



DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,  
SECOND CYCLE, 30 CREDITS  
*STOCKHOLM, SWEDEN 2017*

# **Evaluating the Handling of New Assignments in a Global Truck Platooning System Using Large- Scale Simulations**

**ERIK IHRÉN**

**KTH ROYAL INSTITUTE OF TECHNOLOGY  
SCHOOL OF ELECTRICAL ENGINEERING**

---

# Evaluating the Handling of New Assignments in a Global Truck Platooning System Using Large-Scale Simulations

Erik Ihrén  
eihren@kth.se

## Abstract

Truck transportation is a big part of goods transportation around the world, with over 100,000 trucks per day in Sweden alone. By using adaptive cruise control, trucks can drive very close together in a platoon. Driving close together leads to a significant reduction in air resistance, which in turn results in lower fuel consumption. This thesis investigates a global coordination system that trucks can register with, which will plan how all the trucks should drive. This system then has to regularly change the speed profiles of individual trucks throughout the system to automatically have them form ' platoons' so that they can save as much fuel as possible.

This thesis specifically investigates what happens when a new truck assignment is registered with the system. When this happens, the system should take this into consideration and change the truck platoons. This might mean breaking up existing platoon or forming new ones. This has to be done regularly in an efficient manner as more and more assignments are added to the system. To investigate the fuel savings of this system, a simulation engine was created. Different parameters, such as update frequency and the time in advance that the system is informed of new assignments, are evaluated to see how they impact the result. In simulations of 5,000 trucks in one day, the coordination system was able to accomplish a reduction in fuel consumption of around 7%.

---

## Sammanfattning

Lastbilstransport är en viktig del av godstransportering runt om i världen. Bara i Sverige kör över 100 000 lastbilar per dag. Genom att använda adaptiv farthållare kan lastbilar köra mycket nära varann i en konvoj. Att köra tätt ihop leder till en markant reduktion i luftmotstånd, vilket i sin tur leder till lägre bränsleförbrukning. Detta examensarbete undersöker ett globalt koordineringssystem som lastbilar kan registrera sig i. Detta system planerar exakt hur varje lastbil ska köra. Systemet måste sedan regelbundet ändra hastighetsprofilerna för de olika lastbilarna i systemet så att de automatiskt formar konvojer som sparar så mycket bränsle som möjligt.

Mer specifikt undersöker detta examensarbete vad som händer när nya lastbilsuppdrag registreras i systemet. När detta händer, bör systemet ta hänsyn till detta och potentiellt ändra konvojerna. Detta kan innebära att man bryter upp befintliga konvojer eller bildar nya. Detta måste ske regelbundet på ett effektivt sätt, allt eftersom fler och fler uppdrag läggs till i systemet. För att undersöka bränslebesparingarna i detta system skapades en simuleringsmotor. Olika parametrar, såsom uppdateringsfrekvens och förvarning innan ett nytt lastbilsuppdrag startar, utvärderas för att se hur de påverkar resultatet. I en simulering av 5 000 lastbilar på en dag uppnådde systemet en minskning i bränsleförbrukning på cirka 7%.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem Statement . . . . .	2
1.3	Evaluation . . . . .	4
1.4	Thesis Outline and Contributions . . . . .	4
<b>2</b>	<b>Background</b>	<b>6</b>
2.1	Platooning . . . . .	6
2.2	Coordination System . . . . .	7
2.3	Current Vehicle Platooning Projects . . . . .	7
<b>3</b>	<b>Computing Fuel-Efficient Plans for Platooning</b>	<b>8</b>
3.1	Plan and Assignment Models . . . . .	8
3.2	Intersection of truck paths . . . . .	9
3.3	Calculating adapted plans . . . . .	11
3.4	Coordination graph . . . . .	12
3.5	Clustering Solution for Coordination Graph . . . . .	13
3.5.1	Greedy clustering . . . . .	15
3.5.2	Random clustering . . . . .	16
3.5.3	Sub-modularity clustering . . . . .	17
<b>4</b>	<b>Dynamic Updating of Coordination System</b>	<b>19</b>
4.1	Updating existing edges . . . . .	20
4.2	Creating new edges between old nodes . . . . .	22
4.3	Updating the clustering solution . . . . .	23
<b>5</b>	<b>Simulation Setup</b>	<b>24</b>
5.1	Truck assignments . . . . .	25
5.2	Assignment Start Interval . . . . .	29
5.3	Generated Data During a Simulation Run . . . . .	30
5.4	Testing of the Simulation Results . . . . .	32
5.5	Simulation Parameters . . . . .	33
5.6	Simulation Fuel Model . . . . .	34
5.7	Size of Test Datasets . . . . .	35
5.8	Hardware and Software for Simulation . . . . .	36
<b>6</b>	<b>Simulation Results</b>	<b>36</b>
6.1	Clustering methods . . . . .	36
6.2	Steady State of Active Trucks . . . . .	38
6.3	Effect of Increasing Horizon . . . . .	39

---

6.4	Effect of Increasing Update Frequency . . . . .	41
6.5	Expected fuel consumption . . . . .	42
6.6	Plan Durations . . . . .	43
6.7	Percentage of platooning trucks . . . . .	44
6.8	Capacity for Real-time System . . . . .	44
<b>7</b>	<b>Conclusion &amp; Future work</b>	<b>46</b>
7.1	Conclusion . . . . .	46
7.2	Future work . . . . .	47
<b>8</b>	<b>References</b>	<b>48</b>

---

# 1 Introduction

This section introduces the thesis. The thesis seeks to reduce fuel consumption of trucks by having trucks automatically form platoons. Section 1.1 looks specifically at why we are interested in how to reduce fuel consumption and why it is important. It also introduces the concept of platooning and how it can be used to lower the fuel consumption.

In Section 1.2 we formulate the problem of how to create a global coordinator system that controls the trucks in the system to form platoons efficiently. It also looks at how this thesis differs from previous research by looking more into how to handle new assignments.

In Section 1.3 we look at how to evaluate our solution to the problem statement by creating a simulation engine. Finally, in Section 1.4 the rest of the thesis is summarized, section by section.

## 1.1 Motivation

Economies around the world rely on transportation to deliver goods. Some of the most common ways of transporting goods and wares is by using ships, trains, trucks, and airplanes. One of the most flexible modes of transportation is using trucks. Roads are more widespread than rail-tracks and airports, and getting trucks to transport goods from point A to point B is less of a logistical challenge than most alternatives.

The fuel consumption of trucks is an important issue, not just for the companies using them, but for society. While reducing the fuel consumption means reduced cost (leading to a higher profit) for the companies, it also implies lower emissions and therefore less pollution. At this time, governments are increasingly aiming to cut down on CO2 emissions [7] [19]. Reducing fuel consumption is therefore a topical and important issue.

To reduce fuel consumption on a large scale, the concept of platooning is used. A platoon in this context is a set of trucks driving close together in a line. The front truck is called the platoon leader and the rest are the platoon followers. The trucks in the platoon can then use cooperative adaptive cruise control to drive close to each other.



Figure 1: A truck platoon of Scania trucks. [4]

When the platoon drives closely together, it causes a slipstream effect, reducing the air resistance for the trucks in the back [11]. This can be seen in Figure 2. Moving the truck forward requires energy. As air resistance increases [9] the truck has to spend more energy to move forward. Since more energy means spending more fuel to move forward, this technique can be used to reduce fuel consumption. To fully take advantage of this, trucks can slow down or speed up at certain points to meet up with other trucks for parts of their trip.



Figure 2: The air flow for three trucks driving in a platoon. [5]

## 1.2 Problem Statement

To make this concept a reality, a system is needed to calculate the best way for trucks to form platoons that minimize fuel consumption. In this system, a global coordinator would receive requests from truck companies detailing assignments for trucks to go from point A to point B. Using the knowledge of all the active assignments, the coordinator system can decide how the different trucks should drive to form platoons that minimize fuel consumption. The coordinator system can control the trucks using a concept of "truck plans". A truck plan specifies how a truck will move from the start to the

---

destination. A truck plan consists of a route to the destination and speed information, detailing the speed to drive at at any given point of time. A truck also has an assignment describing where it starts, where it needs to go, and the deadline by which it has to arrive.

Previous research [35] has considered the problem of truck platooning where, given a set of trucks with assignments, the system assigns them each a plan. The system starts with full information of every assignment that will enter the system. The system calculates the plans once, and never changes any of the plans. However, this is not compatible with a real-world system where new truck assignments come in regularly. Since this system only calculates the plans once, it will eventually be done when all the initial trucks stop platooning.

This thesis seeks to investigate a long-running system that handles truck platooning in a fuel efficient manner while being able to handle new assignments. A long-running system is here defined as a system that runs continuously, receiving and executing request after request, while keeping its internal state up-to-date. This would run as a coordination system on the Internet. As trucks start their journey, they send a request to the coordination system, registering with the system. The coordination system keeps track of the position of all the trucks, and has the ability to update the plans in the system.

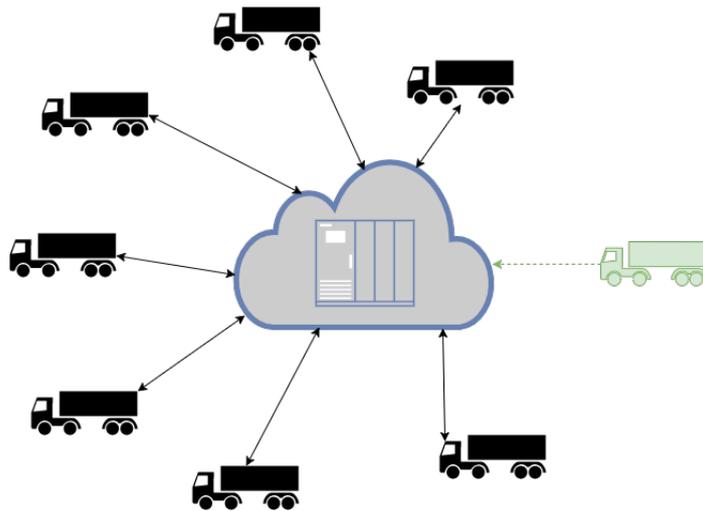


Figure 3: The system communicates back and forth with registered trucks. When a new assignment comes in, the new truck registers with the system.

---

A long-running system brings a lot of new challenges that has to be dealt with. Instead of running a one-time calculation to generate plans for a set of trucks, the algorithm needs to run continuously. As more trucks enter the system, new platoons may have to be formed and added to the system. Existing platoons may have to split up as trucks joins different platoons.

For example, imagine three trucks driving in a platoon when a fourth truck gets registered to the system. The system may come to the conclusion that one of the platooning trucks would save more fuel if it split up from the platoon and sped up to meet the new truck to form another platoon.

The main question that this thesis seeks to answer is: What percentage of fuel saving is possible to achieve with this kind of long-running system when running with larger, realistic amounts of trucks in the system? What are the parameters that impact the fuel saving results, and finally how long does simulations take and are they feasible for a real-time system?

### **1.3 Evaluation**

A simulation engine was created to investigate the viability of a long-running system, and the performance of different algorithms used in the system. For evaluation, several large test sets of truck assignments were randomly generated to model real world transport assignments. These truck assignments were then used in the simulation to investigate the effect of systematic parameter variations on the obtained fuel consumption reduction.

To calculate the efficiency of this system, we needed to calculate how much fuel is saved by following the plans calculated by the coordination system compared to driving like normal and following their default plans. Since it is easy to judge the performance of a system, it also becomes easy to evaluate different methods and algorithms in the system.

The system was tested on several different datasets. Each dataset is a set of truck assignments with randomly sampled start positions and goals. Different algorithms and different parameters were then swapped between to see what net impact it had on the fuel savings.

### **1.4 Thesis Outline and Contributions**

This section provides an outline of the thesis and describes the content and contributions of the different sections.

---

**Section 2: Background** This section provides the background of the thesis. It looks at the current state of platooning, and mentions related works.

**Section 3: Computing Fuel-Efficient Plans for Platooning** This section looks at the core of the platooning system, based on work by van de Hoef et al [37] [38]. This is the base of the coordination system, on which the handling of new assignments is later built.

**Section 4: Dynamic Updating of Plans** This is the main theoretical contribution of the thesis. This section investigates how to handle new assignments and incorporate them into the system. This involves updating the coordination graph in an efficient manner and recalculating the solution.

**Section 5: Simulation Setup** This section explains the details behind the simulation engine built for evaluating the system. It explains how the input assignments were generated, what data is generated after running a simulation, which parameters we are looking at, the size of the datasets as well as the software and hardware that the simulations were run on.

**Section 6: Simulation Results** These are the results from the simulation. It shows that for the equivalent of 5% of Sweden's truck traffic, it is possible to reduce fuel consumption with around 7%. The result also shows that increasing the horizon (how long before a truck starts that the system gets informed of the assignment) improves the result. Increasing the frequency at which plans are updated also improves the fuel savings. Finally, other aspects are investigated as well, such as the similarities between the different clustering methods, the average length of plans, and the percentage of platooning trucks at any given time.

