to the general concave function considered above, and is particularly valid when all trucks are of the same type, forming single-class platoons as known in the literature. We also assume that the objective of trucks is to travel along energy-minimizing paths, paths that minimize their energy consumption in the absence of any platooning benefit, given pre-specified link speeds. However, they can wait at intermediate nodes as long as doing so does not compromise their travel time windows.

This special case, considered by several previous studies (Larson et al., 2013; Larsson et al., 2015; Larson et al., 2016), deserves special attention. First, it remains a reasonable simplification to the general scheduled platoon planning problem as the assumptions are not overly restrictive. The simplification makes the model parameters much easier to calibrate or specify, yielding a model particularly appropriate for a sketch or strategic planning exercise to obtain the ballpark estimate of the benefit of truck platooning. Second, the problem possesses an interesting structure that allows us to cast it as the problem of Minimum Weight Pairwise Distance Preservers in the network design literature, enabling the development of an efficient algorithm with approximation bounds.

5.1. Model

To form the restricted sub-network of a truck k, G_k, we modify the pre-processing procedure proposed in Masoud and Jayakrishnan (2017) by discarding the links that do not lie on any energy-minimizing path that connects the origin of the truck to its destination and satisfies their travel time window constraints.

Following the model presented in Section 2.1, the itinerary for every truck k can be demonstrated by a path in the time-expanded network G. This path will start at the node $n_k^o = (T_k^{ED}, O(k))$ and ends at the node $n_k^d = (T_k^{LA}, D(k))$. Finally, the objective is to find a directed path for each truck in its restricted time-expanded network to minimize the total energy cost.

Note that every directed path from n_k^o to n_k^d in G_k is an energy shortest path, because G_k is the restricted network of truck k, only containing links that lie on energy-minimizing paths. Because for each link in the time-expanded network we have a unique pre-specified speed, we can denote by $e(s_i, s_j)$ the unique amount of energy required to pass through the link $l = (t_i, s_i, t_j, s_j)$. As such, the total energy consumption in the absence of any platoon savings is constant, given as follows:

$$\sum_{l=(t_i,s_i,t_j,s_j)\in E(G)} \sum_{k\in K} x_l^k \; e(s_i, s_j) = \sum_{k\in K} \sum_{l=(t_i,s_i,t_j,s_j)\in E(G)} x_l^k \; e(s_i, s_j) = \sum_{k\in K} e(s_k^o, s_k^d) \tag{9}$$

As assumed previously, platooning yields zero saving for the leading truck and the same percentage of energy savings for the following trucks regardless of their position in the platoon. The total energy consumption can thus be written as in Eq. (10).

$$\sum_{l=(t_i,s_i,t_j,s_j)\in E(G)} [\sum_{k\in K} (1-\alpha) \; x_l^k \; e(s_i, s_j) + \sum_{l=(t_i,s_i,t_j,s_j)\in E(G)} \alpha \; \eta_l \; e(s_i, s_j)] \tag{10}$$

where η_l is a binary variable indicating whether there exists at least one truck passing through link l. The first term is a factor $(1-\alpha)$ of the total energy consumption without platooning. This term essentially captures the total energy consumption of all following trucks, plus $(1-\alpha)$ of the leading trucks' consumption. The second term captures the remaining α factor of the energy consumption of the leading trucks (note that a single truck is viewed as the leader of a one-member platoon). Given the first term is constant, Model P1 reduces to the following optimization problem:

$$\min \quad z = \sum_{l=(t_i,s_i,t_j,s_j)\in L} \eta_l \; e(s_i, s_j) \tag{11a}$$

$$\text{s.t.} \quad \sum_{\substack{t_i,s_i:\\ l=(t_i,s_i,t,s)\in L}} x_l^k - \sum_{\substack{t_j,s_j:\\ l=(t,s,t_j,s_j)\in L}} x_l^k = d_{t,s}^k \qquad \begin{matrix}\forall k \in K,\\ \forall (t,s) \in N(G_k)\end{matrix} \tag{11b}$$

$$x_l^k \le \eta_l \qquad \forall l\in E(G), \forall k: G_k \bigcap l \neq \emptyset \tag{11c}$$

$$x_l^k \in \{0,1\} \qquad \begin{matrix}\forall k \in K,\\ \forall l \in E(G_k)\end{matrix} \tag{11d}$$

$$\eta_l \in \{0,1\} \qquad \forall l\in E(G) \tag{11e}$$

where

$$x_l^k = \begin{cases} 1 \text{ If } k \text{ travels on } l \\ 0 \quad \text{Otherwise} \end{cases} \tag{11f}$$

$$d_{t,s}^k = \begin{cases} -1 & \text{If } s = O(k) \& t = T_k^{ED} \\ 1 & \text{If } s = D(k) \& t = T_k^{LA} \\ 0 & \text{Otherwise} \end{cases} \tag{11g}$$

Viewing the link energy consumption of a truck $e(s_i, s_j)$ as a link cost, the above formulation essentially finds, in the time-expanded network G, a sub-network Z that contains, for each truck k, a directed path in G_k from its origin to destination while minimizing the sum of all link costs in Z. Below we further connect the problem with Minimum Weight Pairwise Distance Preserver (Coppersmith and Elkin, 2006; Elkin and Peleg, 2007).

Definition 1. Given a directed graph $G = (V, E)$, with non-negative edge costs $c : E \to R^+$, and a collection of node pairs $P \subseteq V \times V$, we say a subgraph Z of G is a Minimum Weight Pairwise Distance Preserver of G if it minimizes the total link costs, $\sum_{e\in E(Z)} c(e)$, while the distance from s to t in Z is the same as that in G.

It is trivial to observe that finding the optimal solution to the platooning problem is equivalent to finding the Minimum Weight Pairwise Distance Preservers in the undirected counterpart of the time-expanded network, and the set of unions of OD pairs is the set P in Definition 1. Hardness for distance preservers can be established by applying the technique provided in Elkin and Peleg (2000) for proving hardness of the client–server spanner problem.

In addition, the objective function (11a) can be equivalently written as maximizing the total fuel consumption savings, which is α times the total energy consumption of all following trucks (without platooning). After dropping the coefficient α, the new objective function becomes (12a).

$$\text{Max} \quad z = \sum_{l=(t_i,s_i,t_j,s_j)\in L} (\sum_{k:\, l\in G_k} x_l^k - \eta_l) e(s_i, s_j) \tag{12a}$$

$$\text{s.t.} \quad (11b) - (11e) \tag{12b}$$

Below we will develop an approximation algorithm to solve the problem, and provide bounds on the quality of the solution.

5.2. LP relaxation

The Pairwise Distance Preserver problem has been shown to have tight integrality gap for their LP relaxations (Chlamtáč et al., 2017). Below we propose an LP-relaxation-based approximation algorithm. In the literature, in addition to such an LP relaxation approach, another decomposition approach is often used to solve the Pairwise Distance Preservers problem. However, as we illustrate with an example in Appendix C, most of the decomposition structures previously investigated in the literature are not suitable to approximate the objective function (12a).

