The Vehicle Platooning Problem: Computational Complexity and Heuristics

Erik Larsson and Gustav Sennton

KTH Royal Institute of Technology, Automatic Control Department. Stockholm, Sweden

Jeffrey Larson*

Argonne National Laboratory, Mathematics and Computer Science Division. Argonne, IL, USA

Abstract

We create a mathematical framework for modeling trucks traveling in road networks, and we define a routing problem called the platooning problem. We prove that this problem is NP-hard, even when the graph used to represent the road network is planar. We present integer linear programming formulations for instances of the platooning problem where deadlines are discarded, which we call the unlimited platooning problem. These allow us to calculate fuel-optimal solutions to the platooning problem for large-scale, real-world examples. The problems solved are orders of magnitude larger than problems previously solved exactly in the literature. We present several heuristics and compare their performance with the optimal solutions on the German Autobahn road network. The proposed heuristics find optimal or near-optimal solutions in most of the problem instances considered, especially when a final local search is applied. Assuming a fuel reduction factor of 10% from platooning, we find fuel savings from platooning of 1-2% for as few as 10 trucks in the road network; the percentage of savings increases with the number of trucks. If all trucks start at the same point, savings of up to 9% are obtained for only 200 trucks.

Keywords: Vehicle Platooning, Computational Complexity, Vehicle Routing

*Corresponding author
Email address: jmlarson@anl.gov (Jeffrey Larson)

1. Introduction

Companies have significant economic and environmental incentives for reducing the fuel consumption of heavy-duty vehicles (HDVs). Since fuel costs represent a third of the total operational costs of an HDV (Schittler, 2003), even small advances in fuel efficiency will noticeably increase profits for many organizations. Because vehicles account for a large percentage of total carbon emissions—20% according to Schroten et al. (2012), a quarter of which comes from HDVs—reductions in HDV fuel usage can yield substantial progress toward achieving carbon reduction goals. For example, the European Commission (2011) has stated goals of decreasing carbon emissions by 60% by 2050; such ambitious goals can be achieved only by a multifaceted approach.

In addition to ongoing research into engine efficiency and aerodynamic vehicle design, a supplementary method for reducing fuel use is to form vehicle platoons. By driving vehicles in a single lane in close proximity, as can be seen in Figure 1, fuel reductions of up to 20% are possible for the nonleading vehicles; (Robinson et al., 2010; Bonnet and Fritz, 2000).

Such platooning is profitable, however, only under certain circumstances. The reduction in fuel use depends on the distance between the trucks in a platoon and on the speed of the platoon. Bonnet and Fritz (2000) show trailing HDVs traveling at 80 km/h experience a 21% fuel reduction when the distance between the vehicles is 10 m, while the fuel reduction is 16% for an intervehicle distance of 16 m. The fuel reductions for the same vehicles and distances traveling at 60 km/h are approximately 16% and 10%, respectively. Naturally, safety considerations must be addressed when driving HDVs at such close distances; see Tatchikou et al. (2005) and Taleb et al. (2010). Demonstrating that platoons can operate safely in a variety of settings must be shown before they can be adopted on public roadways.

Some platooning paradigms, such as PATH (Browand et al., 2004) or Dolphin (Tsugawa et al., 2000), assume the existence of roadside systems to facilitate intervehicle communications. We assume the vehicles themselves are equipped with the necessary technologies (e.g., LIDAR, WiFi) for platoon formation and maintenance. Such technologies are increasing found on new HDVs; see, for example, Shladover (2007).

In addition to safety and technological concerns, excessive traffic can greatly reduce platooning benefits, since low-speed platooning would provide almost no reduction in aerodynamic drag. Since vehicles will likely not be platooning through large urban centers, we assume throughout that the time required to travel a road is fixed independent of time, as is the case for large portions of the U.S. Interstate Highway System. We consider this case since these long stretches of low-traffic road are likely where platooning will provide the most fuel-saving benefits. Routing individual vehicles (and platoons) through a time-varying network is an active area of research; see Lecluyse et al. (2009).

Most research in the vehicle platooning literature concerns the maintenance and safe maneuvering of an existing platoon of vehicles; see Kavathekar and Chen (2011). Little attention has been paid to optimally coordinating the formation and dissolution of platoons to minimize total fuel use as many vehicles move throughout a road network. The few articles that propose methods for increasing platooning opportunities acknowledge the difficulty of finding the exact routing that minimizes fuel use (Baskar et al., 2013; Larson et al., 2013), but none formally address the problem's computational complexity.

Figure 1: Three heavy-duty vehicles platooning to collectively reduce fuel consumption.

In this paper we attempt to maximize the amount of fuel saved by vehicles capable of platooning on a road network. If platooning opportunities are present, the routes may differ slightly from the obvious shortest path routes, in order to maximize fuel savings. We formally define the platooning problem, a vehicle routing problem concerned with minimizing the fuel consumption by platooning trucks given a collection of starting points, destinations, and deadlines. We do not consider the effects of traffic on the fuel consumption. The motivation behind this approach is our desire to isolate the problem and regard the computational complexity of a pure deterministic problem. For vehicle routings that also address the effects of traffic congestion but do not consider platooning, see Franceschetti et al. (2013).

We show that this platooning problem is NP-hard, even for simple cases when all trucks start at the same point and time, and it is therefore infeasible to solve anything but small instances exactly, unless P = NP. To find solutions for small instances of the platooning problem, we formulate it as an integer linear program (ILP), which can be solved by using existing ILP solvers. We present and compare two heuristics and a local improvement algorithm for solving common instances of the platooning problem. We show that the heuristics by themselves often produce decent, but not excellent, solutions. These solutions can be greatly improved by using the local improvement algorithm, resulting in solutions close to the optimum in many cases.

In addition to the general case of the platooning problem, we specifically address the case where every HDV starts at the same node. We solve very large instances of the same-start platooning problem on actual road networks, often yielding significant savings over every truck taking its shortest path to its destination. Instances of such a problem arise in a variety of real-world situations, for example, a distribution center where many HDVs leave simultaneously for various destinations, or at major junctions throughout a road network. Fuel optimal routes utilizing platoons can be calculated for trucks approaching an intersection in the road network, as in the paradigm presented in Larson et al. (2013, 2015), or for trucks stopping at common locations such as weigh stations, fuel stations, or customs checkpoints. One can view HDVs approaching a common destination as the inverse of the same-start platooning problem and can therefore solve this case by similar measures. For these reasons, the same-start platooning problem receives special attention throughout this paper. We note that our methods can solve the same-start platooning problem more efficiently than the general problem.

We emphasize that we can find solutions to large-scale, real-world instances of the platooning problem. We consistently produce fuel-optimal solutions for instances of the platooning problem on the German Autobahn road system with hundreds of HDVs. We are unaware of any other platooning formulation that can find optimal solutions for any nontrivial instance of the vehicle platooning problem for more than 5 vehicles.

The structure of the paper is as follows. In Section 2, we create a mathematical framework for the platooning problem and use this framework to prove a number of theorems regarding optimal platoon routings. Readers that are not interested in the proofs of the following sections can skim through or skip large parts of this section. In Section 3 the computational complexities of different versions of the problem are considered. After establishing that the problem is NP-complete, a conversion of the platooning problem into an ILP is considered in Section 4. Because of the established computational complexity, we attempt to solve the problem heuristically in Section 5. In Section 6 we provide comparisons of the performance of the different solvers presented in the article. Section 7 concludes the paper.

2. Background

This section contains a number of definitions that create a framework for the modeling of trucks traveling between different locations in a road network. We want to minimize the total fuel consumption, using the fact that a truck traveling behind another truck in a platoon uses only a fraction η of its normal rate.

We model an arbitrary road network using a finite, connected, directed graph $G = (V, E)$; each road in the network is denoted by an edge $e \in E$, and intersections of the network are represented by vertices $u \in V$. Each edge e has a non-negative integer length $w(e)$ associated with it. Similar to the claims of Ahuja et al. (1993), we consider only integer lengths for edges in E. Furthermore, we assume that for each edge $(i, j) \in E$ the edge $(j, i) \in E$. In the graph G, trucks are allowed to travel at a set of different speeds, H, which are represented as positive integers. The cost of traversing an edge e alone, or leading a platoon, with a certain speed v is given by $c(e, v) = w(e) \cdot f(v) > 0$, where $f(v) > 0$ is the fuel cost per unit distance. The calculation of $f(v)$ should take into consideration known properties of a section of road, for example, its grade (slope). Note that a single road does not necessarily correspond to a single edge; long edges can be (and are) subdivided by adding vertices.

2.1. Definitions

We now define many of the terms and variables used throughout the paper. For ease of reference, Table A.1 and Table A.2 in Appendix A contain the most important definitions and notations.

Definition 1. An *edge traversal* T is an ordered tuple

$$T = (e, t, v) \in E \times \mathbb{Z} \times H$$

describing the traversal of an edge e beginning at time t, traveling at a speed v.

Note 1. *For an edge traversal* $T = (e, t, v)$ *and the fuel cost function* c*, it is sometimes convenient to write* $c(T)$ *instead of* $c(e, v)$ *since the fuel cost of an edge traversal is time independent, as explained earlier.*

Definition 2. A *truck path* P starting at $u \in V$ and ending at $u' \in V$ is a sequence of edge traversals

$$P = \{(e_i, t_i, v_i)\}_{i=1}^{k},$$

where $\{e_i\}_{i=1}^k \subset E$ is a path in the graph G starting at u and ending at u', $\{v_i\}_{i=1}^k \subset H$ is a sequence of speeds, and $\{t_i\}_{i=1}^k \subset \mathbb{Z}$ is an increasing sequence of times satisfying $t_1 \geq 0$ and

$$t_{i+1} \geq t_i + \frac{w(e_i)}{v_i}.$$

The *start time* is defined as t_1, and the *finish time* is defined as $t_k + \frac{w(e_k)}{v_k}$.