**Section 7: Conclusion & Future work** In the final section, conclusions are drawn regarding the viability of the system. The section discusses some of the limitations and restrictions made in the thesis to limit the scope of the investigation. It also looks at what the maximum capacity of the system would be that still allows it to run in real-time. Finally, it looks at future work for things that are left to investigate.

---

## 2 Background

Platooning is a wide research area with a lot of aspects to investigate. There are many benefits to platooning. It allows you to reduce fuel consumption, traffic, and CO<sub>2</sub> emissions. The background is mainly divided into two parts. There is research on platooning, where two vehicles have to drive close to each other to achieve a reduced air resistance. This is primarily discussed in Section 2.1. Section 2.2 looks at how to use the concept of platooning with a coordinator system to create and modify several platoons on a larger scale. Finally, Section 2.3 looks at some current research projects that are working on bringing efficient platooning into the real world.

### 2.1 Platooning

Platooning of trucks dates back to the 1990's with the Chauffeur project [34] [21] [22]. For the trucks to form platoons, they need to drive very close to each other. A common technology for this is adaptive cruise control, which allows vehicles to follow the speed of the vehicle ahead. There is still a lot of research being done on adaptive cruise control [17] [26], but for platooning purposes there is also more specialized research into cooperative adaptive cruise control [28] [33]. This is a type of adaptive cruise control where the vehicles communicate using Vehicle-to-Vehicle (V2V) communication.

The idea is to let the cruise control system perform the longitudinal vehicle control [30]. By controlling the throttle and breaks, the system can make sure the vehicle stays at a close, but safe distance to the vehicle in front. The latitudinal control is handled by the driver of the vehicle, who is in charging of steering.

In addition to the work on the technology for these platooning system, there have been experimental studies [27] with real trucks to study the act of creating and splitting up platoons.

When platooning, trucks have to decide on a platoon speed. This speed may change based on traffic or other factors, so planning is needed to do this as efficiently as possible. There have been multiple studies on how to do this well [18] [29].

There has also been multiple studies on the effect that platooning can have on traffic [20] [31]. As for studies on fuel savings, experimental studies [12] [13] have shown fuel savings of 8-20% for a truck that follows behind another

---

truck. Although this depends on the truck speed and the distance between the trucks.

While platooning has a lot of positive effects, it can also come with some risks. Trucks driving at a lower speed than the rest of traffic can lead to other drivers trying to pass. However it can be hard to overtake a long platoon of trucks. This can lead to safety issues where drivers get themselves into an unsafe situation trying to overtake a long platoon. This issue, and the specific factors involved have been studied in [24].

## 2.2 Coordination System

There has also been a lot of research on coordination systems for platooning. In one study [25], instead of looking at how a global system can control all trucks, it distributes the problem by reducing it to what they call the "local controller problem". As a solution to the local controller problem, they investigated placing controllers at junctions, and focusing on whether two trucks coming towards the intersection should platoon or not. They refer to a global controller, controlling the timing and routes of all the trucks, as "computationally intractable".

There has been studies [23] of global platooning systems before. The study by Kammer looks at both a global and distributed solution and get fuel savings around 5-8%, depending on the parameters and number of trucks.

This work is largely based on papers written by van de Hoef et al [10] [35] [37] [38], who formed the basis for the theory described in Section 3.

## 2.3 Current Vehicle Platooning Projects

This section describes some current, or recent, projects related to truck platooning.

**California PATH** - PATH is a research and development program from University of California, Berkeley which is focused on Intelligent Transportation Systems. The program was founded in 1986 and is focused on addressing the challenges of transportation in California [15]. Recently, the program has done several studies on platooning [13] [16] [30] [34].

**COMPANION Project** - COMPANION is a project group consisting of several partners including Scania, KTH and Volkswagen. The main goal of

---

COMPANION is to develop a "real-time coordination system to dynamically create, maintain and dissolve platoons, according to a decision-making mechanism, taking into account historical and real-time information about the state of the infrastructure (traffic, weather, etc.)." [1].

**SARTRE** - Safe Road Trains for the Environment (SARTRE) is a European platooning project that ran between 2009 and 2012. Its mission was to investigate technologies behind platooning and forming of a "road train". The road train is controlled both longitudinally and latitudinally by the vehicle at the front of the train. This was demonstrated successfully in Sweden at the end of the project.

### 3 Computing Fuel-Efficient Plans for Platooning

This section seeks to present the underlying method for computing a fuel-efficient way to create the platoons. It is focused on looking at the trucks that are currently registered in the system and calculating a good solution for them from scratch. This section mostly summarizes the relevant results of the work presented in [36] [37] [38].

In Section 4, the method for computing fuel-efficient plans will be expanded to investigate how to handle new truck assignments.

First the model of the problem is described in Section 3.1. Then before we start generating fuel-efficient platoons on a large-scale, we limit the scope to focus on the platooning of two trucks. Assume there are two trucks with separate assignments. We need to calculate if these two have any part of their paths in common. Otherwise they will never be able to form a platoon. This is investigated in Section 3.2.

The next step is to see if there is a fuel-efficient plan for two trucks to form a platoon. This is described further in Section 3.3. This information can then be used to reduce the bigger problem into combining pairwise plans into a graph, as described in Section 3.4.

#### 3.1 Plan and Assignment Models

Initially, we define our model. A truck assignment is defined by  $A = (x^s, x^d, t^s, t^d)$ , where  $x^s$  is the start,  $x^d$  is the destination,  $t^s$  is the start time and  $t^d$  is the

---

deadline, by which time the truck has to be at the destination.

We also define the concept of a plan as  $P = (e, v, t)$ , where  $e$  is the list of edges  $(e_0, e_1, \dots, e_n)$  in the route from  $x^s$  to  $x^d$ . This list of edges does not change over time. As a limitation in the model, the system never changes the route of a truck. A plan describes how a truck's assignment should be fulfilled. The plan contains speed information, which is stored in  $v$  and  $t$ , where  $v$  describes the different speed changes to do to synchronize with the other trucks in the platoon. The list  $t$  describes the times when these speed changes should take place.

The point of the system is to generate plans that minimize fuel consumption. These plans can change as the truck is driving. This can happen when a new truck assignment is created, if for example, a truck would save more fuel platooning with the newly added truck than with the one it is currently platooning with.

For the platoons, two trucks are considered to platoon if they are driving along the same road on the same position. The fact that one of the trucks will have to be slightly behind the other truck is ignored for simplicity, and instead it is assumed that they have the same position.

### 3.2 Intersection of truck paths

We want to look at two trucks and determine if they can form a platoon. The trucks each have a path that they will take to reach  $x^d$ , the destination of the truck's assignment. This path is represented as a list of edges in a graph. To start out with, we need to calculate if these paths intersect each other, meaning that they share a route. The second step is to look at the timing aspect, to see if the trucks will be able to meet up and platoon on the shared route. This section (3.2) solely looks at finding shared routes. The timing aspect is calculated in Section 3.3.

The two trucks will each have a route to their destinations. We define these routes as  $e^1 = (e_0^1, e_1^1, \dots, e_n^1)$  and  $e^2 = (e_0^2, e_1^2, \dots, e_n^2)$ .

We make an assumption that these two routes intersect each other for at most one continuous section. To show this assumption, we define a function  $k(x) : e \rightarrow \{0, 1\}$  to determine if the first truck's path intersect the second truck's path in edge  $x$ . More formally:  $k(x) = \exists y : e_x^1 = e_y^2$ . Our assumption

---

can now be written as

$$\forall i, j, x [i < x \wedge x < j \wedge k(i) \wedge k(j) \rightarrow \neg k(x)]$$

In other words, if  $e^1$  intersects  $e^2$  in  $i$  and  $j$ , there is no point  $x$  between  $i$  and  $j$  where  $e^1$  does not intersect  $e^2$ .

The reason we are making this assumption is that we assume that trucks are rational and pick the shortest path between  $i$  and  $j$ . Assuming that both trucks get the same result for the shortest path between  $i$  and  $j$ , they should stay together the whole way.

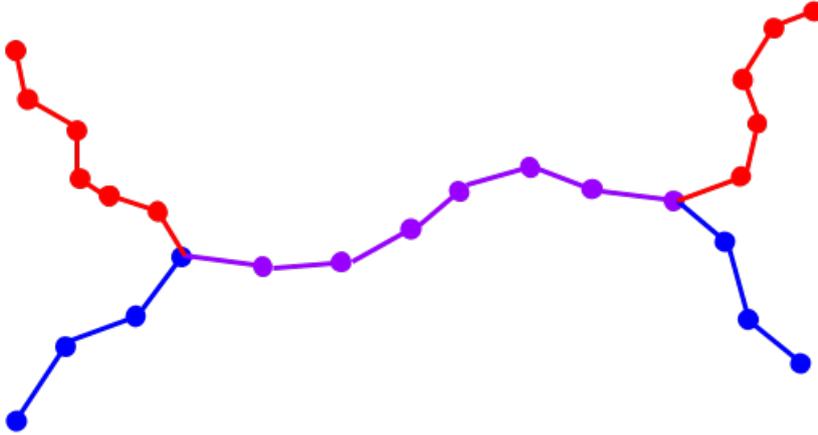


Figure 4: Two paths in a road network, shown in red and blue. The common path for the two trucks is shown in purple.

The first observation, which lets us transform this into a polynomial problem, is that edges in the route are unique for each route. Since there is only a start and a destination, there is no point in visiting the same route edge multiple times, as that would effectively mean driving in circles.

Since we previously assumed that the routes only intersect in at most one continuous section, we can calculate the total amount of edges in the intersection as  $b = |e^1 \cap e^2|$ . If we can find the first edge where  $e^1$  and  $e^2$  intersect, we can describe the full intersection as  $\{e_a^1, \dots, e_{a+b}^1\}$ , where  $a$  is the index in  $e^1$  of the first intersection. This can be found by going through  $e^1$  linearly and for each edge, see if it is a part of  $e^1 \cap e^2$ , which is a constant-time operation.

---

Calculating the intersection therefore ends up being a linear operation, since it just needs to do a set intersection of edges, and then walk through one of the paths and find the first intersection.

---

**Algorithm 1** Path intersection

---

```

1: procedure LONGESTCOMMONPATH( $e_1, e_2$ )
2:    $Intersections \leftarrow e_1 \cap e_2$ 
3:    $FirstIntersection \leftarrow 0$ 
4:   for  $edge \leftarrow e_1$  do
5:     if  $edge \in Intersections$  then
6:        $break$ 
7:    $FirstIntersection \leftarrow FirstIntersection + 1$ 
return  $e_1[FirstIntersection : FirstIntersection + |Intersections|]$ 

```

---

### 3.3 Calculating adapted plans

In the previous section, we looked at whether two trucks have a common path to their destination. We did not look at if it is at all possible for the trucks to meet up and platoon, and if it is feasible for them to do so.

To do this, we use methods based on previous research [37] [38]. The basic idea is to have one of the trucks keep going its normal speed and then calculate the fuel-optimal way for the other truck to reach it. We calculate this both for the first truck adapting to the second truck and for the second truck adapting to the first truck.

An adapted plan consists of three parts. First there is the merge, where the follower drives at one speed while catching up with the leader. Then there is the platooning part where the trucks are driving together in the platoon, driving the same speed as the leader. Finally there is the part after the platoon splits up. At this point, the follower drives at the speed necessary to reach its destination.

We need to calculate the most fuel-optimal merge speed. Driving faster might allow the trucks to platoon longer, but driving faster also burns more fuel. This thesis bases the fuel-optimal merge speed calculation on the research by van de Hoef et al [38] which assumes a linear affine fuel-model and calculates the merge speed from that.

When the merge speed has been calculated, the other parts become easier.

---

The platoon speed is already decided based on the speed that the platoon leader is going. The speed after the split is going to be the slowest acceptable speed that still allows the truck to make it to its goal in time for the deadline.

An adapted plan details the interactions between a coordination leader and a coordination follower. The coordination follower follows the adapted plan and changes its speed to catch up to the coordination leader. Meanwhile, the coordination leader continues on like normal, following its default plan. This is done so it can be combined later on. By having the leader drive like normal, we ensure that we can have multiple followers following the same coordination leader, forming a platoon with more than two trucks.

### 3.4 Coordination graph

After calculating the adapted plans between each pair, according to the method discussed in Section 3.3, we need to store it. We will look at which pairs of trucks can platoon, and how much fuel can be saved. This information can then be saved in a coordination graph. This is a weighted directed graph, describing relationships where one truck can adapt to another one to save fuel. The edge weight describes the fuel saved if the two trucks form a platoon.