For our LP relaxation, we follow the work of Bodwin and Williams (2016) by relaxing the integrality of the variables η_l and x_l. The relaxed variables η_l can be interpreted as the capacity of the link l and the relaxed variables x_l as the flow associated with trucks passing through the link.

$$\text{Max} \quad z = \sum_{l=(t_i,s_i,t_j,s_j)\in L} \Big(\sum_{k:\, l\in G_k} x_l^k - \eta_l \Big)\, e(s_i, s_j) \tag{13a}$$

$$\text{s.t.} \quad \sum_{\substack{t_i,s_i:\\ l=(t_i,s_i,t,s)\in L}} x_l^k - \sum_{\substack{t_j,s_j:\\ l=(t,s,t_j,s_j)\in L}} x_l^k = d_{t,s}^k \qquad \begin{array}{c}\forall k \in K,\\ \forall (t,s) \in N(G_k)\end{array} \tag{13b}$$

$$x_l^k \le \eta_l \qquad \begin{array}{c}\forall k : G_k \bigcap l \neq \emptyset,\\ \forall l \in E(G_k)\end{array} \tag{13c}$$

5.3. Approximation algorithm

The core idea of the approximation algorithm is the same as the one described in Section 4 for our DP algorithm. However, here we use the LP relaxation (13) to modify the restricted subgraphs of the trucks in the system.

After solving the LP relaxation problem, we first construct a prioritized subgraph G_k^* for each truck k from the union of links l with $x_l^k > 0$ in the subgraph G_k. Doing so avoids forming greedy platoons, which may eliminate the possibility of forming a globally optimal platooning schedule (See Section 5.4 for an illustration of this phenomenon). After forming the subgraphs G_k^*, we assign non-negative weights to the edges of the original graph to establish the new time-expanded network G^*. If the edge $l = (t_i, s_i, t_j, s_j) \in L$ is in r_l number of subgraphs G_k^*, we assign the weight $p_l = (r_l - 1)e(s_i, s_j)$ to the edge l. The weight p_l represents the potential of link l for saving energy because r_l trucks can potentially pass it through and form a platoon.

After creating the weighted graph G^* we try to find the longest path in G^* and denote it by p. Note that forcing all trucks to pass through the overlap of their subgraph G_k^* and p will result in maximum platoon savings on a spatio-temporal path. Furthermore, note that the overlap of G_k^* and p is a subpath of the path p. Then, for each truck k whose subgraph G_k^* has a nonempty overlap with p, we delete k from the set of trucks and add two other trucks, k_1 and k_2. The origin of the truck k_1 is the same as that of k, and the destination of k_1 is the node with the smallest timestamp that lies on the overlap of truck k's subnetwork with p. Analogously, the origin for truck k_2 is the node with the largest timestamp that lies on the overlap of truck k's subnetwork with the path p, and the destination of k_2 is the same as that of truck k. This approach allows us to take advantage of the remaining flexibility of truck k's trip—the part of the trip that is not intersecting with p.

Before going any further we provide the pseudo code for our proposed approximation algorithm as follows:

Algorithm 3 Approximation Algorithm

```
Stop = 0;
while Stop ≠ 1 do
    Stop = 1
    for k ∈ K do
        Generate the energy shortest path subnetwork for truck k following the procedure described in Section 5.1;
    end for
    Solve the LP relaxation proposed in Section 5.2
    Modify the restricted subgraphs based on the optimal solution to the LP relaxation to obtain the subgraphs G_k^*.
    Form the weighted directed graph G^* from the subgraphs G_k^*.
    Let p and f be the longest path and its weight in G^*, respectively.
    if f > 0 then
        Stop = 0
    end if
    for k ∈ the set of all trucks in the system do
        Identify the overlap of the path p and G_k^*. Define two new trucks, k_1 and k_2, as described earlier in Section 5.3.
        Delete k from the set of trucks.
    end for
end while
```

Theorem 2. *The approximation algorithm has a $|K|$-approx guarantee, where $|K|$ is the number of trucks in the system.*

Proof. It is sufficient to prove that there exists a path that obtains at least $\frac{1}{|K|}$ of the total savings in the optimal solution. We first generalize the notion of savings to the continuous relaxation setting. To do so, we follow the definition of the savings in a link l in the binary optimization problem and compute the total savings as $\left(\sum_{k:l\in G_k} x_l^k - \eta_l\right) \alpha\, e(s_i, s_j)$. For the sake of simplicity and without loss of generality, we set $\alpha = 1$. We introduce a new variable s_l to denote the savings on link l, and let s_l^* denote the energy cost of link l on the energy-minimizing path in the LP relaxation problem. Now, denote the optimal solution to the LP by (x^*, y^*). We introduce r_l as the number of trucks with $x_l^{k*} > 0$ and r_l^* for the corresponding value at the optimal solution to the LP. For any link-based solution, (x^*, y^*), there exists an equivalent path-based solution f^* where f_p^{k*} is the flow on path p for truck k. Let us assume we choose one of the trucks with equal probability $\frac{1}{|K|}$ and a path for truck k with $f_p^{k*} > 0$, and force all other trucks to pass through the overlap of their restricted subgraph and the path p. Note that this overlap is a subpath of p. As such, we can compute the probability that a link l is being taken in the resulting randomized path by the summation over all different $|K|$ scenarios of choosing any of the $|K|$ trucks. Moreover, the savings we can obtain if we pass through l is the unique value $(r_l^* - 1)\ e(s_i, s_j)$ and does not depend on which scenario result in picking l. Putting it together, we can compute the expected savings for a given link l as follows:

$$\mathbf{E}[s_l] = \frac{1}{|K|}\sum_{k=1}^{|K|} x_k^{l*}(r_l^* - 1)\ e(s_i, s_j) = \frac{(r_l^* - 1)e(s_i, s_j)}{|K|}\sum_{k=1}^{|K|} x_k^{l*}$$

As $\eta_l^* \geq 0$, we have:

$$\geq \frac{r_l^* - 1}{|K|}\Big(\sum_{k=1}^{|K|} x_k^{l*} - \eta_l^*\Big)\ e(s_i, s_j)$$
$$\geq \frac{r_l^* - 1}{|K|} s_l^* \geq \frac{1}{|K|} s_l^*$$

Summing up over all links $l \in E(G)$ yields:

$$\sum_{l\in E(G)} \mathbf{E}[s_l] \geq \frac{1}{|K|}\sum_{l\in E(G)} s_l^*$$

Moreover, the savings in the LP relaxation formulation is an upper bound on the savings for the integer programming formulation, so the proof is complete. □

It is trivial to observe that the approximation algorithm will obtain the optimal solution for the two-truck case.

5.4. Role of LP relaxation

This subsection compares the performance of the approximation algorithm without LP relaxation versus that with LP relaxation. With an illustrative example, we show that the greedy longest path algorithm provides $\frac{1}{k}$ of the optimal savings, while the longest path equipped with the LP relaxation provides the optimal solution.

Definition 2. We define a strip quadrilateral SQ_n to be a graph that is obtained by adding two new vertices, say x_i, y_i, for each pair of adjacent vertices $(u, v) \in P_{n+1}$, the simple path of length n, and replacing the edge (u, v) by the union of four edges $(u, x_i), (x_i, y_i), (v, y_i), (u, v)$ on u, v.

Introducing SQ_n^*, the weighted directed graph is obtained from SQ_n where in each quadrilateral (u, x_i, y_i, v) the weight of the edges (u, v) and (u, x_i) is 1 and the weight of the two other edges is k. Moreover, all the edges in the underlying directed path P_{n+1} are directed from one of the endpoints of P_{n+1} to the other one. The other edges are directed towards their corresponding y_i vertex of the quadrilateral they belong. Fig. 3 demonstrates the structure for SQ_n^*. Now, let us consider the case where we have $2n+1$ trips. The first trip is between the two endpoints of the strip quadrilateral. Moreover, for any quadrilateral (u, x, y, v) in SQ_n^* we have two associated trips, one from u to y and the other from x to y. We also consider the weight of an edge as the spatial distance represented by that edge. Also, note that for the trips that have length k, the choice of a shortest spatial route is unique, while for the trips with length $k+1$ there are two choices for the spatial shortest route. The origins and destinations have been depicted in Fig. 3. To avoid unnecessary complexity, we have not depicted the time dimension for these trips. However, we can assume that all these trips are completely flexible, meaning that their corresponding time window does not put any restriction on their route choice. For a value of n satisfying $n = k + 1$, applying the greedy algorithm without the LP relaxation to solve (12) yields a solution where the flexible trips are forced to pass through their overlap with the P_{n+1}. This results in a total savings of n. However, if the flexible trips choose their other route, the total savings will be nk. It is not hard to see the solution to the LP relaxation is the optimal solution in this case and obtains the optimal nk savings using our LP modified greedy algorithm.