Note 2. *If for some i in the truck path P,*

$$\Delta t_i = t_{i+1} - t_i + \frac{w(e_i)}{v_i} > 0,$$

this corresponds to a truck waiting at a certain node during a waiting time *of Δt_i.*

Definition 3. A *truck mission* is an ordered tuple

$$M = (s, d, \tau) \in V \times V \times \mathbb{Z}_+,$$

where $s \neq d$, representing the starting point s, the destination d, and the deadline τ of a truck.

Definition 4. Given a list of truck missions

$$[(s_1, d_1, \tau_1), ..., (s_N, d_N, \tau_N)], \; s_n \in V, d_n \in V, \tau_n \in \mathbb{Z}_+,$$

and a set of allowed speeds H, a *platoon routing* S is a list of truck paths

$$S = [P_1, ..., P_N],$$

where path P_n starts at s_n, ends at d_n, and has a finish time earlier than τ_n.

For a platoon routing S, we define $N_S(T)$ as the number of different truck paths in S containing the edge traversal T. $N_S(T)$ is called the *platoon size* of T or the number of trucks in a platoon on T.

Definition 5. The *fuel cost* of a platoon routing S is

$$C(S) = \sum_{T : N_S(T) > 0} c(T) \cdot (1 + \eta(N_S(T) - 1)),$$

where η is a platooning cost factor $0 < \eta < 1$.

For a fixed input, a platoon routing S is said to be optimal if no other platoon routing yields a smaller fuel cost.

Note 3. *The fuel cost for any nontrivial platoon in* S *(i.e., including an edge traversal* T *with platoon size greater than 1) will be less than the sum of the costs for the individual trucks within the platoon. A truck traveling behind another truck in a platoon has a fuel cost of only* $c(T) \cdot \eta$*, owing to a reduction in air resistance. Therefore* $N_S(T) - 1$ *trucks receive the reduced fuel cost while the leading truck consumes the full amount.*

Definition 6. The *platooning problem* consists of finding the optimal platoon routing for a finite list of truck missions on a graph G. If G is a planar graph, the problem is called the *planar platooning problem.*

The *unlimited platooning problem* is a special case of the platooning problem where the deadlines $\tau_n = \infty$ for $n = 1, ..., N$, and $H = \{v\}$.

The *decision version of the platooning problem* consists of deciding whether, given a list of truck missions on a graph G and an integer K, it is possible to find a platoon routing with cost less than or equal to K.

Note 4. *Given a platoon routing* S *for an instance of the platooning problem, the fuel cost calculation can be performed in polynomial time. Consequently, the decision version of the platooning problem is NP-complete if and only if the platooning problem is NP-hard.*

From here on, this article will be concerned mainly with the unlimited platooning problem. Even though $\tau_n = \infty, \forall n$, each valid platoon routing must end in the respective destination point d_n. This prevents HDVs from stalling indefinitely at a node to avoid consuming fuel.

2.2. Basic Results

We can now use these definitions to prove properties about solutions to the platooning problem.

Definition 7. A *cycle* in a truck path P is a nonempty contiguous *subpath* of P, namely, a subsequence of P, where the first and last vertex are the same.

Theorem 2.1. *In an optimal platoon routing for the platooning problem, no truck path will contain a cycle; in other words, no HDV will return to a node that it already visited.*

Proof. Suppose there is an optimal platoon routing S in which a truck path P contains a cycle O starting and ending at $u \in V$. We create a new platoon routing S' by letting the HDV in question wait at u instead of traversing the cycle, thereby removing O from P. Since edge traversals are removed in S',

$$N_{S'}(T) < N_S(T).$$

for each $T \in O$.

In a platoon routing S,

$$\begin{aligned} C(S) &= \sum_{T:N_S(T)>0} c(T) \cdot \eta \cdot N_S(T) + c(T)(1-\eta) \\ &> \sum_{T:N_{S'}(T)>0} c(T) \cdot \eta \cdot N'_S(T) + c(T)(1-\eta) = C(S'), \end{aligned}$$

since $c(T) \cdot \eta > 0$. $C(S') < C(S)$ contradicts the optimality of S and therefore no truck returns to an earlier visited node in an optimal platoon routing. □

Definition 8. The *fuel cost* of a truck path $P = \{T_i\}$, is defined as

$$c(P) = \sum_{T_i \in P} c(T_i).$$

Theorem 2.2. *There exists an optimal platoon routing for the unlimited platooning problem in which no two trucks split and then merge again. More rigorously, there exists an optimal platoon routing such that for any pair of its truck paths P_1 and P_2 the following holds: If two subpaths $Q_1 \subset P_1$ and $Q_2 \subset P_2$ start in $u \in V$ and end in $v \in V$ and have intersecting waiting times at both u and v, then $Q_1 = Q_2$.*

Proof. Let S be an optimal platoon routing with fuel cost $C(S)$ in which there are two paths P_1 and P_2 containing subpaths $Q_1 \subset P_1$ and $Q_2 \subset P_2$ both starting at a node $u \in V$ and ending at $v \in V$. Without loss of generality we may assume that $C(Q_1) \leq C(Q_2)$. Let S' be the platoon routing S where P_2 has Q_2 replaced by Q_1. Note that this is still a valid platoon routing since Q_1 and Q_2 have intersecting waiting times.

If an edge traversal in Q_2 has platoon size greater than one, then the reduction in total fuel cost for removing that edge traversal is $c(T) \cdot \eta$. Consequently, by removing Q_2 from P_2, the total fuel cost is reduced by at least $C(Q_2) \cdot \eta$. By inserting Q_1 into $P_2 \setminus Q_2$, we introduce an extra fuel cost of $C(Q_1) \cdot \eta$ since Q_1 is already a subpath of P_1. Hence, the fuel cost of S' is

$$C(S') = C(S) - \eta \cdot C(Q_2) + \eta \cdot C(Q_1) = C(S) + \eta(C(Q_1) - C(Q_2)) \leq C(S),$$

since $C(Q_1) \leq C(Q_2)$. Since $C(S)$ was an optimal platoon routing, $C(S) \leq C(S')$, and hence $C(S) = C(S')$. This implies that for every optimal platoon routing, where a pair of trucks splits at a node u and then merges again at a node v, there is an optimal platoon routing where they share the same truck path from u to v. □

3. NP-Completeness

Theorem 3.1 states the computational difficulty of the general platooning problem. The proof is a reduction from set covering, which Karp (1972) shows is NP-complete, to the unlimited platooning problem. This reduction shows that the platooning problem on general graphs is hard even when deadlines are ignored. However, one can reasonably assume that most road networks correspond to planar graphs. It is hence useful to obtain results on the difficulty of the planar platooning problem as well. Theorem 3.2 shows that the planar platooning problem is NP-complete as well.

3.1. Reduction to the Unlimited Platooning Problem

Theorem 3.1. *The decision version of the platooning problem is NP-complete.*

Proof. Given a finite set $A = \{1, 2, ..., N\}$, a collection of subsets $B \subset \mathcal{P}(A)$, and an integer K, an instance (A, B, K) of the set covering problem consists of determining whether it is possible to find a subcollection $M \subset B$, $|M| \leq K$, such that each element of A is an element in at least one of the sets of M.

(A, B, K) can be reduced to an instance of the platooning problem by creating a graph $G' = (V, E)$ in the following way. First, create a starting node s, the nodes $m_1, m_2, ..., m_{|B|}$ and the nodes $r_1, r_2, ..., r_N$. The node m_i here represents a subset $t_i \in M$, and the node r_n represents the element $n \in A$. Create edges from s to each of the nodes $m_1, m_2, ..., m_{|B|}$, with weight 1. Call these edges left edges. Finally create an edge from m_i to r_n, with weight $1 + \frac{1}{\eta}$, if and only if subset t_i contains the element n. Call these edges right edges. Figure 2 illustrates the setup.

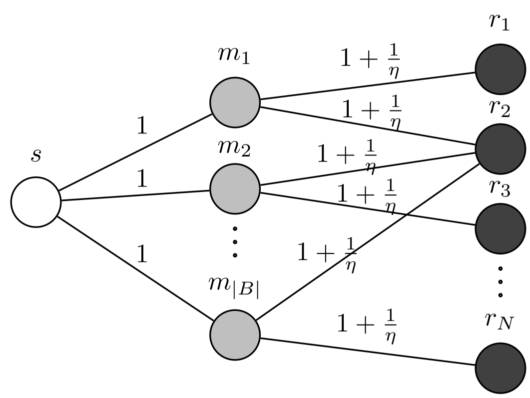


Figure 2: Graph G' created from an instance of the set covering problem. Each node m_i represents a subset in B, and each node r_n represents an element in A. The white node represents a starting node and dark gray nodes represents destination nodes.

To create a platooning problem, we let $H = \{v\}$ such that $f(v) = 1$, and we introduce truck missions

$$M_i = (s, r_i, \infty)$$

for $i = 1, \ldots, N$.

A truck can save at most $1-\eta$ in fuel cost by platooning on a left edge. Since $\left(1 + \frac{1}{\eta}\right) \cdot \eta > 1 > 1 - \eta$, it follows that the cost of a right-edge traversal, even when platooning, is greater than the maximal platooning savings on a left-edge traversal. Thus, in an optimal platoon routing all truck paths will contain as few right-edge traversals as possible. Hence, every truck path will contain only one left-edge traversal and only one right-edge traversal.