There are not edges between every pair of nodes. There are two possibilities for why two nodes would not have an edge between them. First, it might not be possible for two trucks to platoon. Even if they have part of the path in common, it might not be possible for them to meet, since one of them might have passed the common path hours before the other truck gets there. It is also possible that the adapted plan is less fuel efficient than the default plan because of an increased fuel consumption during the merge phase with increased speed. In other words, the edge weight could be negative. These edges are not shown in the graph.

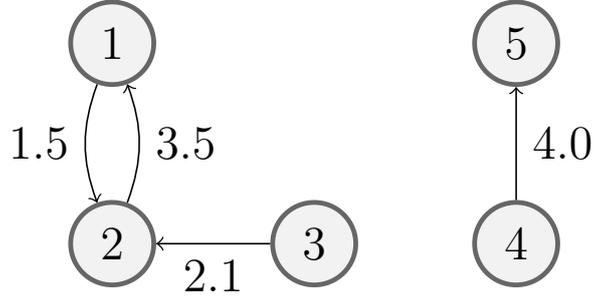


Figure 5: A simple coordination graph example for 5 trucks

Figure 5 shows schematically an example of a coordination graph. In this example, truck #1 would save 1.5 units of fuel if it forms a platoon by adapting to truck #2. Truck #3 would save 2.1 units of fuel if it adapted to truck #2.

More formally, using the notation from the publication by van de Hoef et al [36], a coordination graph is defined as a weighted directed graph  $\mathcal{G}_c = (\mathcal{N}_c, \mathcal{E}_c, \Delta F)$ . Recall that the elements of  $\mathcal{N}_c$  represent the trucks.  $\mathcal{E}_c \subseteq \mathcal{N}_c \times \mathcal{N}_c$  is a set of edges, and  $\Delta F : \mathcal{E}_c \rightarrow \mathcal{R}^+$  are edge weights, such that there is an edge  $(n, m) \in \mathcal{E}_c$ , if the adapted plan of  $n$  to  $m$  saves fuel compared to  $n$ 's default plan, i.e.,  $\mathcal{E}_c = (i, j) \in \mathcal{N}_c \times \mathcal{N}_c : \Delta F(i, j) > 0, i \neq j$ .

### 3.5 Clustering Solution for Coordination Graph

The next step is to select which trucks will run adapted plans and which will keep their default plans. We define a "coordination follower" as a truck that follows an adapted plan to catch up with another truck. We also define a "coordination leader" as a truck that stays on its normal speed while coordination followers adapt to it, to form a platoon. We want to select the coordination leaders in a way that maximizes fuel savings.

The result of the clustering algorithm is a set of coordination leaders. Each coordination follower will then pick the coordination leader that would be the most beneficial for it to follow. We can use this to generate the set of plans for our trucks. The coordination leaders will each follow their own default plan, moving towards their goal at a constant speed. The coordination followers will pick the leader with the highest edge weight for that follower.

$\mathcal{N}_c$  is the set of nodes in the coordination graph. Each representing a truck. We then define  $\mathcal{N}_l$  as our set of coordination leaders and  $E^-(v)$  as the total

---

fuel savings we get from picking a node  $v$  as a coordination leader.  $E^-(v)$  is the sum of all incoming edges that are not from nodes in  $\mathcal{N}_l$ .

If we were to pick only one coordination leader for the graph component  $\{1, 2, 3\}$  in Figure 5, we would get  $E^-(1) = 3.5$ ,  $E^-(2) = 1.5 + 2.1 = 3.6$ ,  $E^-(3) = 0$ . This means that picking truck #2 as our coordination leader would lead to the highest fuel savings. It is also easy to see that truck #5 in the other component of the graph would be a good choice for a coordination leader, since  $E^-(5) = 4.0$  while  $E^-(4) = 0$ .

The solid lines in Figure 6 represents followers following a leader. While the dashed line between node 2 and node 1 represents that node 2 could follow node 1, but since node 2 is a leader, it will not.

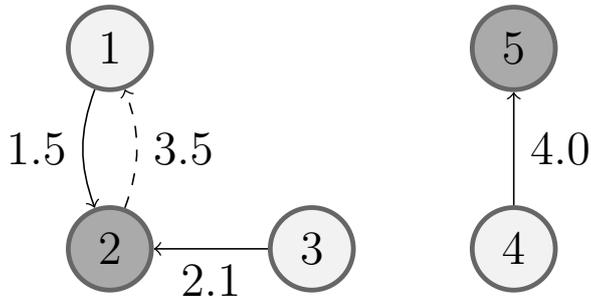


Figure 6: The optimal solution to the graph in Figure 5, with the coordination leaders marked in gray.

We now define a node  $n$ 's outgoing neighbors as nodes with an edge  $n$  to them. We can write this mathematically using the notation from [36]:  $\mathcal{N}_n^o = \{i \in \mathcal{N}_c : (n, i) \in \mathcal{E}_c\}$ .

We can now formally define what these clustering methods are trying to maximize: Given as input a coordination graph  $\mathcal{G}_c = (\mathcal{N}_c, \mathcal{E}_c, \Delta F)$  find a subset  $\mathcal{N}_l \subset \mathcal{N}_c$  of nodes that maximizes

$$\sum_{i \in \mathcal{N}_c \setminus \mathcal{N}_l} \max_{j \in \mathcal{N}_i^o \cap \mathcal{N}_l} \Delta F(i, j)$$

This is defined as *Problem 1* in [36]

The brute-force solution would be to enumerate all possible combinations of coordination leader and coordination follower, and see which one gives the highest total fuel savings. However, that algorithm would be  $O(2^n)$ . Even

---

for a small amount of trucks, this quickly becomes infeasible. In fact, it has been shown that calculating an optimal solution is NP-hard [35]. Instead, it would be preferable to come up with a heuristic that can give us good results, while still being able to scale to a large number of trucks.

Using the simulation tool developed for this project, we can evaluate several different clustering algorithms and see how they perform. In the subsections below we take a closer look at the clustering algorithms investigated.

### 3.5.1 Greedy clustering

Using the greedy clustering, we define the operation of "swapping" a node. Swapping a node means assigning it as a leader, except if it is already a leader, we assign it as a follower.

For each node, we keep track of how much fuel we would save if we decided to swap it from its current role to the other role. We then swap the node with the highest improvement, and update our list of fuel savings. We keep doing this until there are no more nodes that we can swap that would bring us higher fuel savings. For example, if the algorithm determines that swapping node  $X$  from follower to leader would lead to a fuel improvement of 5, but swapping  $Y$  from a leader role back to a follower is worth 7, the algorithm will swap  $Y$ . The swapping goes both ways.

This gives us no guarantees of how good the resulting clustering solution will be. Instead it can get stuck in a local maximum, leading to a sub-optimal solution.

---

**Algorithm 2** Greedy Clustering

---

```
1: procedure GETLEADERS(Nodes)
2:   Leaders  $\leftarrow \emptyset$ 
3:   NodeGains  $\leftarrow \{x : \text{Gain}(x, \text{Leaders}) \forall x \in \text{Nodes}\}$ 
4:   BestNode  $\leftarrow 0$ 
5:   while BestNode  $\neq -1$  do
6:     BestNode  $\leftarrow \text{GetHighestGainNode}(\text{NodeGains})$ 
7:     if BestNode  $\in \text{Leaders}$  then
8:       Leaders  $\leftarrow \text{Leaders} \setminus \{\text{BestNode}\}$ 
9:     else
10:      Leaders  $\leftarrow \text{Leaders} \cup \{\text{BestNode}\}$ 
11:      NodeGains  $\leftarrow \{x : \text{Gain}(x, \text{Leaders}) \forall x \in \text{Nodes}\}$ 
12:   return Leaders
13: procedure GAIN(Node, Leaders)
14:   total  $\leftarrow 0$ 
15:   if Node  $\in \text{Leaders}$  then
16:     BestLeader  $\leftarrow$  Leader neighbor X with highest
       FuelSavings(Node, X)
17:     total  $\leftarrow \text{total} + \text{FuelSavings}(\text{Node}, \text{BestLeader})$ 
18:     for neighbor  $\in \text{Graph}$  do
19:       BestNeighborLeader  $\leftarrow$  Leader neighbor X with highest
       FuelSavings(neighbor, X)
20:       CurrentFuelSavings  $\leftarrow \text{FuelSavings}(\text{neighbor}, \text{Node})$ 
21:       NewFuelSavings  $\leftarrow$ 
       FuelSavings(neighbor,
       BestNeighborLeader)
22:       total  $\leftarrow \text{total} + (\text{NewFuelSavings} - \text{CurrentFuelSavings})$ 
23:     else
24:       for neighbor  $\in \text{Graph}$  do
25:         CurrentFuelSavings  $\leftarrow$  FuelSavings(neighbor,
           LeaderOf(neighbor))
26:         NewFuelSavings  $\leftarrow \text{FuelSavings}(\text{neighbor}, \text{Node})$ 
27:         if NewFuelSavings  $>$  CurrentFuelSavings then
28:           total  $\leftarrow \text{total} + (\text{NewFuelSavings} - \text{CurrentFuelSavings})$ 
29:       return total
```

---

### 3.5.2 Random clustering

Random clustering works similarly to greedy clustering. It uses the same Gain method as defined in Algorithm 2. The difference is on line 6 in Algo-

---

rithm 3 where instead of picking the node with the highest gain, we pick a random node with positive gain.

This is likely to be worse than the greedy solution and is mostly investigated just for comparison with the greedy solution.

---

**Algorithm 3** Random Clustering

---

```
1: procedure GETLEADERS(Nodes)
2:   Leaders  $\leftarrow \emptyset$ 
3:   NodeGains  $\leftarrow \{x : \text{Gain}(x, \text{Leaders}) \forall x \in \text{Nodes}\}$ 
4:   RandomNode  $\leftarrow 0$ 
5:   while RandomNode  $\neq -1$  do
6:     RandomNode  $\leftarrow \text{GetRandomGainNode}(\text{NodeGains})$ 
7:     if RandomNode  $\in \text{Leaders}$  then
8:       Leaders  $\leftarrow \text{Leaders} \setminus \{\text{RandomNode}\}$ 
9:     else
10:      Leaders  $\leftarrow \text{Leaders} \cup \{\text{RandomNode}\}$ 
11:      NodeGains  $\leftarrow \{x : \text{Gain}(x, \text{Leaders}) \forall x \in \text{Nodes}\}$ 
return Leaders
```

---

### 3.5.3 Sub-modularity clustering

In addition to the greedy and random clustering methods, this thesis investigates the effectiveness of two algorithms by N. Buchbinder et al [14]. In their paper, they put forth two approximation algorithms for solving what they call the "Unconstrained Submodular Maximization" (USM) problem. These are Deterministic USM and Randomized USM. The algorithms are described below in pseudo-code described by N. Buchbinder et al [14].

---

**Algorithm 4** Deterministic USM

---

```
1:  $X_0 \leftarrow \emptyset, Y_0 \leftarrow \mathcal{N}$ 
2: for  $i = 1$  to  $n$  do
3:    $a_i \leftarrow f(X_{i-1} + u_i) - f(X_{i-1})$ 
4:    $b_i \leftarrow f(Y_{i-1} - u_i) - f(Y_{i-1})$ 
5:   if  $a_i \geq b_i$  then
6:      $X_i \leftarrow X_{i-1} + u_i$ 
7:      $Y_i \leftarrow Y_{i-1}$ 
8:   else
9:      $X_i \leftarrow X_{i-1}$ 
10:     $Y_i \leftarrow Y_{i-1} - u_i$ 
return  $X_n$ 
```

---

---

**Algorithm 5** Randomized USM

---

```
1:  $X_0 \leftarrow \emptyset, Y_0 \leftarrow \mathcal{N}$ 
2: with probability  $i = 1$  to  $n$  do
3:    $a_i \leftarrow f(X_{i-1} + u_i) - f(X_{i-1})$ 
4:    $b_i \leftarrow f(Y_{i-1} - u_i) - f(Y_{i-1})$ 
5:   with probability  $a'_i/(a'_i + b'_i)^*$  do
6:      $X_i \leftarrow X_{i-1} + u_i$ 
7:      $Y_i \leftarrow Y_{i-1}$ 
8:   else (with the complement probability  $b'_i/(a'_i + b'_i)$ ) do
9:      $X_i \leftarrow X_{i-1}$ 
10:     $Y_i \leftarrow Y_{i-1} - u_i$ 
return  $X_n$ 
```

---

\*If  $a'_i = b'_i = 0$ , we assume  $a'_i/(a'_i + b'_i) = 1$ .

The (1/2)- and (1/3)-approximations<sup>1</sup> given by the two algorithms is only guaranteed if the clustering problem is a nonnegative submodular function. This thesis does not seek to prove that this is true for the clustering problem, so we do not have these guarantees. However, since the algorithms return a subset that maximizes a function, they should still be applicable for our situation.