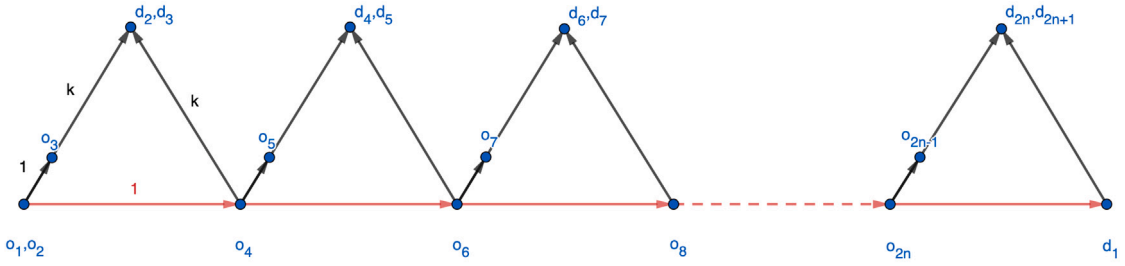

Fig. 3. The spatial graph SQ_n^* of an illustrative example.

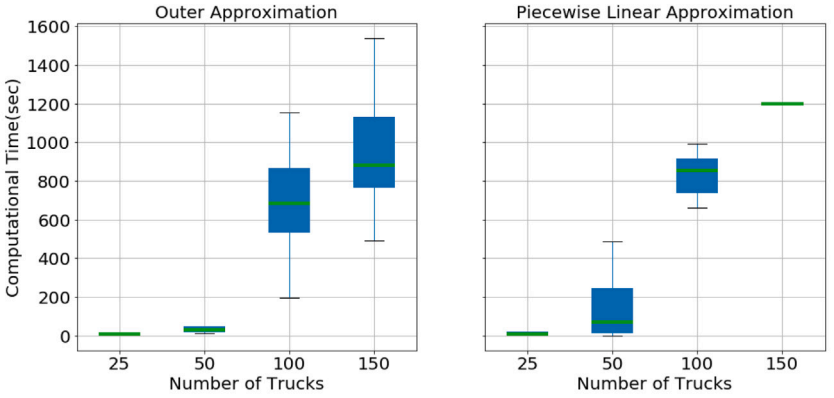

Fig. 4. OA v.s. PC-LA computational time for homogeneous fleet.

6. Numerical experiments

In this section, we present the results of a series of numerical experiments conducted to examine the performance of the proposed OA algorithm, DP-based heuristic and LP-relaxation approximation algorithm. The OA algorithm is implemented in GAMS/CPLEX platform. We use MATLAB for implementing the DP-based heuristic and the approximation algorithm. The computational tests are conducted on a 3.4 GHz Dell Optiplex 990 Pentium i7-2600 computer with 8 GB RAM on the 64-bit version of the Windows 10 operating system. This numerical section is divided into four different sections. In the first two parts, we study the computational performance of the proposed OA algorithm on a small- to medium-sized network, followed by the third section that compares all three methods in terms of the achieved energy savings while changing the flexibility of truck travel schedules. Finally the last section is dedicated to the DP heuristic where we test the algorithm in realistic settings.

6.1. Outer approximation: Homogeneous fleet

In this section we benchmark the OA algorithm against the pieces-wise linear approximation (PC-LA) proposed by Magnanti and Stratila (2004) and also highlight the performance improvement as a result of implementing the local search after solving OA-MP. Throughout this subsection, we assume the objective function that reflects the energy consumption is characterized as $f_l(x_l^1, x_l^2, \dots, x_l^K) = (\sum_{c\in C} \sum_{k\in K_c} \alpha_c x_l^k)^{(1/n)}$, where α_c is the fuel consumption coefficient for trucks of class c and n is a real number greater than 1. Function f_l by definition is a concave function. However, since $x_l^k = x_l^{k^n}$, owing to the fact that x_l^k is a binary variable, we can rewrite the function as $f_l(x_l^1, x_l^2, \dots, x_l^K) = (\sum_{c\in C} \sum_{k\in K_c} \alpha_c x_l^{k^n})^{(1/n)}$. This seemingly minor modification converts the new function to the family of norm functions, which are convex (proof provided in Lemma 3 in the appendix). We set $n = 1.06$ to replicate the results obtained via an empirical study by Zabat et al. (1995). In this experiment, we assume that the fleet is homogeneous and we have only one vehicle class with $\alpha_c = 1$. Similar to Larsson et al. (2015), we adopt the Hanan grid as the test network in this section (Hanan, 1966). In a Hanan grid, the edge weight is equal to the Euclidean distance between the nodes. We also follow the setting in Larson et al. (2016) to consider two cases for the speed of a link in the time-expanded network. The links in the time-expanded network either indicate the waiting at a physical station or indicate a movement with the uniform speed limit of the network along one of the physical links for our numerical examples. Furthermore, we consider four fleet sizes, namely 25, 50, 100 and 150.

For every fleet size we generate 20 random networks, and then compare the computational time of OA and PC-LA. Both algorithms are implemented with an optimality gap of 10%. For the former, we set 30 iterations as the maximum number of iterations

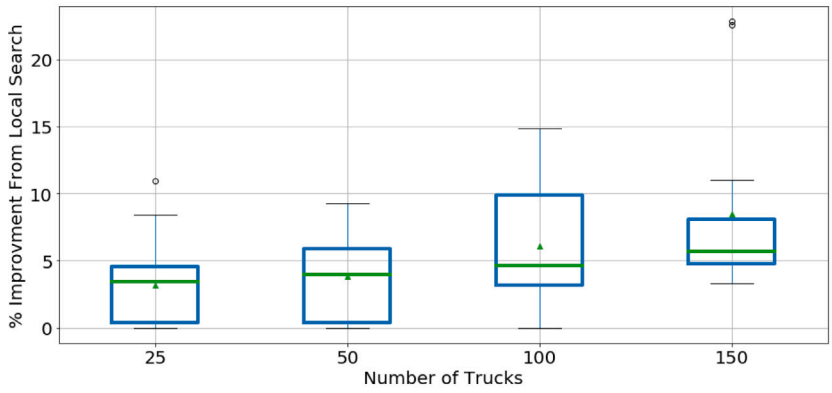

Fig. 5. Local Search Improvement.