We now show that there is a solution to the set covering problem, using at most K subsets if and only if there is a platoon routing on the graph G', with a total cost of at most $K + (N - K)(1 - \eta) + N(1 + \frac{1}{\eta})$. Suppose A can be covered with a subset $M \subset T$, where $|M| = k \leq K$. Then there is a platoon routing containing k different left-edge traversals, each reaching from s to one of the $m_i \in M$. The cost of the platooning routing is

$$\begin{aligned}&(\text{cost for left-edge leaders}) + (\text{cost for left-edge followers}) + (\text{cost for right-edges traversals})\\ &= k + (N - k)\eta + N(1 + \frac{1}{\eta})\\ &\leq K + (N - K)\eta + N(1 + \frac{1}{\eta}),\end{aligned}$$

since there are k platoons traveling a distance of 1 each (k platoon leaders with $N - k$ platoon followers) and since each truck path also contains a right-edge traversal with platoon size equal to one.

It remains to show that if there is an optimal solution to the platoon routing problem on G' with cost less than or equal to $K+(N-K)\eta+N(1+\frac{1}{\eta})$, then there is a set covering of size less than or equal to K. We show the contrapositive. Assume that the smallest set covering of (A, B, K) is a subset $M \subset B$, where $|M| = k > K$. In an optimal platoon routing on G' the truck paths must contain at least k left edges in order for every truck mission to be completed. This is true since each HDV must reach its destination and in order to do that the platoon routing must contain enough middle nodes such that every destination node is "covered." This results in a cost of at least

$$k + (N - k)\eta + N(1 + \frac{1}{\eta}) > K + (N - K)\eta + N(1 + \frac{1}{\eta}),$$

since $N \geq k > K$ and $0 < \eta < 1$.

We conclude that there is a platoon routing on G' with cost less than or equal to $K+(N-K)(1-\eta)+N(1+\frac{1}{\eta})$ if and only if there is a set covering $M \subset B$ of P with $|M| \leq K$. Consequently, the decision version of the platooning problem with a single starting node is NP-complete. □

Note 5. *As a direct consequence of the NP-completeness of the decision version of the unlimited platooning problem, the platooning problem and its unlimited version are both NP-hard.*

3.2. Reduction to the Planar Platooning Problem

Having shown that the platooning problem on general graphs is NP-complete, we now show that the decision version of the platooning problem on planar graphs is also NP-complete.

Theorem 3.2. *The decision version of the planar platooning problem is NP-complete.*

Proof. The theorem follows from a reduction from the decision version of the rectilinear Steiner arborescence problem (RSAP), which is NP-complete. A rectilinear Steiner arborescence (RSA) is a directed tree with nodes on integer coordinates and with arcs from (i,j) to $(i+1,j)$ and $(i,j+1)$ for all $(i,j) \in \mathbb{Z}^2$. RSAP consists of finding an RSA (1) with total edge length less than or equal to a given integer, (2) rooted at the origin, and (3) having nodes in a given set of points in $\mathbb{Z}^2_+$, the first quadrant of $\mathbb{Z}^2$. For more information about the RSAP, see Rao et al. (1992).

Let (R,K) be an instance of RSAP, where R is a set of points $\{p_1,...,p_N\}$ in $\mathbb{Z}^2_+$ and an integer K. (R,K) can be reduced to an instance of the decision version of the planar platooning problem by creating a graph $G_R = (V_R, E_R)$ with the vertex set

$$V_R = \left\{(x,y) \in \mathbb{Z}^2 \,|\, (x,\cdot) \in R \wedge (\cdot,y) \in R\right\} \cup \{(0,0)\}$$

and the edge set

$$E_R = \{(i,j) \in V_R \times V_R \,|\, (x_i = x_j \vee y_i = y_j) \wedge (\ i \text{ and } j \text{ are neighbors})\}$$

where two nodes i and j are neighbors if there is no other node on the line segment connecting i and j. This means that for each pair of nodes an edge will be drawn between them if they share the same x- or y-coordinate and there is no other node between them.

The edge weight will equal the Euclidean distance between the nodes. The graph G_R is called a Hanan grid; and the search for an RSA may, according to Hanan, without loss of generality be restricted to this grid (Hanan, 1966, Theorem 4). An example of a Hanan grid can be seen in Figure 3. Introduce N truck missions. For each truck n let the starting point be $s = (0, 0)$ and the destination $d_n = p_n$. The set of allowed speeds will be $H = \{1\}$. Without loss of generality, we may assume that the fuel cost per unit distance $f(1) = 1$. For each truck n, a deadline $\tau_n = x_n + y_n$ is introduced. These deadlines imply that in every platoon routing, each truck path from s to d_n must be a shortest path from s to d_n with length $\|d_n\|_1$ in the graph G_R. By construction of the platooning problem instance, all edge traversals must go from left to right or from the bottom up.

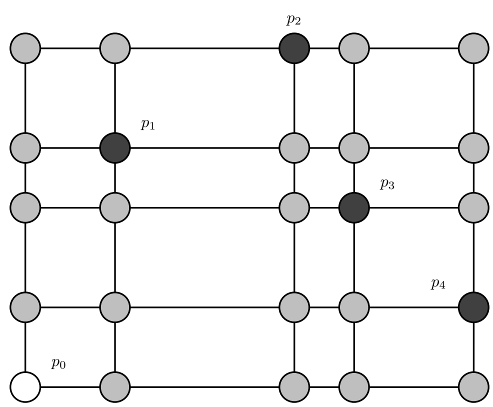


Figure 3: Hanan grid created during reduction from RSA. White indicates starting node, and black indicates destinations.

We will now show that there exists a rectilinear Steiner tree with edge length less than or equal to K if and only if there is a platoon routing for the created platooning problem instance on G_R with total fuel cost less than or equal to

$\eta D + (1-\eta)\cdot K$, where

$$D = \sum_{n=1}^{N} \|d_n\|_1.$$

Note that the total edge length of the RSA is the sum of the lengths of the edges in the RSA, while D is the sum of the distances from the start to every destination.

First, assume there is an RSA with total edge length equal to $k \le K$. For a given RSA there is a corresponding platoon routing S; since the RSA defines a tree, there is only one possible route for each HDV starting at the origin to reach its destination. The total path length (the length of the union of all paths) in the platoon routing S corresponding to this RSA will then be k, and on each of the edge traversals in the platoon routing only one platoon (consisting of one or more HDVs) will drive. Since the total length of all edge traversals still will be D, the total fuel cost will equal

$$\begin{aligned} C(S) &= (\text{cost for trucks driving first}) + (\text{cost for trucks driving behind}) \\ &= k + \eta \cdot (D-k) \\ &= \eta D + (1-\eta)\cdot k \\ &\le \eta D + (1-\eta)\cdot K. \end{aligned}$$

To prove the equivalence, we need to show that if there is an optimal platoon routing to the created platooning problem instance with cost less than or equal to $\eta D + (1-\eta)\cdot K$, then there is an RSA with total edge length less than or equal to K. To this end, we show the contrapositive by supposing that there is no RSA with total edge length less than or equal to K. We further assume that the minimal edge length is $k > K$. Consider an optimal platoon routing S. According to Theorem 2.2, we may assume S to be a platoon routing where no HDVs meet again after having split up. Hence, the union of paths in S will be a tree, and it will in fact be an RSA since every truck path in S must be a shortest path from the origin to a destination. The length of this RSA must hence be at least k, and the total fuel cost of this platoon routing is given by

$$C(S) = k + \eta \cdot (D-k) = \eta D + (1-\eta)\cdot k,$$

which decreases with k. Hence, there cannot be a platoon routing with cost less than or equal to

$$\eta D + (1 - \eta) \cdot K.$$

This implies that the decision version of the planar platooning problem is NP-complete. □

Note 6. *Since the decision version of the planar platooning problem is NP-complete, it follows directly that the planar platooning problem is NP-hard.*

4. Integer Linear Programming Formulation

In this section, we convert the unlimited platooning problem into an ILP. The need for integer variables in our formulation arises because fractional vehicles cannot traverse an edge and because the fuel consumption of a platoon is a piecewise linear function of the number of trucks forming the platoon. We first describe an integer linear programming formulation for the unlimited platooning problem where all truck missions share the same starting node, a scenario that occurs throughout the real world. We then form an ILP for the general unlimited platooning problem. In both formulations the fuel cost per unit distance is assumed to be 1. This does not limit the validity of the solution since one can scale the final result by $f(v)$ to obtain the correct fuel cost. We also propose an extension of the ILP formulation to the most general platooning problem where finite deadlines and a nontrivial set of speeds are allowed.

Let $G = (V, E)$ be a graph, and let $[(s_1, d_1, \tau_1), ..., (s_N, d_N, \tau_N)]$ be a fixed list of truck missions. The different versions of the platooning problem are equivalent to the following ILP problems; by solving them, an optimal platoon routing is easily obtained.

4.1. Unlimited Platooning Problem - Shared Starting Node

We formulate the unlimited platooning problem where $s_1 = \cdots = s_N = s$ for some node $s \in V$ and $\tau_1 = \cdots = \tau_N = \infty$. The variables used in this ILP formulation are contained in Table A.3 in Appendix A.