---

<sup>1</sup>This means that the output of these algorithms will be at least half or one-third (respectively) as good as the optimal solution. It means the algorithms have guaranteed lower bounds, described as fractions of the optimal solution.

---

## 4 Dynamic Updating of Coordination System

There are multiple reasons why the coordination system needs to update the clustering solution as time goes on. It is possible that since the last time the system calculated the clustering solution, some trucks have split up from their platoons and are now ready to merge with another platoon. There will also be new trucks that register with the coordination system.

When a new assignment comes in, we will need to update the coordination graph described in Section 3.4. First of all, we need to add a node for the new truck. We also need to check if this new truck can platoon with anyone in the graph, and create edges for this in the graph. Figure 7 shows how an existing graph is updated with new edges when a new node is introduced.

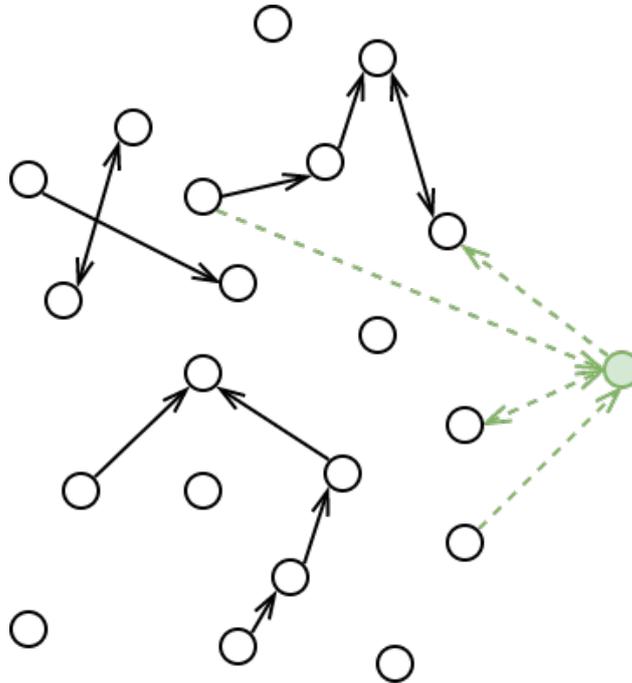


Figure 7: When a new node (shown in green) is added to the graph, new edges are created to the nodes that the new node can platoon with.

However, there is another aspect that needs to be handled. Since time has passed since the graph was last updated, the relationships between existing trucks will have changed. The current position of each truck will have changed, meaning the potential merge positions have changed. Trucks might also have been driving faster than normal since the last update. This means

---

that there is a possibility for new platooning opportunities that were not visible in the model before.

#### 4.1 Updating existing edges

Edges in the graph represent a plan where one truck follows another in a platoon, with the edge weight being the fuel saved by following the plan. The amount of fuel saved changes over time. An adapted plan generally consists of three phases. The merging phase where the trucks are catching up to each other, the platooning phase where the trucks are driving close together, and the split phase which happens after the trucks split up to go their separate ways.

During the merge, the follower will either be speeding up or slowing down to catch up with the leader. For the sake of simplicity we will focus on the case where the follower speeds up. This expends more fuel than driving at its normal speed, but the assumption is that the fuel saved while platooning will make up for it if the coordination follower follows the plan to the end. This means that if a platoon follower follows the adapted plan, the plan becomes increasingly attractive until it actually starts platooning. It then becomes less attractive the closer it gets to the end of the platooning phase.

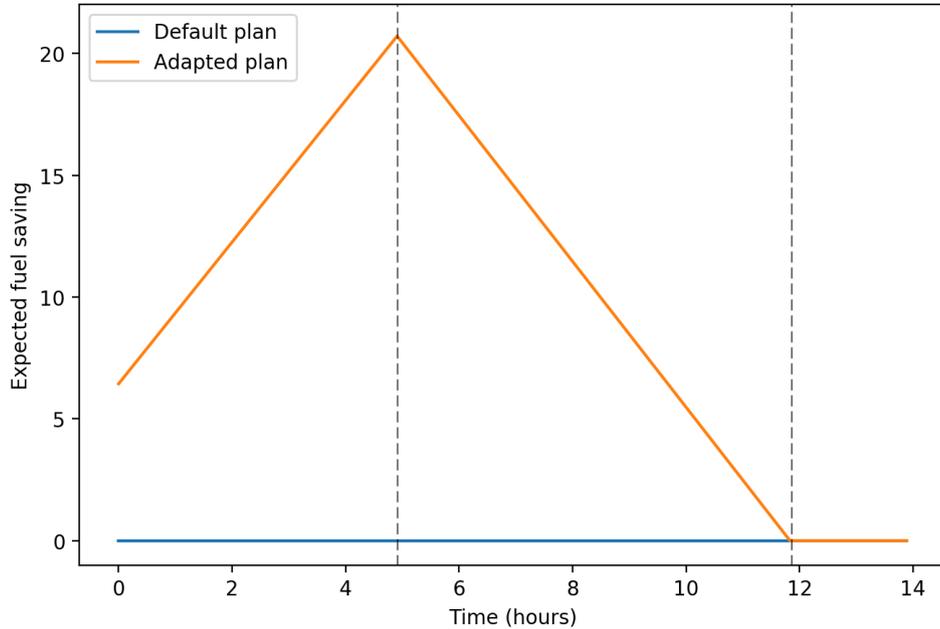


Figure 8: Showing an example of how the remaining expected fuel saving changes over time.

In Figure 8, we can see how the remaining expected fuel savings changes over time. At any given point, this graph shows how much fuel we expect to save if we continue with the plan, compared to if we follow the default plan. At  $t = 0$ , the amount of fuel saved is around 6, which is the total that we are expecting from this plan. Between  $t = 0$  and  $t = 5$ , the follower is merging. As more fuel is spent catching up to the leader, the expected improvement over the default plan increases, since the invested part is already gone. At  $t = 5$ , the graph has reached its peak. At this point, all the fuel savings are yet to come and the fuel investment has already been spent. Finally around  $t = 12$ , the platoon splits up, and the follower starts following its default plan instead.

By calculating a graph like this for each edge, we can quickly update the weights of the edges in the graph. Imagine the graph in Figure 8 represents a plan that corresponds to an edge in our coordination graph. After the plan is created at  $t = 0$ , the edge needs to be recalculated regularly. At  $t = 1$ , instead of recalculating the whole adapted plan, we can look at the graph to determine what the expected fuel saving is. This makes it easier to

---

recalculate plans and calculate how they compare to other plans.

## 4.2 Creating new edges between old nodes

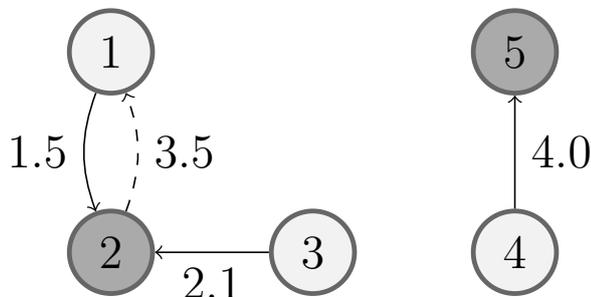
When more trucks are added to the graph, it is possible that some old pairs of nodes that did not have an edge between them previously should now have an edge between them. As mentioned in Section 3.4, the graph only contains edges where the adapted plan has positive fuel savings. It is possible that adapted plans with negative fuel savings will later become positive. For example, if a coordination follower would have to spend a lot of fuel just to get to the platoon, the expected fuel savings may be negative and the edge would not be included in the coordination graph. An hour later, it might have sped up multiple times to join other platoons, and is now further ahead than originally planned. The fuel needed to get to the platoon on time has already been spent, and at this point the fuel benefit from the platoon is higher than the fuel spent to get into the platoon.

There are also other reasons why two trucks that share a common path, would not have an edge in the graph. For example, it is possible that a coordination follower is slowing down as part of an adapted plan. When it slows down to follow another truck, it could become attractive as a coordination leader for another truck. Another example is a truck that has to drive at max speed for an hour, just to platoon for 30 seconds. The fuel saved from platooning for 30 seconds is not worth the fuel spent on driving faster to get to the merge point in time.

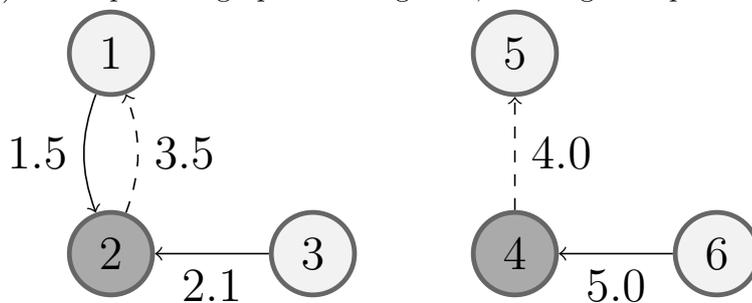
When initially computing pairwise plans we classify them into three categories: beneficial, potential and impossible. Beneficial plans are adapted plans that lead to positive fuel savings. These are represented by edges in the graph. Potential plans are adapted plans that do not lead to positive fuel savings or are not possible right now. However, they have been found to have the potential to be a beneficial plan in the future. These are not present in the graph but are recalculated each time a new node is added to the graph to see if they have become beneficial. The impossible category is for pairs of trucks that will never be able to platoon. This can happen if the two trucks do not have an overlapping path or if one or both of the trucks have already passed the overlapping path. If we come to the conclusion that an adapted plan between two trucks will never be beneficial, we will remember it and never recalculate the adapted plan between those two trucks again.

---

### 4.3 Updating the clustering solution



(a) Initial platoon graph from Figure 6, showing the optimal solution

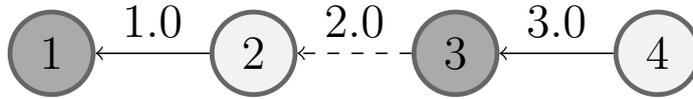


(b) The addition of node 6 changes the optimal solution.

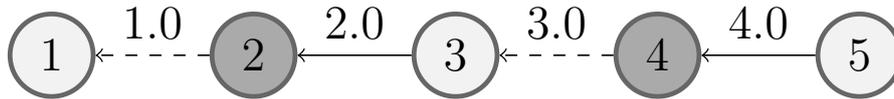
Figure 9: When we add a new assignment, 6, to the graph, one of the platoon leaders change from 5 to 4.

The easiest way to find the updated clustering solution is to recalculate the platoon graph and the clustering solution from scratch every time a new assignment shows up, using one of the algorithms in Section 3.5. This is inefficient since the addition of a new assignment results in a solution that can be very similar to the previous solution. That two clustering solutions are similar means that there is a large overlap in the set of coordination leaders for the two solutions. As shown in Figure 9, the addition of a new node only impacts the right component of the graph, not the left one.

However, finding out the impact of a new assignment is tricky. Even a small change can affect the entire clustering solution. As shown in Figure 10, the addition of a new node impacts the entire clustering solution. Every coordination follower becomes a coordination leader and ever coordination leader becomes a coordination follower. While this is a constructed example to show how the impact can propagate indefinitely, it is still relevant to show that there is no easy way to calculate the full impact.



(a) Initial platoon graph, with #1 and #3 as platoon leaders.



(b) When a new assignment comes in, it completely changes the optimal solution.

Figure 10: An example of how the impact of adding a new assignment propagates through the platoon graph.

Another solution is to allow the new assignments to be followers. This would let us keep the previous optimal solution and just pick the best leader for the new assignment to adapt to. However, this runs into issues when all the original leaders reach their destinations. In a system that ideally keeps running for years, there needs to be a constant influx of both leaders and followers. Adding all new assignments as followers means we will soon start running out of leaders to follow.

In the end, a hybrid solution was chosen. Looking specifically at the clustering algorithms described in Section 3.5, it is sometimes possible to start from the previous clustering solution. Both the greedy and random clustering methods start with an empty set of leaders. They then keep selecting nodes to swap between leader and follower. If we give these methods a non-empty set of leaders from the previous iteration, it will keep us from having to rebuild the solution from scratch. For the submodular clustering method, it is not as obvious how to start from a previous solution. Because of this, it recalculates the clustering solution from scratch every time.

## 5 Simulation Setup

To investigate how a dynamic system would work and scale, a simulation tool was constructed to try out how different methods and ideas perform. A flow chart of the simulation code is shown in Figure 11. After starting and generating an empty coordination graph, the simulation code goes into a loop until there are no more trucks to simulate. The loop first takes in any new truck assignments that have started since the last update. It updates all the trucks, calculating their new positions. It then updates the coordination

---

graph using the methods described in Section 4. It then calculates a new clustering solution using whatever clustering method was specified. Finally, it updates the plan of all active trucks and updates the speed history accordingly.