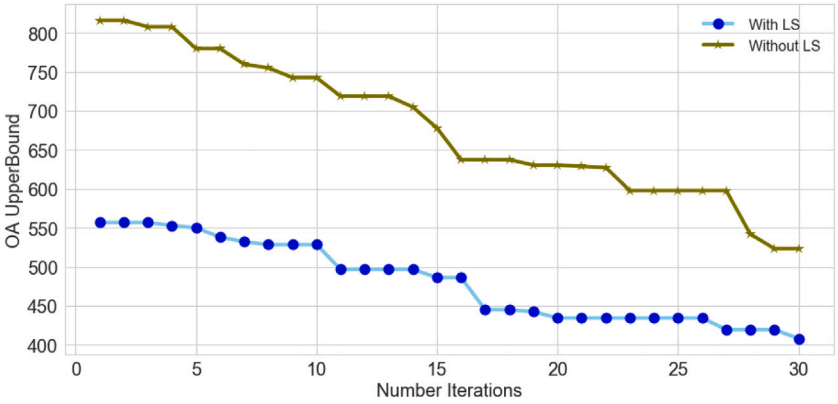

Fig. 6. OA Upper Bound for 150 trucks w/o Local Search (LS).

and the maximum CPU time of 1,200 s is set for the latter. Fig. 4 presents the time comparison between these two methods. It shows that the increase in the number of trucks would result in an increase in computational time for both methods. OA requires less time on average to achieve the desired optimally gap than PC-LA. We should point out that for the cases of the number of trucks equal to 150, the maximum CPU time of 1200 s is reached for PC-LA and therefore the solution reported does not reach the desired optimality gap of 10%.

In order to highlight the importance of implementing the local search within the OA algorithm we design a set of studies to confirm the improvements in lowering the best upper bound when the local search is applied. In particular, the upper-bound improvement is calculated according to relative difference between the OA upper bound without and with the local search. Similar to the previous tests, for every fleet size, 20 networks are randomly generated and the OA algorithm is applied with and without the local search. The results illustrated in Fig. 5 clearly show the importance of applying the local search within the OA-MP in order to improve the quality of the integer solutions. We can also conclude that the improvement in the upper-bound solution tends to increase with the increase in the number of trucks, as expected. Finally, Fig. 6 shows the trajectory of the OA upper bound for 30 iterations, where the upper bounds achieved with the local search are clearly better that the case without the local search.

6.2. *Outer approximation: Heterogeneous fleet*

The goal of this section of the numerical study is to analyze the performance of the OA algorithm while considering a heterogeneous fleet. In particular, in this section we assume that there are three classes of trucks with fuel consumption coefficients of $\alpha_1 = 1$, $\alpha_2 = 0.48$ and $\alpha_3 = 0.44$. The rest of the experiments settings are the same as those in the previous section. As we can see from the results depicted in Fig. 7 the OA algorithm has clearly outperformed in PC-LA across all of the network instances. Once again, the maximum CPU time of 1200 is reached in the latter for network size equals to 1500 and thus the desired optimality of 10% is not achieved.

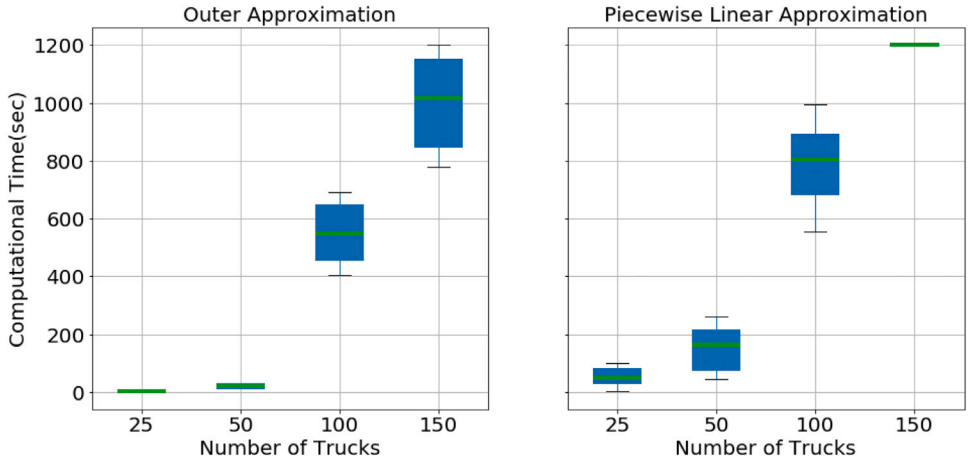

Fig. 7. OA v.s. PC-LA computational time for heterogeneous fleet.

6.3. DP-based heuristic for realistic instances

This subsection is concerned with the performance of the proposed DP-based heuristic. Specifically, we first compare the fuel savings between OA and DP algorithms, and then benchmark the results against the heuristic proposed by Larson et al. (2014) in solving the problem on the Germany highway network up to 10,000 trucks. The network consists of 647 nodes, 695 edges and 12 origins/destinations (Fig. 8). The right panel of Fig. 9 compares the percentage of fuel savings achieved by OA and DP. While the former yields more savings, the latter shows a reasonably good performance. The DP-based heuristic is very fast, which makes it appealing for application in a real-time setting. Moreover, its solution time is only slightly affected by the increase in fleet size and can be applied to solve large-scale instances, to which the OA algorithm fail to prescribe a solution. This is shown in the left panel of Fig. 9, where DP manages to solve the problem with up to 10,000 trucks and outperforms substantially the method proposed by Larson et al. (2013, 2014).

6.4. Sensitivity analysis of schedule flexibility

In this subsection, we first compare the achieved fuel savings among the DP-based heuristic, OA, and the approximation algorithm. In order to compare these three algorithms, we implement them on the setting in Section 5. Second, we examine the impact of time-flexibility of trucks' schedules on the potential fuel savings. All three algorithms are tested on the Hanan grid against two scenarios of truck schedule flexibility. In the first scenario, a time-flexible schedule allows trucks to stay at intermediate nodes while following their energy-minimizing paths between their origins and destinations. In the second scenario, trucks have a tight schedule, i.e., for each truck k the difference between their earliest departure time and latest arrival time, $T_k^{LA} - T_k^{ED}$, equals the time required to travel along the shortest path from $O(k)$ to $D(k)$.

For each scenario we generate 20 sample spatio-temporal networks and report the average performance of these three methods. It is worth pointing out that both DP and the approximation algorithm significantly outperform the OA algorithm across the tests conducted in this section in terms of computational performance, as their solution time is a fraction of a second.

From the results illustrated in Figs. 10 and 11, we can observe that in both scenarios, OA achieves the highest margins of savings, followed by the approximate algorithm and the DP-based heuristic. In the case of inflexible schedules, the difference between the achieved savings becomes small. This confirms our intuition that flexible schedules would create more platooning opportunities and thus yield more savings. In addition, in both scenarios, the approximation algorithm seems to achieve higher margins of savings compared with the DP-based heuristic. In particular, in the case of inflexible schedule, its solution is comparable with that of OA while this difference becomes larger in the case of flexible schedules.

7. Conclusion

In this paper we have investigated the scheduled platoon planning problem in a general setting. Leveraging a time-expanded network, we formulated the problem as a minimum concave-cost network flow problem, which enables us to apply outer approximation cuts, strengthened further with a local search to improve the solution quality. The local search builds on the initial solution provided by the outer approximation algorithm through introducing a series of linear programs to facilitate the process of finding a locally optimal solution. Unlike many other local search approaches, we proved that the proposed local search terminates in a polynomial number of operations in the inputs to the problem.