Definition 9. We define the *same-start unlimited ILP problem* as follows

$$\text{minimize } h = \sum_{(i,j)\in E} w(i,j)\cdot g_{ij} \tag{1}$$

$$\text{subject to } \sum_j x_{ijn} - \sum_j x_{jin} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d_n \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in V, 1 \le n \le N \tag{2}$$

$$b_{ij} = x_{ij1} \vee \cdots \vee x_{ijN} \qquad \forall (i,j)\in E \tag{3}$$

$$g_{ij} = b_{ij} + \eta\left(\left[\sum_{n=1}^{N} x_{ijn}\right] - b_{ij}\right) \qquad \forall (i,j)\in E \tag{4}$$

$$x_{ijn} \in \{0,1\} \qquad \forall (i,j)\in E, 1\le n\le N$$

$$b_{ij} \in \{0,1\} \qquad \forall (i,j)\in E$$

$$g_{ij} \in \mathbb{R} \qquad \forall (i,j)\in E$$

Note 7. *The logical constraints in* (3) *are convertible into linear inequalities. This procedure is explained in Appendix B.1. It is hence justified to call this problem defined an integer linear programming problem.*

We seek to minimize the sum of the joint fuel consumption over each edge (which may be zero if no truck traverses the edge). Constraint (2) ensures that each truck follows a path from the start to its destination. Constraint (3) implies that b_{ij} is set if and only if a truck traverses the edge (i,j). Constraint (4) corresponds to a calculation of the fuel consumption over this edge.

Theorem 4.1. *A cost c is the value of the optimal solution to the same-start ILP problem if and only if c is the cost of an optimal platoon routing for the corresponding same-start unlimited platooning problem. Moreover, using the values of x_{ijn} from the solution, a platoon routing with fuel cost c is retrievable in polynomial time.*

Proof. A platoon routing for the unlimited platooning problem is feasible if for all n, truck path n is a path from s to d_n. As a consequence of Theorem 2.2 and the fact that all HDVs start on the same node, in an optimal platoon routing, all edge traversals over a certain edge have the same time. Consequently, in the same-start ILP formulation we may ignore the times of the edge traversals.

The variable x_{ijn} will be true if truck path n in the platoon routing contains an edge traversal over the edge $(i, j) \in E$, and false otherwise. According to Ahuja et al. (1993, p. 6), the constraint in (2) ensures that for a given HDV n, the edges corresponding to the set variables x_{ijn} will construct a path from s to d_n. The variable b_{ij} is a binary variable for each edge $(i, j) \in E$ and is subject to the constraints in (3).

The constraints in (3) for b_{ij} are set so that b_{ij} is true if x_{ijn} is true for some n, that is, if some HDV traverses (i, j), and false otherwise. Note that the constraints in (3) do not restrict the possible values for the x_{ij} variables. All combinations of paths from start to finish are allowed; hence, for every possible platoon routing for the same-start unlimited platoon problem, there is a corresponding solution to the same-start ILP problem. For the same reason, every solution to the same-start ILP problem has a corresponding platoon routing.

The variable g_{ij} corresponds to the fuel cost per unit distance for the set of trucks that traverses (i, j) in the platoon routing. One can easily see that if no truck traverses (i, j), then g_{ij} is 0; otherwise, g_{ij} is equal to the cost for a platoon leader plus the cost for the trucks following. The objective function is calculated by summing over all edges and equals the total fuel cost of the corresponding platoon routing.

When the objective function h has been optimized, one can easily obtain the truck paths to create a valid platoon routing. For each truck n, construct a truck path by starting at s and traversing G by following edges corresponding to variables x_{ijn} set to true. While doing so, one must keep track of the time taken t_{in} to reach a certain node i. In each step, one appends to the truck path the edge traversal $((i, j), t_{in}, v)$. From each node there will only be one possible edge to traverse, which is guaranteed by Theorem 2.1. Traversing is stopped when d_n is reached. □

Note 8. *With minor modifications (reversing the path retrieval and selecting* *different starting nodes and a shared destination node) the same-start unlimited ILP is applicable to unlimited platooning problem instances where all truck missions share not the same starting node but. rather, the same destination.*

4.2. Unlimited Platooning Problem - Different Starting Nodes

In the next ILP formulation we assume that $\tau_1 = \cdots = \tau_N = \infty$ but allow different starting nodes for the truck missions. When converting this problem without constraints on the starting nodes, the calculation of the total fuel cost is more delicate. An optimal platoon routing may now contain edge traversals that differ only in time, which means that several HDVs can traverse the same edge without platooning. The variables used in the ILP formulation of the unlimited platooning problem are also summarized in Table A.4 in Appendix A.

Definition 10. The *unlimited ILP problem* is as follows

$$
\text{minimize } h = \sum_{(i,j)\in E} w(i,j)\cdot g_{ij} \tag{5}
$$

$$
\text{subject to } \sum_j x_{ijn} - \sum_j x_{jin} = \begin{cases} 1 & \text{if } i = s_n \\ -1 & \text{if } i = d_n \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in V, 1 \le n \le N \tag{6}
$$

$$
(t_{ijn} \ge t_{kin} + w(k,i)) \vee \neg (x_{ijn} \wedge x_{kin}) \qquad \forall i,j,k \in V \text{ s.t } (i,j)\in E \wedge (k,i) \in E, 1 \le n \le N \tag{7}
$$

$$
p_{ijnm} = x_{ijn} \wedge x_{ijm} \wedge (t_{ijn} = t_{ijm}) \qquad \forall (i,j)\in E, 1 \le m \le n \le N \tag{8}
$$

$$
\alpha_{ijn} = x_{ijn} \wedge \neg \left(p_{ijn1} \vee \cdots \vee p_{ijn(n-1)}\right) \qquad \forall (i,j)\in E, 1 \le n \le N \tag{9}
$$

$$
g_{ij} = \sum_{n=1}^{N} (\alpha_{ijn} + \eta\cdot (x_{ijn} - \alpha_{ijn})) \qquad \forall (i,j)\in E \tag{10}
$$

$$
t_{ijn} \le N \cdot \sum_{e\in E} w(e) \qquad \forall e \in E, 1 \le n \le N \tag{11}
$$

$$
x_{ijn} \in \{0,1\} \qquad \forall (i,j)\in E, 1 \le n \le N
$$

$$
t_{ijn} \in \mathbb{Z}_+ \qquad \forall (i,j)\in E, 1 \le n \le N
$$

$$
p_{ijnm} \in \{0,1\} \qquad \forall (i,j)\in E, 1 \le m \le n \le N
$$

$$
\alpha_{ijn} \in \{0,1\} \qquad \forall (i,j)\in E, 1 \le n \le N
$$

$$
g_{ij} \in \mathbb{R} \qquad \forall (i,j)\in E
$$

Note 9. *Once again, we note that it is possible to convert the logical constraints in the above ILP into linear inequalities as explained in Appendix B.2. Hence, the problem is an integer linear programming problem, only formulated more conveniently.*

In this model, we seek to minimize the same objective as in Definition 9. Constraint (6) corresponds to (2) except that it allows for different starting points. Constraint (7) ensures the traversal times for consecutive edges in a truck path increases with at least the edge weight. Constraint (8) calculates a binary variable p_{ijnm} deciding if trucks n and m traverses edge (i, j) at the same time, implying that they are in a platoon. The decision variable α_{ijn} in (9) determines if truck n is a platoon leader of (i, j), using the convention that the truck with the lowest index in a platoon is always the leader. Constraint (10) calculates g_{ij}, the joint fuel consumption over an edge (i, j), taking into account the possibility of multiple platoons at different times. Constraint (11) limits the finish times to make the search space bounded.

Theorem 4.2. *A cost c is the optimal solution to the unlimited ILP problem if and only if c is the cost of an optimal platoon routing to the corresponding unlimited platooning problem. Moreover, using the values of x_{ijn} and t_{ijn} from the solution, a platoon routing with fuel cost c is retrievable in polynomial time.*

Proof. As was the case in the formulation with a shared starting node, the variable x_{ijn} is set if HDV n traverses edge (i, j) in the platoon routing. The constraints in (6) will ensure that, for each HDV n, the edges corresponding to the set x_{ijn} builds a path from s_n to d_n. The variable t_{ijn} corresponds to the time when HDV n started traversing edge (i, j). First we note that there will always be an optimal routing such that all times satisfy the constraints in (11). Choosing a time equal to the value on the right-hand side in (11) would correspond to, for example, an HDV waiting at a node while all other HDVs traverse the whole graph one at a time. This is obviously a generous upper limit for the finish times of any actual platoon routing. The constraints in (7) force t_{ijn} to be greater than or equal to $t_{kin} + w(k, i)$ if there are edges both into node i, (k, i), and out from node i, (i, j), that are traversed by HDV n. This implies that the traversal times increase appropriately during a truck path. The x_{ijn} and t_{ijn} variables are thus constrained to produce valid platoon routings. The remaining variables are only required to provide the proper total fuel cost in the objective function.

Constraint (8) ensures that p_{ijnm} is true if and only if trucks n and m platoon over edge (i, j), that is, both trucks traverse the edge at the same time.

In (9), α_{ijn} is set if x_{ijn} is set to true and no truck with lower index traverses edge (i, j) at time t_{ijn}. This may be interpreted as truck n leading a platoon over (i, j).

The definition of g_{ij} in (10), corresponding to the fuel cost per unit distance for the set of trucks that traverses (i, j), is appropriate because

$$\alpha_{ijn} + \eta \cdot (x_{ijn} - \alpha_{ijn}) = \begin{cases} 0 & \text{if truck } n \text{ does not traverse } (i, j) \\ 1 & \text{if truck } n \text{ leads a platoon over } (i, j) \\ \eta & \text{if truck } n \text{ is in the tail of a platoon over } (i, j), \end{cases}$$

which is exactly the cost per unit distance of the traversal for truck n. The variable g_{ij} hence evaluates to the sum of the costs per unit distance of all edge traversals over (i, j). The objective function sums over all edge traversals in the solution, and this equals the total fuel cost of the corresponding platoon routing.