The updating of the coordination graph is done according to the method described in Section 4.1 and 4.2. The details on how to find a new clustering solution in the coordination graph is explained in more detail in section 4.3. The concept of a speed history is introduced in Section 5.3. Finally, the idea of verifying the results is discussed in Section 5.4.

## 5.1 Truck assignments

First off, we need to decide how to generate our truck assignments which is the input to the simulation. It is possible to take a manually constructed set of truck assignments to test on, but for our purposes, we want to get a general sense of how the system performs. To do this, we will generate assignments randomly.

Each assignment needs a start position, a goal position, a start time, and a deadline. The start and goal positions are randomly sampled from a population density map of Sweden, shown in Figure 12. The start time is randomly picked with a uniform distribution in a 48 hour start interval. The actual routes that the trucks drive in the simulation are shown in Figure 13. The routes are calculated using Open Street Map (OSM) [3]. This map had the shorter routes removed to mainly keep the long highways that we are interested in. Specifically (using OSM terminology), only motorway, primary, and secondary roads were kept, along with their corresponding link roads.

The average length of truck routes after sampling from the density map ended up being around 540 kilometers. For a sense of scale, the distance between Stockholm and Gothenburg is around 470 kilometers.

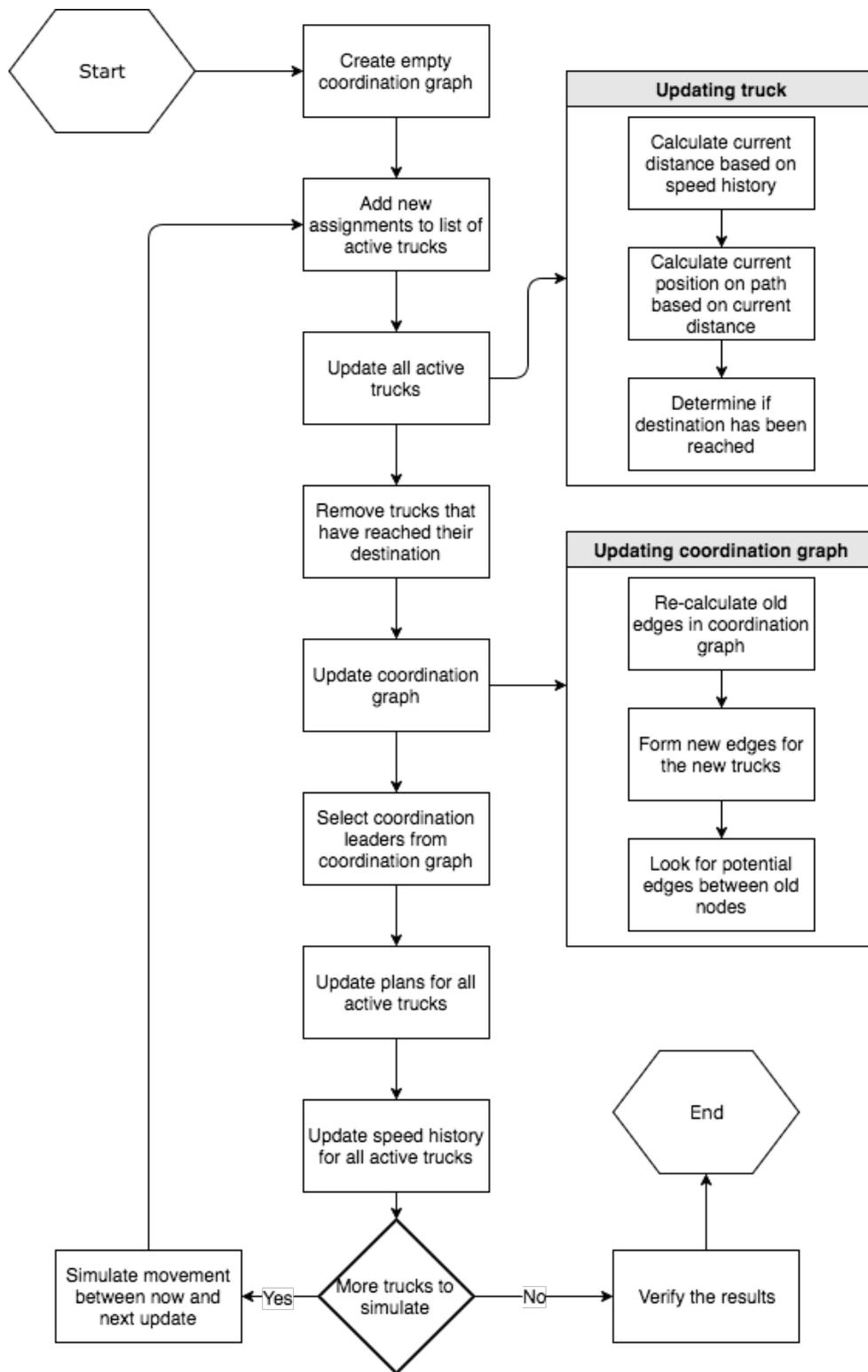


Figure 11: A flow chart of the full simulation behavior.

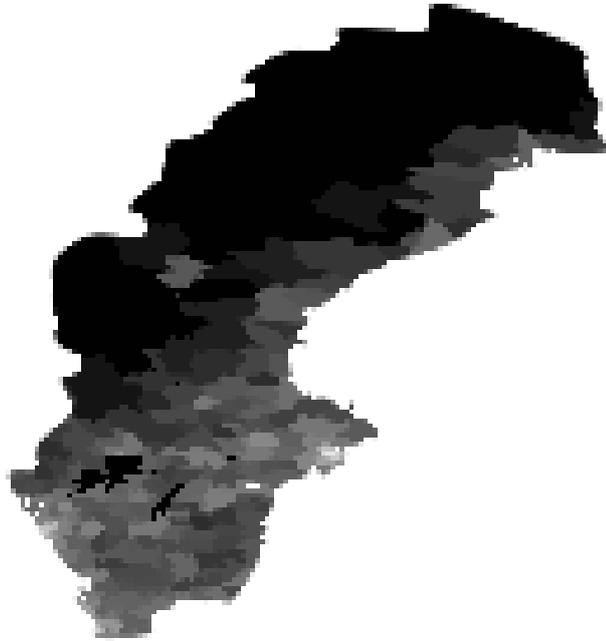


Figure 12: Density map used for sampling random truck assignments [32].



Figure 13: Map showing the routes driven by the trucks in the simulation [2].

---

## 5.2 Assignment Start Interval

To evaluate the performance of the system, we want to get a correct view of how the system performs on average. In other words, imagine that the system runs continuously for a long time. To grade the performance, we would like to know what the fuel consumption is on a randomly picked day.

The thing to note is that it takes a while for the system to reach a steady state. As trucks finish their assignments, new trucks will start driving, keeping the total number of driving trucks close to constant. However, when the system first starts up, there will be a growing amount of trucks as more and more trucks are added to the system, since no trucks have finished their assignments yet. As some of the first trucks start finishing their routes, it will converge towards a more steady number with small fluctuations around the average due to randomized start times and deadlines. A graph of active trucks can be seen in Figure 14. It is taken from a simulation with 800 trucks over 48 hours. A similar graph for the big dataset used for the main results is shown in Figure 18.

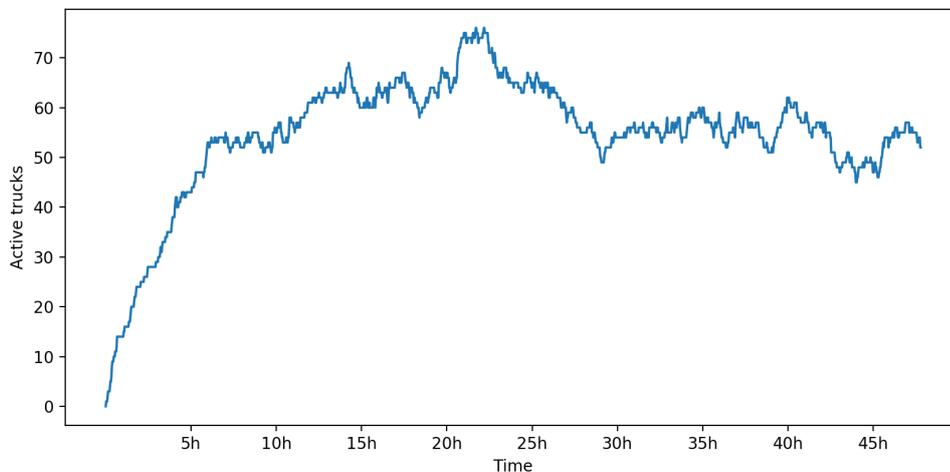


Figure 14: Showing the amount of active trucks over time.

As shown in the figure above, the amount of active trucks converge after around 10 hours. Because of this, we will run simulations with a 48 hour start interval. This means that all the trucks in the simulation will start within a span of 48 hours. For evaluation, we will only consider the trucks that start in the second half of the interval. This means the results are from 24 hours of simulation. The reason we limit ourselves to half of the result is

---

to avoid the transient effects of the initial startup when the amount of trucks in the system is still increasing quickly. The reason this was done instead of initializing the simulation with multiple running trucks was to make sure the previously existing trucks were simulated correctly. Starting the simulation with trucks that are already halfway through their assignments seemed like something that might not fully model the real scenario.

### 5.3 Generated Data During a Simulation Run

This section looks at the data gathered during a simulation, detailing exactly what the trucks did during this time. The results in Section 6 are based on this data, not on predictions made during the simulation. In this case, we are most interested in how the trucks drove. The most important property that the system changes is the speed at different points, so for each truck, a speed history is collected.

The speed history is a list of speed changes, where each speed change contains four elements:

- Start time
- End time
- Speed
- Which other truck, if any, that the truck was platooning with.

These speed changes should cover the entire trip. This means it is possible for every simulated truck to reconstruct the history and see how fast it was going at any point in time and when it was platooning.

An example of what a speed history can look like is shown in Figure 15. This shows exactly what the speed was at any point during a trucks journey. The orange parts show when the truck was driving as a coordination follower. It started by driving at 80 km/h, but quickly sped up to 90 km/h to meet up with a coordination leader. After driving in the platoon for a short time, it decided to speed up again to meet up with a different coordination leader. It drove in this platoon for over three hours before splitting up. At the end, it was able to reduce its speed to around 71 km/h to save fuel and still make it to the destination on time.

In addition, the simulation also generates a plan history, detailing exactly

---

what plan each truck was following at any given time. The details for a specific adapted plan contains information on:

- How much fuel is consumed by following the plan to the end.
- How much fuel would be consumed by instead following the default plan.
- When/Where the truck will merge with the platoon.
- When/Where the truck will split from the platoon.
- When the truck will make it to the destination.
- The speed the truck needs to drive at on its way to the merge point.
- The speed the truck should drive at while in the platoon.
- The speed the truck needs to drive at after leaving the platoon.
- The coordination leader that the truck is following as part of the adapted plan.

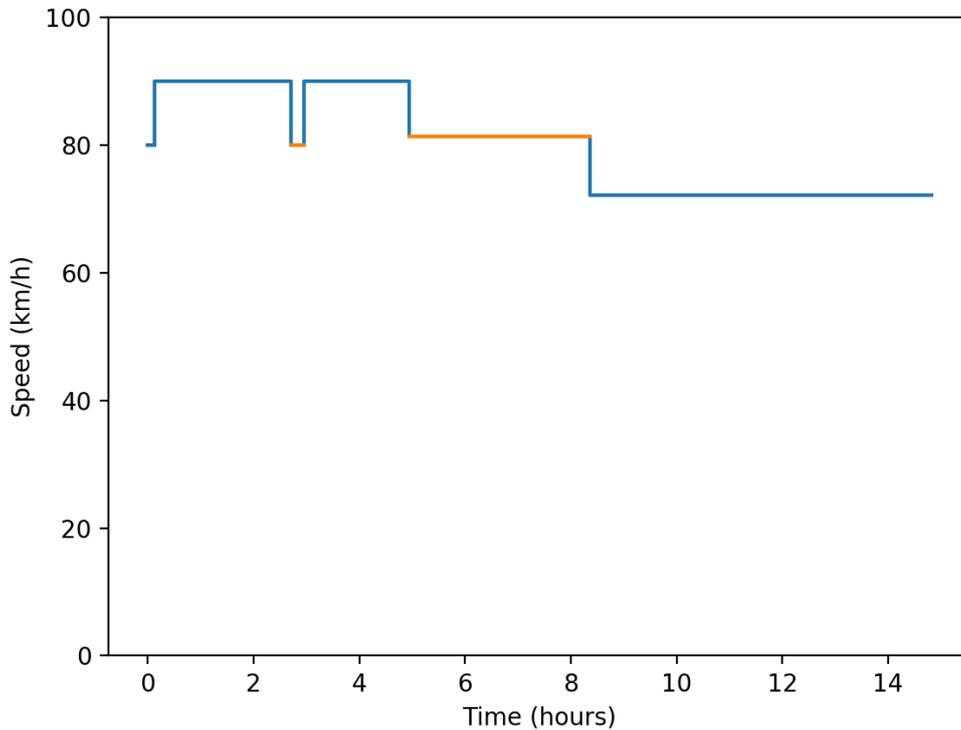


Figure 15: A truck's speed history, showing its speed at every point in time.