Although the proposed outer approximation method significantly improves on solution time and quality and enables scheduled planning for medium networks, it is not suitable for solving large-scale instances or real-time implementation. Therefore, we

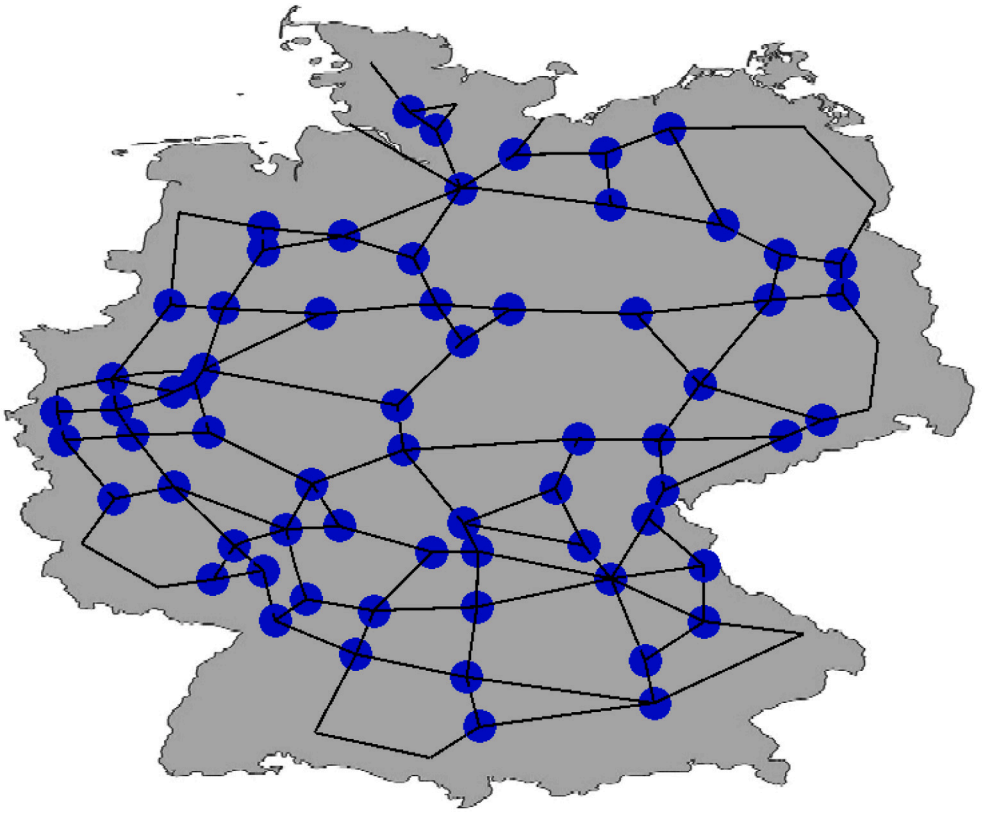

Fig. 8. Germany Network.

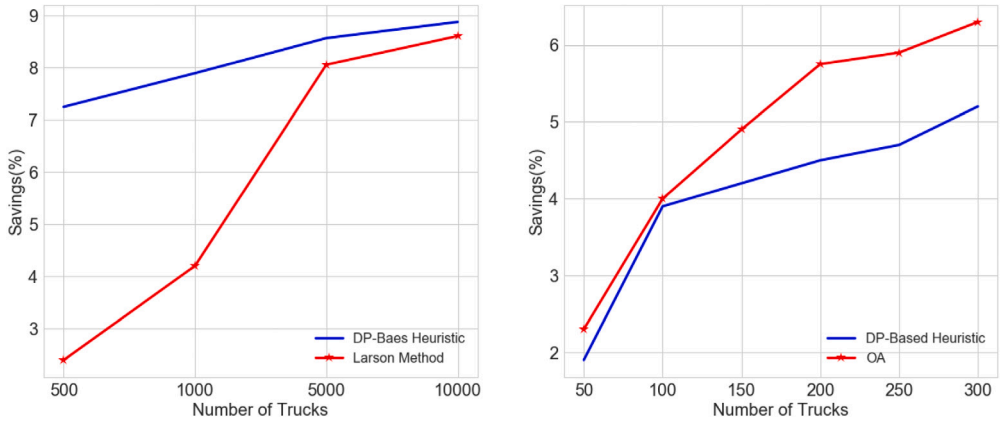

Fig. 9. DP-based Heuristic Performance.

devise a dynamic-programming-based heuristic that can provide a solution of reasonable quality in a timely fashion. Lastly, we study a simplified platoon planning problem investigated previously in the literature, which is particularly suitable for providing ballpark estimates of truck platooning benefits. We connect this problem with the well known "Minimum Weight Pairwise Distance Preservers" problem, and present an approximation algorithm that is superior to previously proposed methods in the literature. Finally, we compare the performance of all algorithms against each other as well as the methods proposed in the literature using a number of synthetic or real instances.

The framework presented in this paper provides a foundation for addressing other planning and operations issues related to truck platooning such as optimal platoon formation and behavioral platoon stability (Sun and Yin, 2019). Our future work can take into account the uncertainty arising in travel schedules as well as the transportation network.

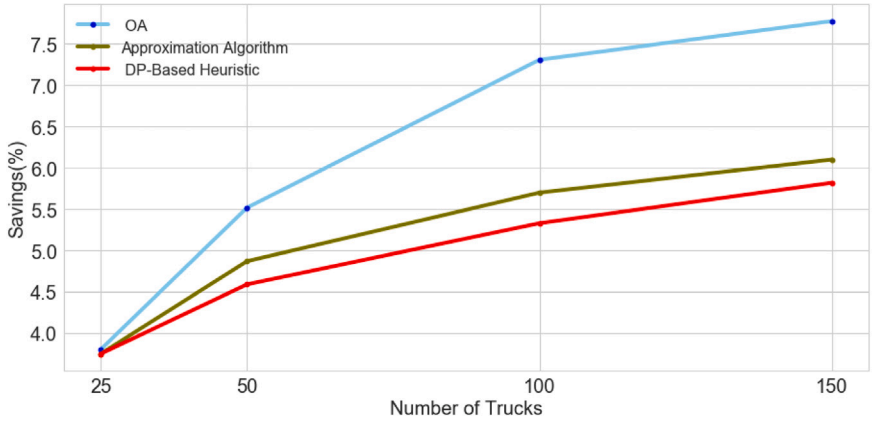

Fig. 10. Fuel Consumption Savings for Flexible Schedule.

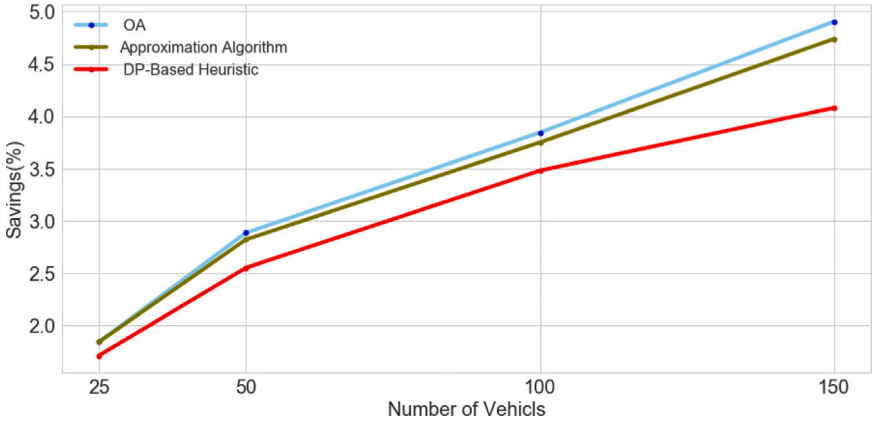

Fig. 11. Fuel Consumption Savings for Inflexible Schedule.