The retrieval of the truck paths forming an optimal platoon routing for the unlimited platooning problem is similar to the procedure explained in the proof of Theorem 4.1. For each truck n, we construct a truck path by starting at s_n and traversing G by following edges corresponding to set variables x_{ijn}. In each step we append to the truck path the edge traversal $((i, j), t_{ijn}, v)$. Once again, guaranteed by Theorem 2.1, from each node there will only be one possible edge to traverse. We stop when d_n is reached. □

4.3. Extension

Having presented the ILP formulation for the unlimited platooning problem, it is straightforward, though tedious, to extend the formulation to include problem instances of the most general platooning problem where truck missions have finite deadlines and the set H contains more than one single speed. As we will show in the Section 6, however, the limit for solving an unlimited ILP problem within a couple of minutes on a reasonable fast computer lies around 10 trucks. A more complex formulation, such as a potential ILP for the general platooning problem, will likely result in even slower resolution times. However, one should note that this is highly dependent on the values of the deadlines; with strict deadlines few platooning opportunities occur and should result in a near trivial and quick solution.

For the interested reader, we here outline such an extension. The formulation is similar to the unlimited ILP formulation. To keep the formulation linear when introducing multiple allowed speeds, however, we introduce a set of binary variables for each truck, each speed, and each edge. A natural addition is to include the variable m_{ijnv}, which is true if truck n traverses edge (i, j) with speed v, and false otherwise. The constraint for deciding whether two trucks platoon, like the one in (8), now needs to include a check to see that both trucks also use the same speed. Other than these extensions, the ILP formulation for the general platooning problem does not differ excessively from the unlimited ILP.

5. Heuristics

While the formulations in the previous section are useful for solving small problems exactly, large-scale problems result in computationally intractable ILPs. For example, an ILP generated by 10 trucks at different starting nodes on a graph of the German Autobahn takes over 20 minutes to solve, using the default Gurobi branch-and-bound ILP algorithm, on a desktop computer with 8 2.6 GHz processors. Since we have shown the platooning problem to be NP-complete, one is forced to settle with heuristic solvers in order to obtain platoon routings for large instances of the problem. In this section two different constructive heuristics and one improvement heuristic—a local search algorithm—are described. The constructive heuristics are derived from a heuristic developed by Larson et al. (2013). Note that in their most simple form described below, these heuristics are solvers for the unlimited platooning problem. For convenience, in this section, we use the word platoon for describing both a single HDV and a group of HDVs.

5.1. Best Pair Heuristic

We have developed an algorithm, henceforth the Best Pair heuristic, for the unlimited platooning problem, based on the heuristic by Larson et al. (2013). Our algorithm iteratively chooses the current best pair of platoons to merge into, reducing the number of truck mission by one. At each step, the goal is to find the optimal combination of both merging and splitting point for a pair of platoons and replacing their earlier missions with one single mission with the merging point as start and the splitting point as destination. Pseudocode for the algorithm is presented in Algorithm 1.

The "best pair of platoons to merge" is defined as the pair of platoons that save the most fuel by merging. The fuel savings are calculated as the difference between letting the two platoons take their shortest paths by themselves (i.e., no platooning between the two platoons even if their shortest paths overlap) and making them merge into a single platoon between a pair of nodes in the graph. Notice that if the Best Pair heuristic is presented with a same-start platooning problem instance, it will produce the same result as the heuristic by Larson et al. (2013).

The best merging and splitting node for a pair of HDVs is computed by iterating over all pairs of nodes in the graph and finding the combination that produces the greatest fuel savings. A naive implementation of the Best Pair heuristic will have the time complexity

$$\mathcal{O}(N^3 \cdot |V|^2),$$

since the search for best pair of platoons and their merging and splitting points takes $\mathcal{O}(N^2 \cdot |V|^2)$. This operation of merging two HDVs can ultimately be performed $\mathcal{O}(N)$ times. When N merges have been accomplished, the result is the entire fleet of HDVs gathered in one platoon. After minor code optimizations a time complexity of

$$\mathcal{O}(N^2 \log N \cdot |V|^2)$$

can be reached. This is achieved by storing (in a tree structure) the savings of all pairs of platoons found so far so that the greatest savings can be found in

Algorithm 1: Pseudocode for Best Pair Heuristic

input : A graph G, a list of starting nodes S and a list of destination nodes D

output: A platoon routing in G

define Platoon : ($start, destination$,set of trucks)

$P \leftarrow \{\}$

foreach *s, d in S, D and the corresponding truck t* **do**

  $P.add(Platoon(s, d, \{t\}))$

end

while *Savings can be made by merging platoons* **do**

  $p_1, p_2 \leftarrow$ the two platoons that save the most by merging.

  $v_1, v_2 \leftarrow$ the best merging and splitting node for p_1 and p_2.

  remove p_1 and p_2 from P

  add to P a new platoon from v_1 to v_2 with the trucks of p_1 and p_2

end

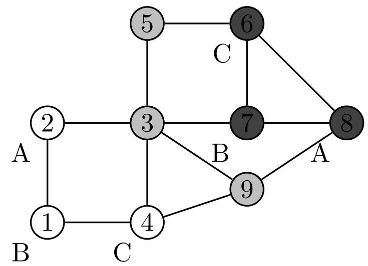
$\mathcal{O}(\log N)$ time. When two platoons are merged, new savings, corresponding to the savings of the new platoon combined with each of the other platoons, are inserted into the tree structure. This operation has time complexity $\mathcal{O}(N \log N \cdot |V|^2)$ and is carried out at most N times.

An example run of the Best Pair heuristic can be seen in Figure 4. White nodes represent starting nodes of platoons, and black nodes represent destination nodes. Each letter in one of the figures represents a truck, and the edge length of all the edges in the given graph is 1. The algorithm runs as follows. In the initial state the savings of the pairs of trucks (A,B), (A,C) and (B,C) are compared. Trucks A and B are then chosen to merge at node 2 and split at node 7 since that produces savings of $2(1-\eta)$ fuel cost. Note that the algorithm could just as well have chosen pair (B,C) which also produces savings of $2(1-\eta)$ fuel cost by platooning from node 4 to node 7. In Figure 4(b) platoon C and AB can platoon over the edge (3,7), and a new platoon, ABC, seen in Figure 4(c) is therefore created. Since no more platoons can be formed, the algorithm is now done.

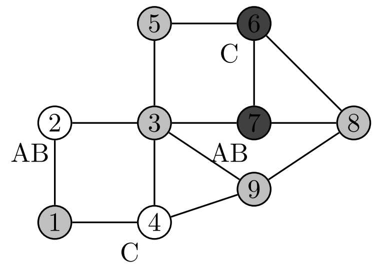
5.2. Hub Heuristic

The idea of the heuristic presented in this section, the Hub heuristic, is to drive platoons through certain nodes called hubs. By selecting such hubs we replace a general platooning problem with multiple subproblems that are easier to solve. The heuristic works by partitioning the trucks and selecting a hub for each partition. To find a platoon routing for a problem instance where each HDV must drive through a certain hub, we first solve the problem of driving the HDVs from their starting nodes to the hub and then solve the problem of driving the HDVs from the hub to their destinations. Both problems can be solved with a same-start solver such as the heuristic described by Larson et al. (2013). The pseudocode for the Hub heuristic can be seen in Algorithm 2.

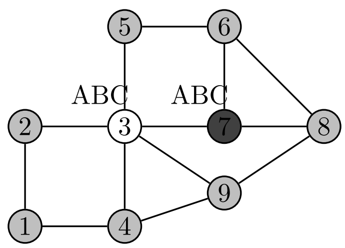
The partitioning of the trucks and the selection of hubs can be made in a multitude of ways. In our implementation of the Hub heuristic, we attempt to merge platoons or trucks with the largest incentive to drive together. We



(a) In the initial state of the Best Pair heuristic all trucks have different starting and destination nodes.



(b) Trucks A and B merge at 2 and split at 7.



(c) Platoon AB and truck C merge at 3 and split at 7

Figure 4: Example run of the Best Pair heuristic

do so by assigning a rating to each edge in the graph for each truck. The rating measures how probable a truck is to drive over a given edge. We can then compare such edge ratings to see whether a pair of trucks should form a platoon. This should generate good platoon routings since two trucks that have a highly ranked edge in common are likely to save fuel by platooning over this edge. For each platoon we create a vector of edge ratings (a real number for each edge representing the ”incentive” for the platoon to drive over that edge). To calculate how compatible two platoons are, we pointwise multiply their edge rating vectors and take the sum over the resulting vector.

We calculate each edge rating in constant time. Thus, finding the pair of platoons with the greatest joint edge rating vector can be done in $\mathcal{O}(N^2 \cdot |E|)$ time by finding the joint edge rating vector of each pair of platoons currently

Algorithm 2: Pseudocode for Hub Heuristic

input : A graph $G = (V, E)$, a list of starting nodes S and a list of destination nodes D representing a set of trucks T

output: A platoon routing in G

```
1 Choose a partition P of T
2 foreach part p ∈ P do
3     Choose a hub h ∈ V
4     Solve the problem of driving trucks in p from their starting nodes to h.
5     Solve the problem of driving trucks in p from h to their destination
      nodes.
6     Combine these two solutions to create a solution to the original
      problem for the trucks in p.
7 end
```

available. Such a search is performed a maximum of N times in the Hub heuristic, because after N merges we end up with one single part in the partition. The time complexity of a naive implementation of the Hub heuristic is

$$\mathcal{O}(N^3 \cdot |E| + N^2 \cdot |V|).$$

The second term in the time complexity stems from having to solve the same-start problems that the Hub heuristic produces. Solving these subproblems using the Best Pair heuristic has time complexity $\mathcal{O}(N^2 \cdot |V|)$.