## 5.4 Testing of the Simulation Results

In a complex system like this, it is important to know that the results are correct. To help with this, a tool was written to test the results at the end of each simulation.

This tool performs the following tests:

- **Speed history reaches destination:** Check if the total distance driven according to the speed history is enough to let the truck reach the destination from the start point. It should also not be longer than necessary to reach the destination.
- **Speed history start time:** Check if the speed history starts at the same time as the truck.
- **Speed history end time:** Check if the speed history ends before the truck's deadline.

- 
- **Speed history is continuous:** Check if there are no gaps in the speed history. Each speed change in the history should start where the previous change ended and there should be no overlap.
  - **Speed history arrival time:** Check if the arrival time of the truck is at the same time as the speed history ends.
  - **Not leading and following simultaneously:** Check if there is no truck that is both a platoon follower and a platoon leader at the same time.
  - **Platoon match:** Check that for each plan to platoon, both trucks made it to the merge point at the same time and split at the same time.

Checking all of these things increases the confidence in the result. Even if there are problems in the implementation, the solution given is plausible in the sense that it is realistic. This makes the fuel consumption reduction into a lower bound of what you can expect from the system. There could be a better solution with a correct implementation, but the solution produced is also feasible.

As an example, early in development there was a bug that caused the system to overestimate the fuel consumption reduction because many trucks were missing part of their speed history. Since the speed history was incomplete, the system assumed that no fuel was spent during this missing period. This kind of bug was caught by the "Speed history is continuous" test.

## 5.5 Simulation Parameters

The following parameters are used to configure the simulation:

- **Minimum speed:** This is the minimum speed that the trucks drive. The minimum speed is set to 70 km/h.
- **Maximum speed:** This is the maximum speed that the trucks drive. The maximum speed is set to 90 km/h.
- **Nominal speed:** This is the normal speed that trucks drive if they are in a platoon or if they are not forming a platoon with another truck. The nominal speed is set to 80 km/h.
- **Minimum intersection length:** Trucks only form a platoon if the common path between a pair of trucks is at least 5 kilometers.

- 
- **Horizon:** The horizon describes how far in advance the system gets to know about new truck assignments. If the horizon is set to 10 minutes, the truck registers in the system 10 minutes before it starts its journey. Increasing this value allows the system to consider new assignments earlier in the planning process. For most of the simulations, this value was set to 0 seconds, to show the worst-case scenario. However, in Section 6.3, it shows how different horizon lengths impact the results.
  - **Interval:** This describes how often the plans will be recalculated. The plans can be recalculated each time a new assignment is registered, or at regular intervals. For the simulations, the interval was set to 10 minutes. However, in Section 6.4, we investigate how different interval lengths impact the results.

## 5.6 Simulation Fuel Model

For the simulation, we use a linear fuel model derived from [10]. When a truck is not platooning, the fuel consumption per distance traveled at a certain speed is calculated using

$$f(v) = 4.80 \cdot 10^{-5} + 8.42 \cdot 10^{-6} \cdot v$$

. When platooning, it is calculated using the formula

$$f_p(v) = 8.54 \cdot 10^{-5} + 5.05 \cdot 10^{-6} \cdot v$$

By inserting the normal platooning speed mentioned in the previous section, we can calculate the fuel saving,  $s$ , for a platoon follower driving at the nominal speed of 22.2 m/s (80 km/h).

$$\begin{aligned} s &= f_p(22.2)/f(22.2) = \\ &= (8.54 \cdot 10^{-5} + 5.05 \cdot 10^{-6} \cdot v)/(4.80 \cdot 10^{-5} + 8.42 \cdot 10^{-6} \cdot v) \approx \\ &\approx 0.84054 \end{aligned}$$

This means that platooning gives around 15.9% fuel savings compared to driving like normal.

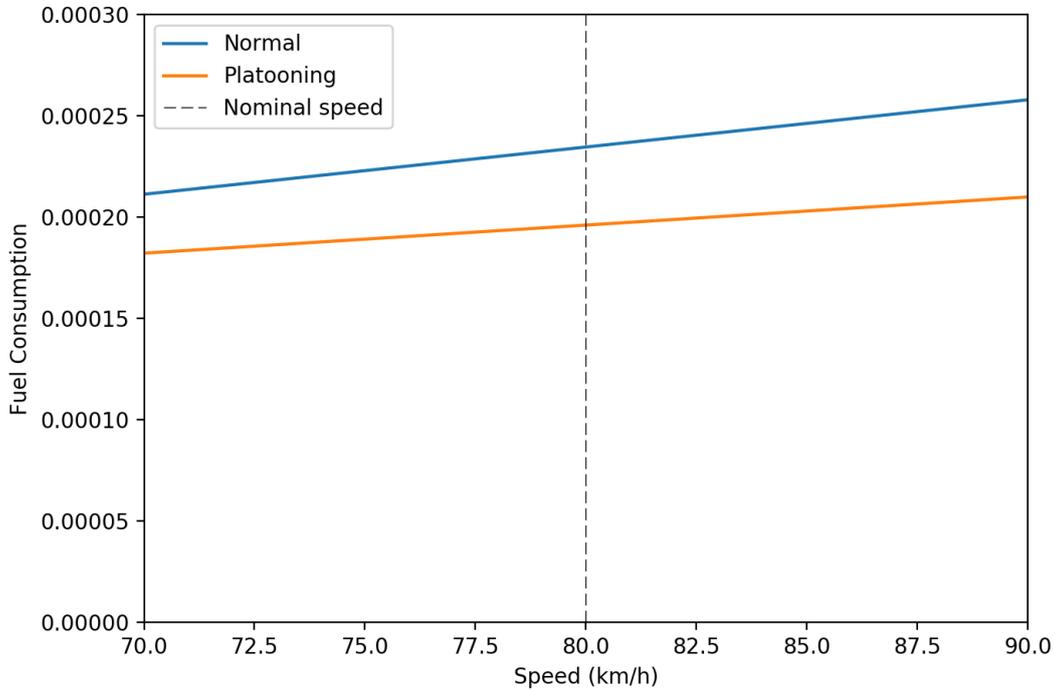


Figure 16: Showing the graph for the two linear fuel models.

Additionally, since we are looking at it from a global perspective, it's not important exactly which truck saves fuel. This means we're assuming that the coordination follower saves fuel, not the platoon follower. This means that the truck that runs the adapted plan to catch up with another truck will be considered to save fuel, even if it ends up in the front of the platoon.

## 5.7 Size of Test Datasets

The results in Section 6 come from running the simulation software on five datasets. Each dataset contains 10,000 truck assignments and a starting interval of 48 hours. The final result was the average of the 5 datasets.

In 2016, Sweden's truck traffic consisted of a total of 39,619,000 truck assignments [6]. This means an average of around 100,000 trucks per day, which means that 10,000 trucks in 48 hours is about 5% of Sweden's normal traffic load. Due to the large number of hyper-parameters such as horizon, interval length, and different clustering methods as well as trying to get an average of several different datasets, a decision was made to limit the test

---

size to allow sufficient time for simulation. This is still a realistic size for the dataset, since it is unlikely that every truck in Sweden would be equipped with the hardware necessary for platooning.

## 5.8 Hardware and Software for Simulation

**Hardware:** The simulation machine contains a Intel Core i7-2600k CPU @ 3.4GHz and 4x4GB DDR3 1333MHz RAM.

**Software:** The simulation machine was running Python 2.7.9 on Debian 8.2.

# 6 Simulation Results

In this section we introduce the results from the simulation. The section shows a lot of different results including the fuel saved by the different clustering methods and the effect of changing parameters such as the horizon and update frequency. It also shows some interesting simulation metrics such as the percentage of trucks that are platooning at any given time and the average plan durations.

## 6.1 Clustering methods

First let us look at a baseline for the four different clustering methods described in Section 3.5. The greedy, the random, and both the deterministic and randomized version of the USM (Unconstrained Submodular Maximization) clustering methods. The result in Figure 17 is taken from running each of the clustering methods using the simulation methods specified in Section 5.

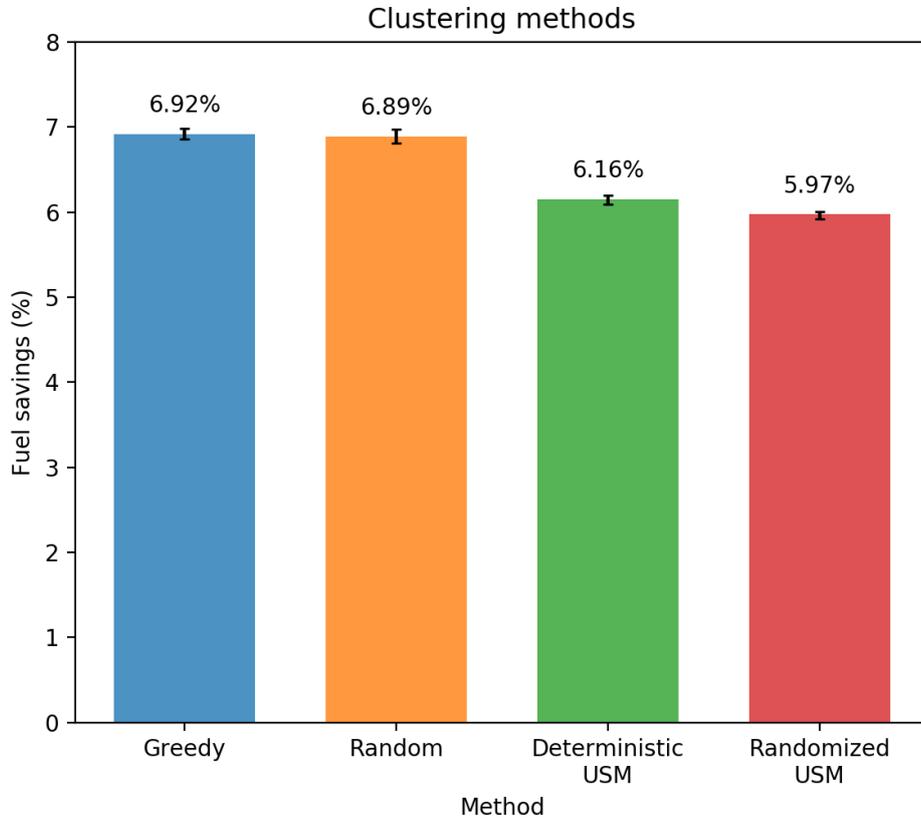


Figure 17: Showing the percentage fuel saving for each clustering method. The black lines at the top of each bar represents the spread of the result, from the minimum value achieved to the maximum.

The greedy algorithm ends up with the best average fuel saving of 6.92%. Both of the USM algorithms have significantly worse fuel savings than the greedy and random algorithms. The black bar shows the spread between the minimum and maximum values from the simulation. From the figure, this spread seems to be very low. The highest spread comes from the random method, which makes sense since it is based on picking a random node at each step.

It is reasonable that the greedy method ends up higher than random, since greedy picks the best leader at any given time, instead of a random one. It is a bit unexpected to see the USM methods perform significantly worse than random and greedy.

---

To shorten the necessary simulation time, the rest of the results are solely focused on the greedy clustering method, since it seems to perform better than the other algorithms. Unless stated otherwise, the rest of the results came from using the greedy clustering method.

In addition to the average fuel savings of the different methods, another interesting aspect to investigate is how similar the solutions provided by the different methods are. To calculate this, the percentage of leaders that the solutions had in common were aggregated over the entire duration of the simulation. The result can be seen in Table ??.

	Greedy	Random	Deterministic USM	Randomized USM
Greedy	100%			
Random	47.40%	100%		
Deterministic USM	42.43%	43.77%	100%	
Randomized USM	42.43%	40.92%	50.60%	100%

Table 1: Percentage of clustering solutions that are the same. Upper triangle is blank since the table is symmetric.

The similarity between two methods  $m1$  and  $m2$  was calculated by taking the average of

$$(leaders(m1) \cap leaders(m2)) / (leaders(m1) \cup leaders(m2))$$

for every step in the simulation.

The table above shows that the clusterings solutions from the different methods differ quite a lot. Just like in Figure 17, we can see that the two USM methods are closer to each other than to the rest. The same is true for greedy and random. This makes sense considering they share a common algorithmic base.