CRediT authorship contribution statement

Mojtaba Abdolmaleki: Methodology, Investigation, Writing – original draft. **Mehrdad Shahabi:** Methodology, Investigation, Writing – original draft. **Yafeng Yin:** Conceptualization, Methodology, Investigation, Writing – reviewing & editing, Supervision. **Neda Masoud:** Conceptualization, Writing – reviewing & editing.

Acknowledgments

The work described in this paper was partly supported by research grants from the National Science Foundation, United States (CPS-1837245, CMMI-1904575 and CMMI-2046372) and the USDOT Tier-1 University Transportation Center Freight Mobility Research Institute (FMRI).

Appendix A

Lemma 3. *Every Norm function,* $F(x) = \|x\|_p,\ \forall p \geq 1$ *is convex.*

Proof. Assume x_1 and x_2 are two points that belong to the domain of $F(x)$ and λ as a positive number between zero and one.

$$\begin{aligned} F(\lambda x_1 + (1-\lambda)x_2) &= \|\lambda x_1 + (1-\lambda)x_2\| \\ &\leq \|\lambda x_1\| + \|(1-\lambda)x_2\| \text{ (from triangular inequality)} \\ &= \lambda\|x_1\| + (1-\lambda)\|x_2\| \end{aligned}$$

which is the definition of a convex function and the proof is complete. □

Appendix B

Proof of Theorem 1. To prove the first part of Theorem 1, we denote by $\vec{x}$ and $\vec{y}$ two consecutive solutions to the optimization problem described in (7). If the gradient of the objective function z at both $\vec{x}$ and $\vec{y}$ are equal. Then, both optimization problems will result in the same solution $\vec{y}$, and this concludes the proof. As such, we can assume $\nabla z(\vec{x}) \neq \nabla z(\vec{y})$. In this case, as the flow in each link of the graph G should be less than $|P|$, we conclude:

$$z(x) = \sum_{l \in L} f_l(Y_l) \leq \sum_{l \in L} |P| max_c(f_l(e_l^c)) \tag{14}$$

where e_l^c is the vector with unit vector with number of trucks y_l^c equals 1 for class c and zero otherwise.

Therefore, it is sufficient to prove that if we can improve the objective function by solving the optimization problem (7), we can improve it by at least γ for some constant $\gamma > 0$.

To complete the proof note that the inner product of $\vec{y} - \vec{x}$ and the difference in the gradient of z at $\vec{y}$ and $\vec{x}$ can be written as:

$$(\nabla z(\vec{y}_l) - \nabla z(\vec{x}_l))(\vec{y}_l - \vec{x}_l) = \sum_l [\nabla f_l(\vec{x}) - \nabla f_l(\vec{y})].[\vec{y}_l - \vec{x}_l]$$

Here, we use $\vec{x}_l$ and $\vec{y}_l$ to denote the vector Y_l in optimization problem (1) at solutions $\vec{x}$ and $\vec{y}$, respectively. The concavity of the functions f_l yields:

$$[\nabla f_l(\vec{x}) - \nabla f_l(\vec{y})].[\vec{y}_l - \vec{x}_l] \leq 0$$

As we have $\nabla z(\vec{x}) \neq \nabla z(\vec{y})$, there should exist a link l such that $\vec{y}_l \neq \vec{x}_l$. Strong concavity of functions f_l yields:

$$f_l(\vec{y}).[\vec{y}_l - \vec{x}_l] \leq -\gamma \|\vec{y}_l - \vec{x}_l\|^2 \leq -\gamma$$

where the last inequality follows from the fact that both $\vec{y}_l$ and $\vec{x}_l$ are integral points. As a result, we have:

$$(\nabla z(\vec{y}_l) - \nabla z(\vec{x}_l))(\vec{y}_l - \vec{x}_l) \leq -\gamma \tag{15}$$

On the other hand, the concavity of the objective function in optimization problem (7) yields:

$$\nabla z(\vec{x})^T.(\vec{y} - \vec{x}) \leq 0 \tag{16}$$

Now we can combine (15) and (16) to obtain:

$$\nabla z(\vec{y})^T.(\vec{y} - \vec{x}) < -\gamma$$

We prove the second part of Theorem 1 in three separate steps. First, we introduce a class of functions as well-bent functions. Then, we prove the statement of the theorem is true for the class of well-bent objective functions. Finally, we prove every concave objective function can be approximated by a function in the class of well-bent functions.

Let us fix an $\epsilon > 0$ to take any arbitrary value greater than 0.

Definition 3. We call a function well-bent if there exists a $\theta > 0$ that satisfies the following condition for any $n \in \mathbb{N}$:

$$f'(n+1) - f'(n) > \frac{\theta}{n^{1+\epsilon}} \quad or \quad f'(n+1) - f'(n) = 0 \tag{17}$$

Lemma 4. *Theorem 1 holds if the energy consumption for each link, $f_l(.), \forall l \in E(G)$, is in the class of well-bent functions.*

Proof. Let us denote by $\vec{x}$ and $\vec{y}$ two consecutive solutions to the optimization problem described in (7). If the gradient of the objective function z at both $\vec{x}$ and $\vec{y}$ are equal. Then, both optimization problems will result in the same solution $\vec{y}$, and this concludes the proof. As such, we can assume $\nabla z(\vec{x}) \neq \nabla z(\vec{y})$. In this case, we prove that if we can improve the objective function by solving the optimization problem (7), we can improve it by at least $O(\frac{1}{P^{4+\epsilon}|E(G)|^2})$, i.e., $z(\vec{x}) - z(\vec{y}) > \frac{\kappa}{P^{4+\epsilon}|E(G)|^2}$, for some constant $\kappa > 0$. This result together with (14) from the proof for the first part of Theorem 1 will conclude the proof. The concavity of the objective function in optimization problem (7) yields:

$$\nabla z(\vec{x})^T.(\vec{y} - \vec{x}) < 0 \tag{18}$$

Moreover, in the single-class case the total energy consumption can be separated over links as follows:

$$z(\vec{x}) = \sum_l f_l(\sum_k x_l^k)$$

As such, we can write the difference in the gradient of z at $\vec{x}$ and $\vec{y}$ as:

$$\nabla z(\vec{x}) - \nabla z(\vec{y}) = (f'_{l_1}(\sum_k x_{l_1}^k) - f'_{l_1}(\sum_k y_{l_1}^k), \ldots, f'_{l_m}(\sum_k x_{l_m}^k) - f'_{l_m}(\sum_k y_{l_m}^k))$$

Taking the inner product with the vector $\vec{x}-\vec{y}$ yields:

$$(\nabla z(\vec{x})-\nabla z(\vec{y}))^T(\vec{x}-\vec{y})=\sum_l[f_l'(\sum_k x_l^k)-f_l'(\sum_k y_l^k)][\sum_k x_l^k-\sum_k y_l^k] \tag{19}$$

As the functions $f_l(.)$ are concave we can conclude:

$$[f_l'(\sum_k x_l^k)-f_l'(\sum_k y_l^k)][\sum_k x_l^k-\sum_k y_l^k]\le 0 \quad \forall l\in E(G) \tag{20}$$

As we have $\nabla z(\vec{x})\neq\nabla z(\vec{y})$, there should be a link $l_i\in E(G)$ with $f'(\sum_k x_{l_i}^k)\neq f'(\sum_k y_{l_i}^k)$. The function $f_{l_i}(.)$ being concave and well-bent yields:

$$[f_{l_i}'(\sum_k x_{l_i}^k)-f_{l_i}'(\sum_k y_{l_i}^k)][\sum_k x_{l_i}^k-\sum_k y_{l_i}^k]<\frac{-\theta}{P^{1+\epsilon}} \tag{21}$$