Just as in the case of the Best Pair heuristic, we can improve the time complexity by storing the savings of each pair of parts in a tree structure so that the largest savings can be retrieved in $\mathcal{O}(\log n)$ time. This optimization produces a time complexity of

$$\mathcal{O}(N^2 \log N \cdot |E| + N^2 \cdot |V|) = \mathcal{O}(N^2 \log N \cdot |E|).$$

5.3. Local Search

In addition to the two construction heuristics, we consider the following improvement heuristic. The improvement heuristic is a local search algorithm that tries to enhance a given platoon routing S by updating a single truck path in S. The goal of the local search algorithm is, given a platoon routing for a set of truck missions, to find the optimal truck path for one of these truck missions given that every other truck path in the platoon routing remains fixed, except possibly for the edge traversal times. The local search algorithm is a generalization of Dijkstra's shortest path algorithm where a truck can not only move alone over edges but also platoon over them where possible. Pseudocode for the local search algorithm can be seen in Algorithm 3.

If all truck paths except for the one currently being improved are immutable during the local search, then there might emerge platooning opportunities that we miss because the current truck does not reach the relevant edges in time. Since we are interested in maximizing our improvement heuristic for the unlimited platooning problem, it is advisable to let all other trucks wait extra time before each edge traversal. This approach will result in more platooning opportunities and hence a better platoon routing.

The order in which we choose the truck paths to improve could be important when running the local search algorithm. In our implementation, we iterate over the truck paths in lexicographic order and improve the truck path of one truck at a time, until no single truck path can be improved anymore, that is, until a local optimum is reached.

The complexity of the local search algorithm is similar to that of a standard Dijkstra's algorithm. The only difference is the number of possible edge traversals; there can be N traversals in the local search algorithm for each traversal in the standard algorithm. Therefore the complexity of running our local search algorithm to update a single truck path is

$$\mathcal{O}(N \cdot |E| \log(N \cdot |V|)).$$

Algorithm 3: Pseudocode for Local Search Algorithm.

input : A graph G, a Platoon Routing S, a starting node s and a destination node d

output: The cost of a Platoon Routing S' with a cost lower than, or equal to, the cost of S

```
Q ← {}
Q.add(truck n at node s at time and cost 0)
while Q not empty do
    cur ← element with smallest cost in Q
    if cur.node not already visited at an earlier time and smaller cost then
        if cur.node = d then
            return cur.cost
        end
        foreach edge e reaching from cur.node do
            Q.add( cur after moving over e )
            if another truck t drives over e later than cur.time then
                Q.add(cur after platooning with t over e)
            end
        end
    end
end
```

6. Performance

To compare our heuristics, we generated random truck missions on a graph (containing 647 nodes and 1,390 edges) representing Germany's Autobahn network. To generate an instance of the same-start platooning problem, we placed 10, 20, . . . , 200 trucks on a random node in the network and assigned each a random destination. This was repeated 20 times. A similar test case of problems was generated by allowing the starting node for each HDV to be randomly generated. Since we want to compare our methods against the optimum and since the platooning problem with different starting nodes is much more difficult to solve exactly, we were only able to compare our heuristics on examples involving at most 10 HDVs. All computational results were generated with $\eta = 0.9$. The choice of η is motivated by the conclusions drawn in earlier literature. The factor η is set to a more modest value of 10% rather than the possible 21% obtained by Bonnet and Fritz (2000).

We note that the Gurobi optimizer is able to solve instances of the same-start unlimited ILP with up to 200 HDVs in only a few minutes. This capability greatly surpasses that of any other platooning formulation or framework. For example, the only previous attempt at finding the exact solution for a platooning problem (that we are aware of) is that of Kammer (2013). The formulation therein is only capable of solving instances of the same-start platooning problem for fewer than 5 vehicles.

To properly calculate the possible fuel savings from platooning, we define a *trivial routing* as a platoon routing in which each truck path consists of a shortest path from its start to its destination with the earliest possible finish time. Because of the definition of the total fuel cost of a platoon routing, trucks may platoon unintentionally as a consequence of their sharing a simultaneous subpath in the trivial routing. We call this phenomenon *natural platooning* since no outside intervention is needed. It is unclear whether natural platooning occurring during computer simulations would translate into real-world scenarios; two trucks traveling on the same arc at the same time may not necessarily platoon.

6.1. Results

We now present the results from running Gurobi on the exact ILP formulations and the heuristic solvers on problem instances with a variable amount of trucks on the German road network. The results are presented by using box plots.

When calculating the fuel savings, we compare the total fuel cost for a platoon routing to the fuel cost of a trivial routing. Figure 5(a) and Figure 5(b) show the maximum possible fuel savings, in percentage of the fuel cost of the trivial routing, for different instances of the unlimited platooning problem. We here ignore natural platooning, and the trivial cost is merely calculated as the sum of the lengths of the shortest paths from starts to destinations.

The percentages presented in Figure 6(a) through Figure 8(a) are computed as follows

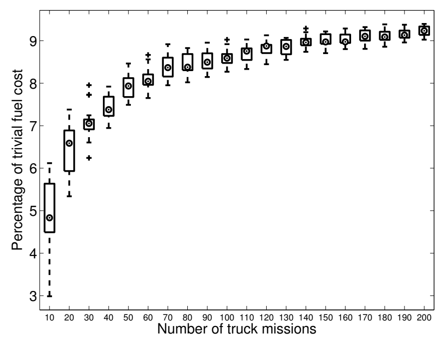
$$\text{Percentage of maximum savings} = \frac{\text{(cost of trivial routing)} - \text{(cost of heuristic solution)}}{\text{(cost of trivial routing)} - \text{(optimal cost)}},$$

where the cost of the trivial routing accounts for natural platooning.

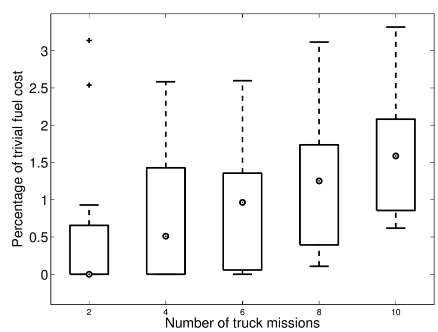
In Figure 6(a) we present the performance of the Best Pair heuristic on the same-start unlimited platooning problem. Figure 6(b) presents the performance of the Best Pair heuristic on the same-start problem, but each solution is improved by the local search heuristic. Figure 7(a) and Figure 7(b) show the performance of the Best Pair heuristic on the different starts unlimited platooning problem, where the latter include improvements from the local search heuristic. Figure 8(a) and Figure 8(b) are the equivalent results for the Hub heuristic.

6.2. Discussion

From Figure 5(a) and Figure 5(b) we conclude that significant fuel savings can be achieved from platooning HDVs. The same-start problem instances naturally present more platooning opportunities since more vehicles are present in



(a) All truck missions share the same starting node.

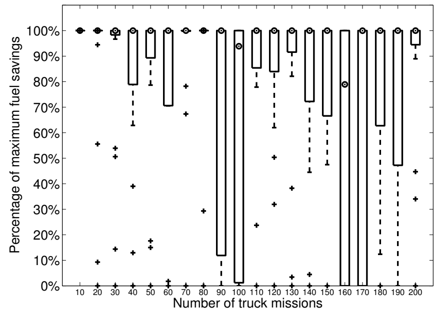


(b) Truck missions are allowed to have different starting nodes.

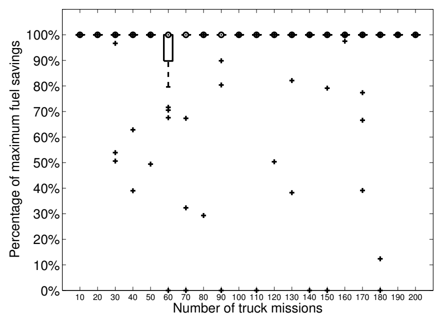
Figure 5: Percentage of the total fuel cost that can be reduced by platooning in the unlimited platooning problem instances with a variable number of trucks. Natural platooning is ignored in the fuel cost of trivial routings.

the larger examples and the trucks' positions are more concentrated, resulting in greater possible fuel savings. Nevertheless, even in platooning problem instances with as few as 10 trucks at different starting nodes, fuel savings of more than 1.5% can be achieved in the majority of cases. We point out that the fuel savings in the different start version of the problem is highly dependent on the starting points and destinations of the trucks; trucks may be placed in the graph in a pattern that provides very few platooning opportunities. Nevertheless, the results of our simulations justify the search for optimal platoon routings.

In Figure 6(a) we can see how the Best Pair heuristic performs on relatively



(a) Best Pair heuristic without subsequent local search

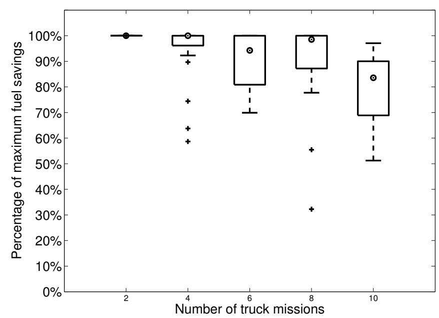


(b) Best Pair heuristic with subsequent local search

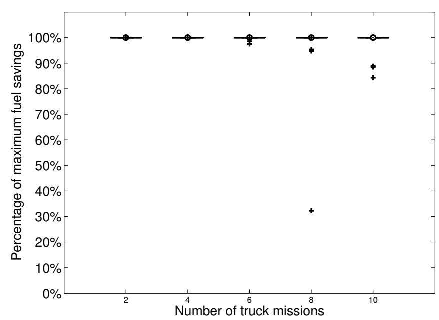
Figure 6: Percentage of maximum fuel savings for the same-start unlimited platooning problem found by the Best Pair heuristic

large problem instances. The heuristic performs well for up to 200 HDVs, with a large amount of the test cases solved optimally. In some test cases, however, where the heuristic completely fails to realize fuel savings when the improvement heuristic is not used. This supplements the results of Larson et al. (2013) and shows that by including more truck missions we can prevent the Best Pair heuristic from finding good platoon routings. After applying the improvement by a local search, we obtain near-optimal results in most cases.