## 6.2 Steady State of Active Trucks

In Section 5.2 we investigated how to set the start interval to get a look at an average day. To show this, we used the number of active trucks over time in a small simulation, as shown in Figure 14. In this section we are showing

---

the same type of graph for our five large datasets. This is shown in Figure 18 below.

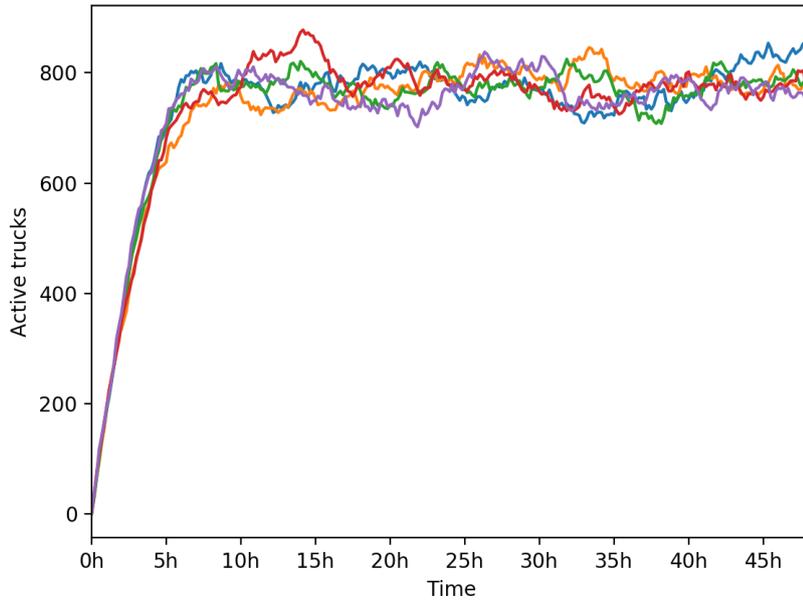


Figure 18: Showing how the active number of trucks changes over time for the five different datasets.

### 6.3 Effect of Increasing Horizon

Next, let us investigate how the horizon, as explained in Section 5.5, affects the fuel saving. The data in Figure 19 shows how the greedy clustering method performs with different horizon lengths ranging from zero to two hours. For each horizon length, the average fuel saving was taken from the five datasets presented in Section 5.7.

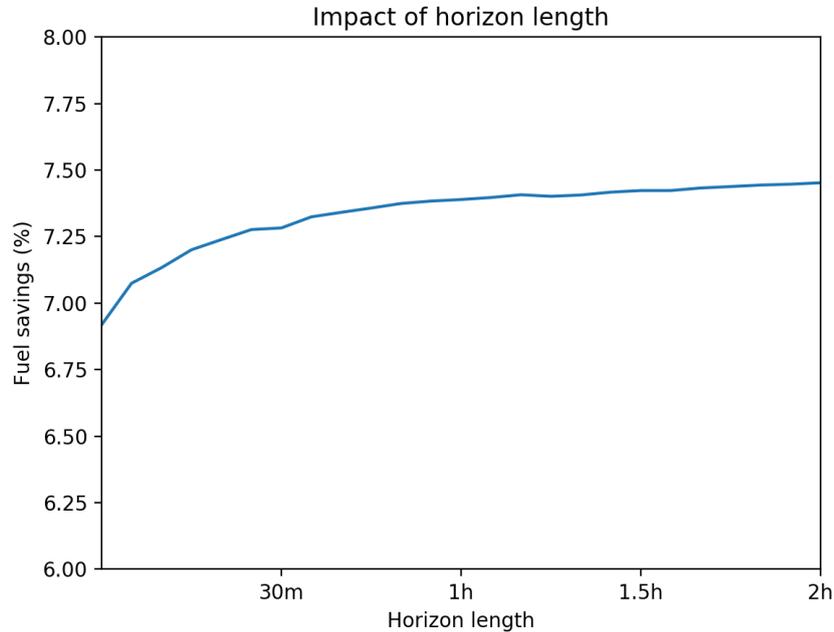


Figure 19: Showing how a longer horizon impacts the fuel saving for the different clustering methods. Note that the y-axis starts at 6.0 .

The data in Figure 19 is not very surprising. The further in advance that a system gets to learn about an upcoming assignment, the better it can plan for it and figure out a good solution. However, there is a limit to how much it can improve the results. The relative improvement quickly falls and it seems to converge towards fuel savings of approximately 7.4%, up from around 6.9%.

The results show that a longer horizon length leads to better results. However, the horizon length depends on the system and truck assignments you're getting. If the horizon length was set to one year, the trucks would have to register a year in advance to get the expected fuel savings. This value does not need to be static. In a real system, trucks should aim to register as early as possible. This means that it's better if more trucks that register early, but for any trucks that register 10 minutes before starting will still be able to join the system.

---

## 6.4 Effect of Increasing Update Frequency

When handling dynamic platooning the system has to re-calculate the platooning possibilities at a regular interval. One thing to investigate is how increasing or decreasing the frequency at which the system recalculates changes the results. A higher frequency means that the system has to recalculate more often while a lower frequency means that new trucks will have to wait longer before a plan is calculated. This data is shown in Figure 20. The implication that this has on the maximum capacity is discussed in Section 6.8.

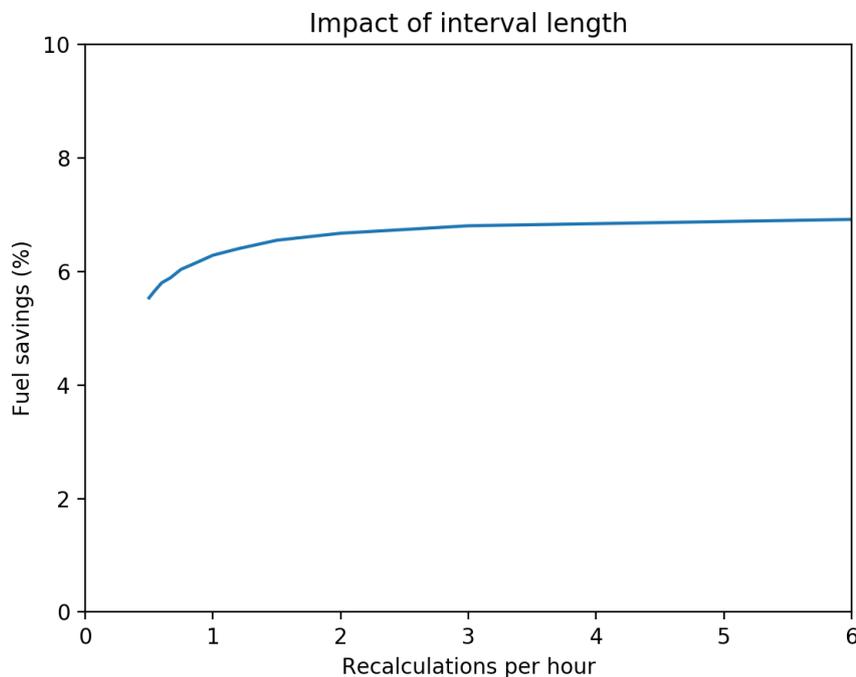


Figure 20: Showing how the update frequency for re-calculations impact the fuel savings.

Figure 20 looks reasonable since the more often the plans are re-calculated, the better. Recalculating often is good for multiple reasons. Not only does it take new trucks into account that have not yet received a plan. It also calculates if there are new better plans for the existing trucks. The graph does not go all the way to the left, since 0 recalculations per hour would mean no fuel savings. The trade-off here is between fuel-consumption and calculation time. More updates per hour means less time to calculate each update. This is discussed more in Section 6.8.

---

## 6.5 Expected fuel consumption

We can also investigate and plot how the expected fuel consumption changes over time, which can be seen in Figure 21. The horizontal line represents the default state.

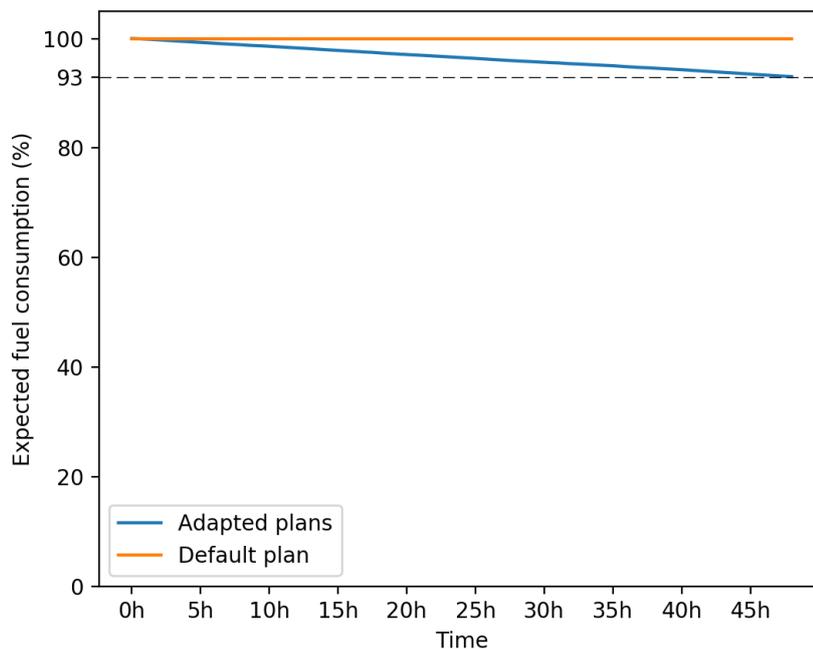


Figure 21: Showing how the expected fuel consumption changes over time as more truck assignments come in.

If every truck in the entire simulation drove without trying to platoon, the expected fuel consumption will not change since trucks will never change their plans. Therefore the orange line is horizontal.

The blue line shows the predicted fuel consumption at each point in time. Note that this does not just include active trucks. Instead it is the total of all trucks that are a part of the simulation dataset. As we get further and further along, more trucks are active and get taken into consideration when calculating the clustering, therefore getting a better predicted fuel consumption.

---

## 6.6 Plan Durations

Another aspect to be investigated is the average plan duration. This is the average time until a truck changes plans. This should not be confused with how fast they reach their destination, since trucks can be assigned to change from one plan to another, before the first plan is finished. The result in Figure 22 shows the average time that the trucks followed each plan.

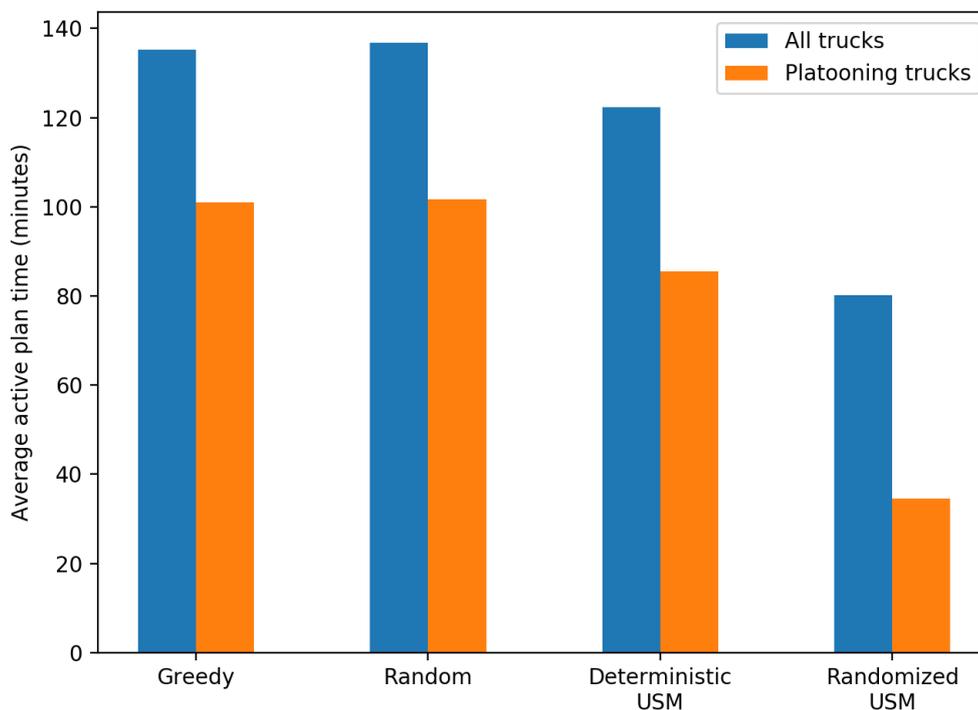


Figure 22: Showing the average plan length, both for all trucks and just trucks that platooned at some point.

For each method, two values are shown. The left column shows the average for all trucks. However, in any simulation there are trucks that do not form a platoon with any other truck and just drive on their own. The right column shows the average value when excluding any truck that just follows their default plan the entire way. The reason we're making this distinction is to try and also see the average for trucks that are actively part of the platooning process.