From (19), (20), and (21) we have:

$$(\nabla z(\vec{y})-\nabla z(\vec{x}))^T(\vec{y}-\vec{x})\le\frac{-\theta}{P^{1+\epsilon}} \tag{22}$$

Summing (18) and (22) yields:

$$\nabla z(\vec{y})^T(\vec{y}-\vec{x})\le\frac{-\theta}{P^{1+\epsilon}} \tag{23}$$

Now, let us denote by η^* the solution to:

$$argmax_{0\le\eta\le1}\left\{\eta|\nabla z(\eta\vec{x}+(1-\eta)\vec{y})(\vec{y}-\vec{x})\le\frac{-\theta}{2P^{1+\epsilon}}\right\}$$

Note that η^* is well defined, since the $\eta=0$ would satisfy the condition stated above. Doing so, we can consider two cases:

- $\eta^*=1$: As such, $\nabla z(\vec{x})(\vec{y}-\vec{x})\le\frac{-\theta}{2P^{1+\epsilon}}$. In this case, we can overestimate the function f at point $\vec{y}$ by the first-order Taylor approximation of z at point $\vec{x}$:

$$z(\vec{y})\le z(\vec{x})+\nabla z(\vec{x})(\vec{y}-\vec{x}) \tag{24}$$

  However, as $\eta^*=1$, we have:

$$z(\vec{x})+\nabla z(\vec{x})(\vec{y}-\vec{x})\le z(\vec{x})-\frac{\theta}{2P^{1+\epsilon}} \tag{25}$$

  which concludes the proof in this case.
- $\eta^*<1$: As the gradient of the function z is continuous, using the intermediate value theorem we have:

$$\nabla z(\eta^*\vec{x}+(1-\eta^*)\vec{y})(\vec{y}-\vec{x})=\frac{-\theta}{2P^{1+\epsilon}} \tag{26}$$

  Combining (26) with (23) yields:

$$(\nabla z(y)-\nabla z(\eta^*\vec{x}+(1-\eta^*)\vec{y}))(\vec{y}-\vec{x})\le\frac{-\theta}{2P^{1+\epsilon}}$$

  Using Cauchy–Schwarz inequality we can derive:

$$\left\|\nabla z(\vec{y})-\nabla z(\eta^*\vec{x}+(1-\eta^*)\vec{y})\right\|_2\left\|\vec{y}-\vec{x}\right\|_2\ge\frac{\theta}{2P^{1+\epsilon}}$$

  Moreover, the maximum euclidean distance between any two basic feasible solutions in the optimization problem (7) is at most $P|E(G)|$, i.e., $\left\|\vec{y}-\vec{x}\right\|_2\le P|E(G)|$. As such, we have:

$$\left\|\nabla z(y)-\nabla z(\eta^*\vec{x}+(1-\eta^*)\vec{y})\right\|_2\ge\frac{\theta}{2P^{1+\epsilon}P|E(G)|} \tag{27}$$

  That being said, the function z has a finite Hessian, so there exists an M such that $\left\|\nabla^2 f(x)\right\|_2\le M$. It is well known that for the concave function z we have $\|\nabla f(x)-\nabla f(y)\|_2\le M\|x-y\|_2$, using this fact together with (27) we can conclude:

$$\left\|\vec{y}-(\eta^*\vec{x}+(1-\eta^*)\vec{y})\right\|_2\ge\frac{\theta}{2MP^{1+\epsilon}P|E(G)|}$$

  Using the upper bound on the maximum distance of two basic feasible solutions we get:

$$|\eta^*|\ge\frac{\theta}{2MP^{1+\epsilon}(P|E(G)|)^2} \tag{28}$$

  We can underestimate the function z at the point y by the first-order Taylor approximation from the value of the function at point $\eta^*\vec{x}+(1-\eta^*)\vec{y}$:

$$z(\vec{y})\le z(\eta^*\vec{x}+(1-\eta^*)\vec{y})+\nabla z(\eta^*\vec{x}+(1-\eta^*)\vec{y})(y-(\eta^*\vec{x}+(1-\eta^*)\vec{y}))$$

  We can rewrite the expression in right-hand side to get:

$$\le z(\eta^*\vec{x}+(1-\eta^*)\vec{y})+\nabla z(\eta^*\vec{x}+(1-\eta^*)\vec{y})(\vec{y}-\vec{x})\eta^*$$

Using (26) and (28) we can simplify further to get:

$$\leq z(\eta^*\vec{x} + (1-\eta^*)\vec{y}) + \frac{-\theta}{2P^{1+\epsilon}} \frac{\theta}{2M P^{1+\epsilon}(P|E(G)|)^2} \tag{29}$$

From (18) we have $z(x) > z(\eta^*\vec{x} + (1-\eta^*)\vec{y})$. Combining this fact with the inequality in (29) concludes the proof. □

Lemma 5. *Any concave function f_l can be approximated by a well-bent function.*

Proof. Given function f_l, we fix a $\zeta > 0$ and define function g_l as a well-bent function that approximates function f_l based on the solution to the differential equation below:

$$\begin{cases} g_l(0) = f_l(0) \\ g_l'(0) = f_l'(0) \end{cases} \tag{30a}$$

$$g_l'(x) = \begin{cases} f_l'(x) & \text{If } f_l'(x) - g_l'(x-1) < \frac{\zeta}{x^{1+\epsilon}} \\ g_l'(x-1) & \text{If } f_l'(x) - g_l'(x-1) \geq \frac{\zeta}{x^{1+\epsilon}} \end{cases} \qquad \forall x \in \mathbb{N} \tag{30b}$$

$$g_l'(x) = (\lceil x \rceil - x) g_l'(\lfloor x \rfloor) + (x - \lfloor x \rfloor) g_l'(\lceil x \rceil) \qquad \forall x \notin \mathbb{N} \tag{30c}$$

By Picard's existence and uniqueness theorem for differential equations for a given P there should exist a unique solution to the system of differential Eq. (30). However, if we define the difference function $h(x) = f_l(x) - g_l(x)$ we have:

$$\begin{cases} h(0) = 0 \\ h'(0) = 0 \\ |h'(x)| < \max\left\{\frac{\zeta}{x^{1+\epsilon}}, \zeta\right\} \end{cases} \tag{31a}$$

Using the fundamental theorem of calculus we have:

$$h(a) = \int_0^a h'(x)dx = \int_0^1 h'(x)dx + \int_1^a h'(x)dx < \zeta + \int_1^a \frac{\zeta}{x^{1+\epsilon}}dx < \zeta(1 + \frac{1}{\epsilon}) \tag{32}$$

If we choose $\zeta = \frac{\delta}{1+\frac{1}{\epsilon}} f_l(1)$, we can approximate the function f_l within a factor of $1-\delta$ by the function g_l. i.e., for any arbitrary a we have $|f_l(a) - g_l(a)| \leq (1-\delta) f_l(x)$. Which concludes the proof. □

Using Lemma 5, we can solve the optimization problem (1) substituting the functions f_l with the functions g_l. Then, we can claim that the objective value at the solution $\vec{x^{**}}$, when f_l is replaced with g_l, is within $1-\delta$ factor of the objective value for the solution to the original optimization problem $\vec{x}$.

Appendix C

There are two main approaches to dealing with a Pairwise Distance Preservers problem in the literature. The first is to use the LP relaxation approach that we used in this paper; the second is to find a good decomposition structure that not only can decompose the original network of these structures, but more importantly, is able to find decomposition blocks with extreme properties in an acceptable polynomial time. One of the handy decomposition structures used in many Steiner-type problems, especially a distance preservers problem, is the family of junction tree algorithms: trees with an in and out branching. This appendix presents an example to show that there exists a graph that does not contain any junction tree with the savings at least $\frac{1}{k^2}$ of the total savings as a subgraph. This suggests why junction trees might not be a good decomposition structure for the savings perspective of the Pairwise Distance Preservers problem.