As can be seen when comparing Figure 7(a) with Figure 7(b) and Figure 8(a) with Figure 8(b) the local search algorithm greatly improves the results of both the Best Pair heuristic and the Hub heuristic. Since the local search is able to



(a) Best Pair heuristic without subsequent local search

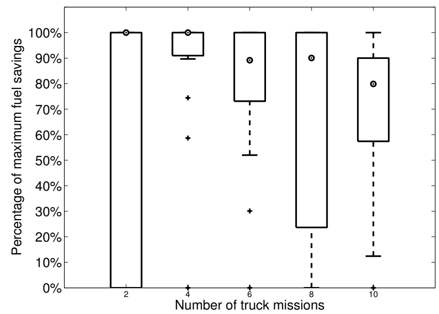


(b) Best Pair heuristic with subsequent local search

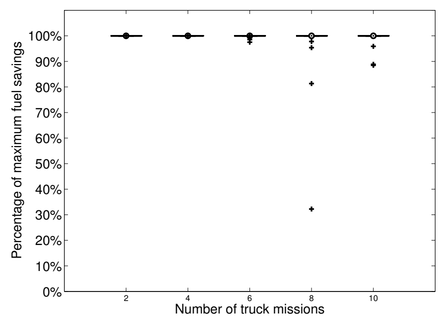
Figure 7: Percentage of maximum fuel savings for different starts unlimited platooning problem found by the Best Pair heuristic

change only a single truck path at a time, we suspect that the improvements to the platoon routings are only minor adjustments. The heuristics combined with these minor adjustments do, however, generate the optimal platoon routings in a vast majority of the problem instances. As for the routes of individual vehicles, the optimal and heuristic solutions all prescribe that the majority of vehicles take their shortest path routes, though there is often (slight) adjustments to their speed to facilitate the formation of platoons.

One may note the wide range in savings in Figure 6(a) and Figure 8(a). This is, in part, due to the Best Pair and Hub heuristics occasionally making irreversible decisions early in the algorithm. For example, the heuristics may



(a) Hub heuristic without subsequent local search



(b) Hub heuristic with subsequent local search

Figure 8: Percentage of maximum fuel savings for the different starts platooning problem found by the Hub heuristic

pair two vehicles that do not platoon in the optimal solution. Once this decision has been made, the heuristic is often committed to a far-from-optimal solution. However, the local search heuristic appears to remedy many of these problems.

The results concerning the heuristics are based on a comparison where the trivial fuel cost was calculated as the sum of all shortest paths between the starting and destination nodes taking natural platooning into account. We believe that using this as the trivial cost produces fairer benchmarks; in the real world, HDVs traveling on the same path will likely take advantage of forming platoons voluntarily. Natural platooning should hence be taken into consideration when evaluating platoon routings.

7. Conclusion

In this paper we minimized the total fuel consumption for HDVs traveling between nodes in a road network by introducing vehicle platooning. The problem of achieving optimal vehicle routings in this aspect was modeled as a graph routing problem—the vehicle platooning problem—which we showed is NP-hard. The NP-hardness applies not only to the general problem but also to special cases such as when all truck missions have the same starting node and no deadlines and to problem instances on planar graphs. To take advantage of already existing software, we formulated different versions of the platooning problem as integer linear programs.

We were able to solve problem instances of up to 200 trucks in a graph representing Germany, when applying the extra constraint that all trucks start on the same node. Removing this constraint, problem instances of size up to 10 HDVs were solved within minutes.

For real-world use, where problem instances of several hundreds or thousands of trucks on graphs much larger than the one studied in this article may occur, one must settle with heuristic or approximate solvers. We proposed three heuristic solvers and compared their results with the optimal solutions obtained by solving the integer linear programming problems. The proposed heuristics perform well on the instances considered. Since these were small problem instances, however, it remains to evaluate the heuristics' performance on larger test cases.

When letting all HDVs start at the same node we found that an optimal platoon routing generated a fuel cost reduction that quickly converged to 9-10%, which is as good as possible considering that platooning vehicles only use 90% of the fuel used by vehicles traveling alone. Substantially smaller problem instances with different starting nodes were solved, though fewer vehicles imply fewer platooning opportunities. Nevertheless, the savings from optimal vehicle platoon routings reveal a significant motivation for continued studies of the platooning problem.

Appendix A. Terminology and Variables

Table A.1: Description of important concepts used in this article

Name	Description
Edge Traversal	A triple consisting of an edge, the starting time of the traversal, and the speed of the traversal
Truck Path	A path from start to destination for a truck, i.e. a list of edge traversals
Truck Mission	A triple containing start, destination, and deadline for a truck
Platoon Routing	A list of truck paths satisfying a set of truck missions
Platoon Size	The number of trucks in a platoon
Platooning Problem	Given a set of truck missions, find a platoon routing with the lowest fuel cost
Unlimited Platooning Problem	Platooning problem without deadlines

Table A.2: Important symbols used in this article

Symbol	Description	Symbol	Description
$G = (V, E)$	graph with vertex set V and edge set E	d_i	destination vertex for truck i
η	fuel reduction factor from platooning	s_i	starting vertex for truck i
$f(v)$	fuel cost per unit distance at speed v	$w(e)$	edge weight of edge e
$c(e)$	fuel cost for traversing an edge e	H	set of allowed speeds
$N_S(T)$	platoon size for edge traversal T	S	platoon routing
M	truck mission $M = [(s_i, d_i, \tau_i)]_i$	T	edge traversal
$C(S)$	fuel cost of platoon routing S	P	truck path
τ_i	deadline for truck i to reach d_i		

Table A.3: Variables used in the ILP formulation of the unlimited platooning problem where all trucks share the same starting node. Variables with indices ij are defined for each edge $(i,j) \in E$, and variables with index n is defined for each truck n.

Name	**Description**	**Type**
x_{ijn}	truck n traverses edge (i,j)	binary
b_{ij}	a truck traverses edge (i,j)	binary
g_{ij}	fuel cost for trucks traversing (i,j)	real

Table A.4: Variables used in the ILP formulation of the unlimited platooning problem. Variables with indices ij are defined for each edge $(i,j) \in E$ and indices n and m corresponds to trucks n and m.

Name	**Description**	**Type**
x_{ijn}	truck n traverses edge (i,j)	binary
t_{ijn}	time when truck n traverses edge (i,j)	bounded integer
p_{ijnm}	truck n and m traverse edge (i,j) at same time	binary
α_{ijn}	truck n has lowest index of all trucks traversing (i,j) at time t_{ijn}	binary
g_{ij}	joint fuel cost for trucks traversing (i,j)	real

Appendix B. Converstion of Logical Constraints

For completeness, we now present the conversion of the logical constraints in Section 4 to linear inequalities.

Appendix B.1. Same-Start Unlimited ILP

Recall the logical constraints in (2).

$$b_{ij} = x_{ij1} \vee \cdots \vee x_{ijN} \quad \forall (i,j) \in E$$

They are equivalent to a number of linear inequalities, namely, the following.

$$\sum_{n=1}^{N} x_{ijn} - N \cdot b_{ij} \leq 0 \quad \forall (i,j) \in E \tag{B.1}$$

$$\sum_{n=1}^{N} x_{ijn} \geq b_{ij} \quad \forall (i,j) \in E \tag{B.2}$$

Suppose x_{ijn} is set for some $1 \leq n \leq N$. Then, the constraint in (2) forces b_{ij} to be true, and b_{ij} must also be set in order to satisfy the constraint (B.1). Now suppose $x_{ijn} = 0$ for all i. Then, (2) enforces that b_{ij} will be false. The constraint in (B.2) also enforces this.

Appendix B.2. Different Starts Unlimited ILP

We will now perform the conversion of logical to linear constraints for the unlimited ILP. Let

$$B = 2 \cdot N \cdot \sum_{e \in E} w(e).$$

The logical constraints in (7)

$$(t_{ijn} \geq t_{kin} + w(k,i)) \vee \neg (x_{ijn} \wedge x_{kin}) \quad \forall i,j,k \in V \text{ s.t } (i,j) \in E \wedge (k,i) \in E, 1 \leq n \leq N$$

are equivalent to the following linear inequalities.

$$t_{ijn} - t_{kin} - B \cdot (x_{ijn} + x_{kin}) \geq w(k,i) - 2B \qquad \forall i,j,k \in V \text{ s.t } (i,j) \in E \wedge (k,i) \in E, 1 \leq n \leq N \tag{B.3}$$

If $(x_{ijn} \wedge x_{kin})$ is false, then (7) does not enforce any constraints on t_{ijn} or t_{kin}. The same is true for (B.3) since the $-2B$ on the right-hand side ensures that the inequality is trivially satisfied, independently of the values of t_{ijn} and t_{kin}. If $(x_{ijn} \wedge x_{kin})$ is true, then (7) constrains t_{ijn} and t_{kin} to satisfy $t_{ijn} \geq t_{kin} + w(i,j)$. In (B.3) $x_{ijn} + x_{kin} = 2$ implying that the inequality is reduced to $t_{ijn} \geq t_{kin} + w(i,j)$. Hence the two formulations are equivalent.