Definition 4. A junction tree is defined as a collection of paths, all passing through the same node v.

Definition 5. We define a twisting path TP_n inductively to be a union of n simple directed paths. While TP_1 is the simple path of length 1, we can obtain TP_n from TP_{n-1} by adding a new path whose overlap with all the other paths is exactly 1. Moreover, the overlap of the new path with any other path is prior to all the other overlaps on those paths if we consider the directed ordering of edges.

Fig. 12 demonstrates TP_3. If we consider the paths that generate TP_n as the trips in a network its easy to see that all junction trees in this network are the union of two paths. As such, the highest weight for junction trees in TP_n is 1. However, the total number of overlaps equals $\binom{n}{2}$. While, as described in Bodwin and Williams (2016), the junction tree approach or any other decomposition approach requires junction trees with higher weight compared to the total savings.

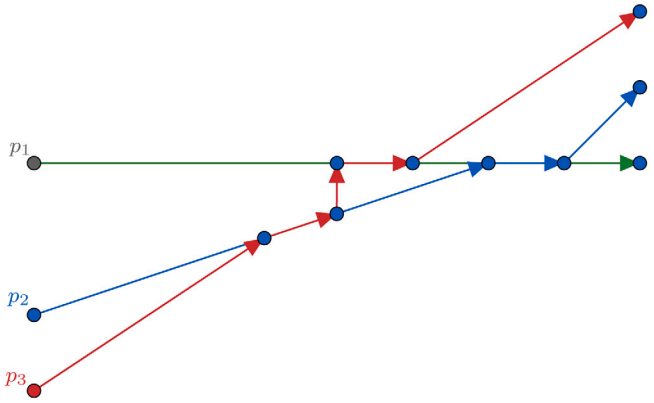

Fig. 12. The spatial graph TP_3.

References

Ahuja, Ravindra K., Magnanti, Thomas L., Orlin, James B., 1988. Network Flows. Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts.

Bhoopalam, Anirudh Kishore, Agatz, Niels, Zuidwijk, Rob, 2018. Planning of truck platoons: A literature review and directions for future research. Transp. Res. B 107, 212–228.

Bodwin, Greg, Williams, Virginia Vassilevska, 2016. Better distance preservers and additive spanners. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, pp. 855–872.

Bonnet, Christophe, Fritz, Hans, 2000. Fuel Consumption Reduction in a Platoon: Experimental Results with Two Electronically Coupled Trucks at Close Spacing. Technical Report, SAE Technical Paper.

Boysen, Nils, Briskorn, Dirk, Schwerdfeger, Stefan, 2018. The identical-path truck platooning problem. Transp. Res. B 109, 26–39.

Browand, Fred, McArthur, John, Radovich, Charles, 2004. Fuel saving achieved in the field test of two tandem trucks.

Chlamtáč, Eden, Dinitz, Michael, Kortsarz, Guy, Laekhanukit, Bundit, 2017. Approximating spanners and directed steiner forest: Upper and lower bounds. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, pp. 534–553.

Coppersmith, Don, Elkin, Michael, 2006. Sparse sourcewise and pairwise distance preservers. SIAM J. Discrete Math. 20 (2), 463–501.

Davila, Arturo, 2013. Report on fuel consumption. SARTRE, Deliv.

Duret, Aurelien, Wang, Meng, Ladino, Andres, 2020. A hierarchical approach for splitting truck platoons near network discontinuities. Transp. Res. B 132, 285–302.

Elkin, Michael, Peleg, David, 2000. The hardness of approximating spanner problems. In: Annual Symposium on Theoretical Aspects of Computer Science. Springer, pp. 370–381.

Elkin, Michael, Peleg, David, 2007. The hardness of approximating spanner problems. Theory Comput. Syst. 41 (4), 691–729.

Fletcher, Roger, Leyffer, Sven, 1994. Solving mixed integer nonlinear programs by outer approximation. Math. Program. 66 (1–3), 327–349.

Hanan, Maurice, 1966. On steiner's problem with rectilinear distance. SIAM J. Appl. Math. 14 (2), 255–265.

Larson, Jeffrey, Kammer, Christoph, Liang, Kuo-Yun, Johansson, Karl Henrik, 2013. Coordinated route optimization for heavy-duty vehicle platoons. In: Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on. IEEE, pp. 1196–1202.

Larson, Jeffrey, Liang, Kuo-Yun, Johansson, Karl H., 2014. A distributed framework for coordinated heavy-duty vehicle platooning. IEEE Trans. Intell. Transp. Syst. 16 (1), 419–429.

Larson, Jeffrey, Munson, Todd, Sokolov, Vadim, 2016. Coordinated platoon routing in a metropolitan network. In: 2016 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing. SIAM, pp. 73–82.

Larsson, Erik, Sennton, Gustav, Larson, Jeffrey, 2015. The vehicle platooning problem: Computational complexity and heuristics. Transp. Res. C 60, 258–277.

Lozovanu, D.D., 1982. Properties of optimal-solutions of a grid transport problem with concave cost function of the flows on the arcs. Engineering Cybernetics 20 (6), 34–38.

Magnanti, Thomas L., Stratila, Dan, 2004. Separable concave optimization approximately equals piecewise linear optimization. In: International Conference on Integer Programming and Combinatorial Optimization. Springer, pp. 234–243.

Masoud, Neda, Jayakrishnan, R., 2017. A decomposition algorithm to solve the multi-hop Peer-to-Peer ride-matching problem. Transp. Res. B 99, 1–29.

Muratori, Matteo, Holden, Jacob, Lammert, Michael, Duran, Adam, Young, Stanley, Gonder, Jeffrey, 2017. Potentials for Platooning in US Highway Freight Transport. Jeffrey Technical Report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Murray, Walter, Ng, Kien-Ming, 2010. An algorithm for nonlinear optimization problems with binary variables. Comput. Optim. Appl. 47 (2), 257–288.

Nourmohammadzadeh, Abtin, Hartmann, Sven, 2016. The fuel-efficient platooning of heavy duty vehicles by mathematical programming and genetic algorithm. In: International Conference on Theory and Practice of Natural Computing. Springer, pp. 46–57.

Shida, M., Doi, T., Nemoto, Y., Tadakuma, K., 2010. A short-distance vehicle platooning system: 2nd report, evaluation of fuel savings by the developed cooperative control. In: Proceedings of the 10th International Symposium on Advanced Vehicle Control. AVEC. pp. 719–723.

Sokolov, Vadim, Larson, Jeffrey, Munson, Todd, Auld, Josh, Karbowski, Dominik, 2017. Maximization of platoon formation through centralized routing and departure time coordination. Transp. Res. Rec.: J. Transp. Res. Board (2667), 10–16.

Sun, Xiaotong, Yin, Yafeng, 2019. Behaviorally stable vehicle platooning for energy savings. Transp. Res. C 99, 37–52.

van Wyk, Franco, Wang, Yiyang, Khojandi, Anahita, Masoud, Neda, 2019. Real-time sensor anomaly detection and identification in automated vehicles. IEEE Trans. Intell. Transp. Syst. 21 (3), 1264–1276.

Zabat, Michael, Stabile, Nick, Farascaroli, Stefano, Browand, Frederick, 1995. The aerodynamic performance of platoons: A final report.