The logical constraints in (8)

$$p_{ijnm} = x_{ijn} \wedge x_{ijm} \wedge (t_{ijn} = t_{ijm}) \quad \forall (i,j) \in E, 1 \leq m \leq n \leq N$$

are equivalent to the following linear inequalities,

$$B \cdot (1 - p_{ijnm}) + (t_{ijn} - t_{ijm}) \geq 0 \quad \forall (i,j) \in E, 1 \leq m \leq n \leq N \tag{B.4}$$

$$B \cdot (1 - p_{ijnm}) + (t_{ijm} - t_{ijn}) \geq 0 \quad \forall (i,j) \in E, 1 \leq m \leq n \leq N \tag{B.5}$$

$$2 \cdot p_{ijnm} - (x_{ijn} + x_{ijm}) \leq 0 \quad \forall (i,j) \in E, 1 \leq m \leq n \leq N \tag{B.6}$$

$$p_{ijnm} \geq (x_{ijn} + x_{ijm}) + (t_{ijn} - t_{ijm}) - B \cdot y_{ijnm} - 1 \quad \forall (i,j) \in E, 1 \leq m \leq n \leq N \tag{B.7}$$

$$p_{ijnm} \geq (x_{ijn} + x_{ijm}) + (t_{ijm} - t_{ijn}) - B \cdot (1 - y_{ijnm}) - 1 \quad \forall (i,j) \in E, 1 \leq m \leq n \leq N, \tag{B.8}$$

where y_{ijnm} is a helper variable deciding which of (B.7) and (B.8) should matter. If y_{ijnm} is true, then (B.7) becomes trivially true and vice versa. Assume $x_{ijn} \wedge x_{ijm}$ is false. Then (8) asserts that p_{ijnm} is false. However, (B.6) ensures that p_{ijnm} can be true only if both x_{ijn} and x_{ijm} are set, and if p_{ijnm} is false, then (B.4) and (B.5) are satisfied independently of t_{ijn} and t_{ijm}. Moreover, (B.7) and (B.8) will be satisfied—one trivially and the other because $(t_{ijn} - t_{ijm}) \leq 0)$ or $(t_{ijm} - t_{ijn}) \leq 0$.

Now let $x_{ijn} \wedge x_{ijm}$ be true. First assume that $t_{ijn} \neq t_{ijm}$. (8) constrains p_{ijnm} to be false. In one of (B.4) and (B.5) p_{ijnm} must be false since either $t_{ijn} - t_{ijm} < 0$ or $t_{ijm} - t_{ijn} < 0$. Both inequalities will then be satisfied. Furthermore, once again (B.7) and (B.8) will be satisfied—one trivially and the other because $(t_{ijn} - t_{ijm}) \leq 0$ or $(t_{ijm} - t_{ijn}) \leq 0$.

Assume that $t_{ijn} = t_{ijm}$, (8) constrains p_{ijnm} to be true. All constraints (B.4), (B.5), and (B.6) are satisfied independently of the value of p_{ijnm}. However, since $(t_{ijn} - t_{ijm}) = 0$, one of the inequalities (B.7) or (B.8) (depending on y_{ijnm}) will become

$$p_{ijnm} \geq 1,$$

which forces p_{ijnm} to be true. The other will be trivially satisfied.

The logical constraints in (9)

$$\alpha_{ijn} = x_{ijn} \wedge \neg \left(p_{ijn1} \vee \cdots \vee p_{ijn(n-1)}\right) \quad \forall (i,j) \in E, 1 \leq n \leq N$$

are equivalent to the following linear inequalities.

$$\alpha_{ijn} + \sum_{k=1}^{n-1} p_{ijnk} \geq x_{ijn} \qquad \forall (i,j) \in E, 1 \leq n \leq N \tag{B.9}$$

$$\alpha_{ijn} \leq x_{ijn} \qquad \forall (i,j) \in E, 1 \leq n \leq N \tag{B.10}$$

$$\alpha_{ijn} \leq 1 - p_{ijnk} \quad \forall (i,j) \in E, 1 \leq k < n \leq N \tag{B.11}$$

Assume x_{ijn} is false. Then (9) sets α_{ijn} to false. The constraints in (B.9) will be trivially true. However, α_{ijn} will be false, since this is enforced by (B.10) and (B.11). Assume x_{ijn} is true. If $p_{ijn1} \vee \cdots \vee p_{ijn(n-1)}$ is false, then (9) constrains α_{ijn} to be true. The same is true for (B.9) since it reduces to $\alpha_{ijn} \geq 1$. If $p_{ijn1} \vee \ldots p_{ijn(n-1)}$ is true, then (9) ensures that α_{ijn} is false. The inequality in (B.9) is satisfied regardless of the value of α_{ijn}.

Acknowledgements

This work was supported by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357, the Swedish Research Council, and the Swedish Foundation for Strategic Research.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc., Upper Saddle River, NJ.

Baskar, L.D., De Schutter, B., Hellendoorn, H., 2013. Optimal Routing for Automated Highway Systems. Transportation Research Part C: Emerging Technologies 30, 1–22. doi:10.1016/j.trc.2013.01.006.

Bonnet, C., Fritz, H., 2000. Fuel Consumption Reduction in a Platoon: Experimental Results with Two Electronically Coupled Trucks at Close Spacing. Intelligent Vehicle Technology doi:10.4271/2000-01-3056.

Browand, F., McArthur, J., Radovich, C., 2004. Fuel Saving Achieved in the Field Test of Two Tandem Trucks. Final Report UCB-ITS-PRR-2004-20. California PATH. URL: http://www.its.berkeley.edu/publications/UCB/2004/PRR/UCB-ITS-PRR-2004-20.pdf.

European Commission, 2011. Roadmap to a Single European Transport Area Towards a Competitive and Resource Efficient Transport System, in: Transport White Paper. COM(2011) 144 Final, Brussels. URL: http://ec.europa.eu/transport/themes/strategies/doc/2011_white_paper/white_paper_com(2011)_144_en.pdf.

Franceschetti, A., Honhon, D., Van Woensel, T., Bektas, T., Laporte, G., 2013. The Time-Dependent Pollution-Routing Problem. Transportation Research Part B: Methodological 56, 265–293. doi:10.1016/j.trb.2013.08.008.

Hanan, M., 1966. On Steiner's Problem with Rectilinear Distance. SIAM Journal on Applied Mathematics 14, 255–265. doi:10.1137/0114025.

Kammer, C., 2013. Coordinated Heavy Truck Platoon Routing Using Global and Locally Distributed Approaches. Master's thesis. KTH - Royal Institute of Technology. URL: http://people.kth.se/~jeffreyl/Thesis_MS-Kammer.pdf.

Karp, R.M., 1972. Reducibility among Combinatorial Problems, in: Miller, R.E., Thatcher, J.W. (Eds.), Complexity of Computer Computations. Plenum Press, pp. 85–103. doi:10.1007/978-1-4684-2001-2_9.

Kavathekar, P., Chen, Y., 2011. Vehicle Platooning: A Brief Survey and Categorization, in: Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, pp. 829–845. doi:10.1115/detc2011-47861.

Larson, J., Kammer, C., Liang, K.Y., Johansson, K.H., 2013. Coordinated Route Optimization for Heavy Duty Vehicle Platoons, in: Proceedings of the

IEEE Intelligent Transportation Systems Conference, pp. 1196–1202. doi:10.1109/itsc.2013.6728395.

Larson, J., Liang, K.Y., Johansson, K., 2015. A distributed framework for coordinated heavy-duty vehicle platooning. Intelligent Transportation Systems, IEEE Transactions on 16, 419–429. doi:10.1109/TITS.2014.2320133.

Lecluyse, C., Woensel, T., Peremans, H., 2009. Vehicle Routing with Stochastic Time-Dependent Travel Times. 4OR 7, 363–377. doi:10.1007/s10288-009-0097-9.

Rao, S.K., Sadayappan, P., Hwang, F.K., Shor, P.W., 1992. The Rectilinear Steiner Arborescence Problem. Algorithmica 7, 277–288. doi:10.1007/bf01758762.

Robinson, T., Chan, E., Coelingh, E., 2010. Operating Platoons on Public Motorways: An Introduction to the SARTRE Platooning Programme, in: Proceedings of the 17th ITS World Congress. URL: http://www.sartre-project.eu/en/publications/Documents/SARTRE_Overview_Final_Paper_ITS_World_Congress_2010.pdf.

Schittler, M., 2003. State-of-the-art and Emerging Truck Engine Technologies for Optimized Performance, Emissions and Life Cycle Costs, in: 9th Diesel Emissions Reduction Conference, Rhode Island. URL: http://www.osti.gov/scitech/servlets/purl/829810.

Schroten, A., Warringa, G., Bles, M., 2012. Marginal Abatement Cost Curves for Heavy Duty Vehicles, in: Background Report. CE Delft, Delft. URL: http://ec.europa.eu/clima/policies/transport/vehicles/heavy/docs/hdv_2012_co2_abatement_cost_curves_en.pdf.

Shladover, S.E., 2007. PATH at 20 - History and Major Milestones. IEEE Transactions on Intelligent Transportation Systems 8, 584–592. doi:10.1109/itsc.2006.1706710.

Taleb, T., Benslimane, A., Letaief, K.B., 2010. Toward an Effective Risk-Conscious and Collaborative Vehicular Collision Avoidance System. IEEE Transactions on Vehicular Technology 59, 1474–1486. doi:10.1109/tvt.2010.2040639.

Tatchikou, R., Biswas, S., Dion, F., 2005. Cooperative Vehicle Collision Avoidance Using Inter-Vehicle Packet Forwarding, in: Proceedings of the IEEE Global Telecommunications Conference, pp. 2762–2766. URL: http://www.bibsonomy.org/bibtex/2f642d4cca09b98c92158c21523c76bd7/dblp.

Tsugawa, S., Kato, S., Matsui, T., Naganawa, H., Fujii, H., 2000. An Architecture for Cooperative Driving of Automated Vehicles, in: Proceedings of the IEEE Intelligent Transportation Systems Conference, pp. 422–427. doi:10.1109/ITSC.2000.881102.