---

## 6.7 Percentage of platooning trucks

Similar to the previous section, it is interesting to investigate how the platooning ends up working in more detail. Figure 23 shows the percentage of active trucks that are followers in a platoon at any given point in time.

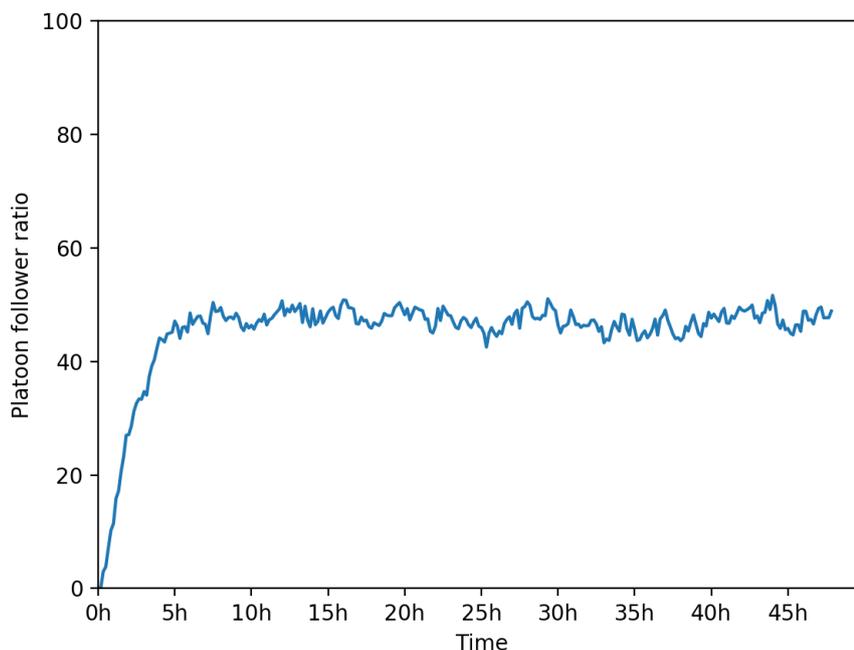


Figure 23: The percentage of trucks that are platooning.

The figure above shows that after the initial start of the system, the percentage of platoon follower quickly converges to around 40-50%. This is consistent with the fuel model in Section 5.6. If all trucks were following in a platoon all the time, the percentage of fuel saved would be around 16.5%. Instead, we achieve around 7% fuel savings on a global scale. This is around 42% of the possible fuel savings, and seems close to fit well with the average percent of platooning trucks, as shown in Figure 23.

## 6.8 Capacity for Real-time System

The coordinating system has an update frequency which determines how often the graph and clustering solutions are recalculated. An update frequency of every 10 minutes was chosen for most of the simulation results.

---

In a real-time system, this means that the system has 10 minutes to finish the calculations before it has to start a new calculation. The time it takes to run this calculation is based on the active trucks that are registered in the system. In Figure 24 the relationship between calculation time and active trucks is investigated.

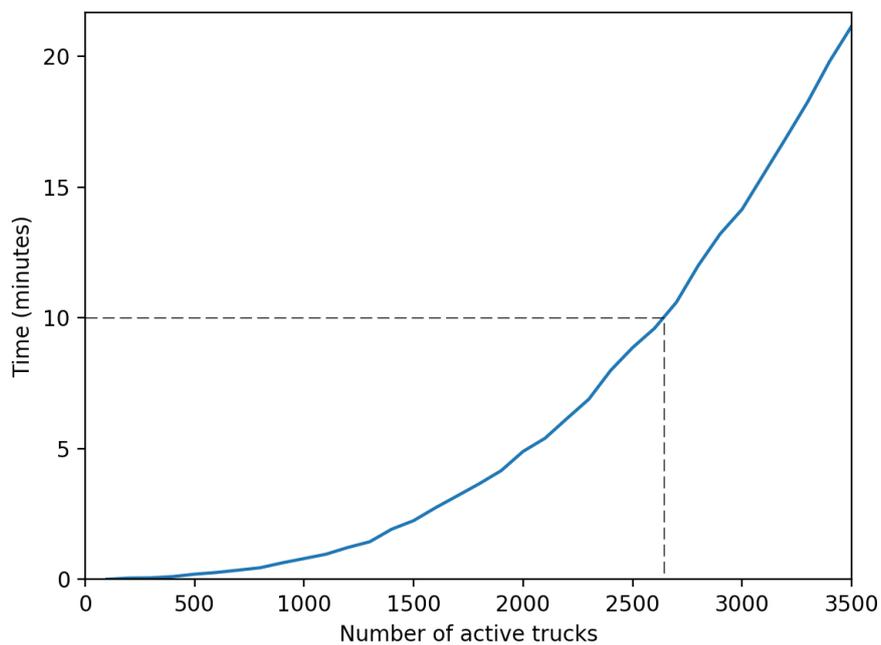


Figure 24: Time for one recalculation depending on amount of active trucks.

Looking at the figure above, we can see how many active trucks we can expect to be able to handle in a system with an update frequency of every 10 minutes. According to Figure 24 it is around 2600 active trucks.

To examine how many trucks the system can handle per day, we use the data from Figure 18. The simulation datasets contained 5000 trucks per day. From the figure we can see that they all have around 750 active trucks at any given time after the initial startup. This means that around 15% of the daily trucks are active on any given time. To confirm this, we can look back at Figure 14 which shows a similar graph for a simulation with 400 trucks per day. 15% of 400 trucks is 60, which matches what we see in the graph. Stepping back and relating this to the the maximum amount of active trucks that the

---

system can handle, we get that the system can handle  $2600/0.15 \approx 17,333$  trucks per day.

The maximum capacity is based on the software and hardware that ran the simulation. Running it on better hardware would likely result in a higher daily capacity. While this is less than 20% of Sweden's daily truck traffic, it shows what is possible with a 6-year old computer. With new, dedicated hardware, much higher maximum capacity should be possible.

## 7 Conclusion & Future work

This section concludes the thesis. It summarizes and draws conclusions from the simulations results. It also looks at areas that require more research and work to investigate further.

### 7.1 Conclusion

This thesis evaluated large-scale simulations for coordination of platoons. New theory was introduced to handle new assignments and the changing of existing assignments. The thesis sought to answer questions about what level of fuel savings one can expect from this system, as well as how suited the system would be for a real-time system. With fuel savings of 7%, this work shows promising potential for a global coordination system.

Even though the implementation of the simulation engine was not optimized for speed, it was still able to simulate a realistic amount of trucks in a reasonable amount of time. Results show that it should be able to simulate around 17,000 trucks per day in real-time. This work also shows that the result can be improved even further, depending on the parameters and computational performance of the system. The more often the plans are recalculated and the further in advance the system gets to know about future assignments, the better fuel savings the users can expect.

Right now, the simulation considers the calculations to be instantaneous, even though they are not. This means that all trucks pause while the system is recalculating plans. In the real world, this will not be true. If it takes a minute for the system to calculate the plans, the plans will be slightly off by the time the calculation finishes. There are several questions to answer here. How much does this affect the results and is it possible to compensate for it by extrapolating the trucks' current position based on historic data of how

---

long the calculations will take?

We also made some other simplifications in order to investigate systematic effects without writing overly complicated code. First of all, the system assumes that all roads have the same speed limits. Since the map mostly contains of highways (as described in Section 5.1), this is not as bad as it may seem. However, varying highway speeds is another area that could lead to disturbances.

The system also assumes that changing the speed is instant. In reality, a truck accelerates slowly. A medium truck can have an acceleration speed of  $0.74 m/s^2$  [39], meaning it can take around 3-4 seconds to accelerate from the nominal speed (80 km/h) to the maximum speed (90 km/h). This could lead to trucks being late to their merge point, which could impact the fuel savings. This could potentially be handled by changing the way adapted plans are created. The code would have to take the acceleration into account. The speed history would also have to be modified to show the gradual change in speed as the truck accelerates and decelerates.

## 7.2 Future work

The method described in this thesis already achieves significant results. However, in addition to the limitations mentioned in Section 7.1, there are other areas that should be investigated further. Traffic, accidents, and road and weather conditions, are all things that can impact the calculations. How much, and mitigations for this problem are things worth investigating before implementing this in the real world. This section looks at other areas that could be improved for further improvements of the whole system.

Right now, the whole system relies on adjusting speed. However, it is possible that the results can be improved by looking at different paths. While a truck might initially follow the shortest path between A and B, it might benefit from switching to a slightly longer path, if there are a lot of other trucks to platoon with on the alternate path. This would require major rework of the way the adapted plans are calculated to make the algorithm consider multiple different paths with potentially multiple different speed profiles. A potential simplification could be to only consider the shortest path that intersects the path of the coordination leader, that way a lot of the existing code and theory could be reused.

Another interesting topic is how the clustering solution changes when new

---

assignments come in. After a new assignment comes in, it is possible that a majority of the new clustering solution is the same. It is also possible that the entire clustering solution is different, as shown in Figure 10. While it is possible to start the greedy and random algorithms from the previous solution, it would be interesting to calculate exactly how much of the solution is affected. This would mean other clustering algorithms that can not start from the previous solution can be optimized to only recalculate the part of the solution that has been impacted by the new assignments.

The current dataset sizes are realistic, but Section 6.8 shows that the maximum capacity is still less than 20% of Sweden's truck traffic. While it is possible that there will be multiple competing providers of this platoon coordination service, it would be interesting to see how much fuel can be saved if all the trucks in Sweden were using the same system.

If the system could be improved to handle hundreds of thousands of trucks instead of tens of thousand, we would get a better idea of the kinds of fuel savings we could expect. To achieve this, the system should probably be rewritten in a language that allows for further performance optimizations, like C++ [8]. It should also be possible to make the system multi-threaded and distribute the workload. Especially the task of calculating the pair-wise adapted plans should be trivial to parallelize which should greatly increase performance and therefore the maximum capacity of the system.

## 8 References

- [1] COMPANION Project.  
<http://www.companion-project.eu/>.
- [2] Mapbox.  
<https://www.mapbox.com/about/maps/>.
- [3] Open Street Map.  
<https://www.openstreetmap.org>.
- [4] Scania Platooning.  
<https://www.scania.com/group/en/2017-scania-takes-lead-with-full-scale-autonomous-truck-platoon/>.
- [5] Slipstream effect.  
<https://www.eutruckplatooning.com/news/397567.aspx>.

- 
- [6] Swedish road goods transport.  
[https://www.trafa.se/globalassets/statistik/vagtrafik/lastbilstrafik/2017/lastbilstrafik\\_2017\\_kvartal2.pdf?](https://www.trafa.se/globalassets/statistik/vagtrafik/lastbilstrafik/2017/lastbilstrafik_2017_kvartal2.pdf?)
- [7] Climate Action Tracker - China, 2016.  
<http://climateactiontracker.org/countries/china.html>  
Accessed: 2017-03-09.
- [8] Z. Alomari, O. E. Halimi, K. Sivaprasad, and C. Pandit. Comparative studies of six programming languages. April 2015.
- [9] F. An and M. Ross. A model of fuel economy and driving patterns. In *SAE Technical Paper*. SAE International, 03 1993.
- [10] B. Besselink, V. Turri, S. H. van de Hoef, K. Liang, A. A. Alam, J. Mårtensson, and K. H. Johansson. Cyber-physical control of road freight transport. *CoRR*, abs/1507.03466, 2015.
- [11] C. Bonnet and H. Fritz. Fuel consumption reduction in a platoon: Experimental results with two electronically coupled trucks at close spacing. In *SAE Technical Paper*. SAE International, 08 2000.
- [12] C. Bonnet and H. Fritz. Fuel consumption reduction in a platoon: Experimental results with two electronically coupled trucks at close spacing. In *Future Transportation Technology Conference & Exposition*. SAE International, aug 2000.
- [13] F. Browand, J. McArthur, and C. Radovich. Fuel saving achieved in the field test of two tandem trucks. June 2004.
- [14] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A tight linear time  $(1/2)$ -approximation for unconstrained submodular maximization. pages 649–658. IEEE, October 2012.
- [15] California PATH. Partners for Advanced Transportation Technology.  
<http://www.path.berkeley.edu/research/automated-and-connected-vehicles/truck-platooning>.
- [16] J. Carbaugh, D. N. Godbole, and R. Sengupta. Safety and capacity analysis of automated and manual highway systems. *Transportation Research Part C*, 6(1):69–99, 1998.
- [17] R. Dang, J. Wang, S. E. Li, and K. Li. Coordinated adaptive cruise control system with lane-change assistance. *Intelligent Transportation Systems, IEEE Transactions on*, 16(5):2373–2383, October 2015.

- 
- [18] Q. Deng and X. Ma. A fast algorithm for planning optimal platoon speeds on highway. *IFAC Proceedings Volumes*, 47(3):8073 – 8078, 2014. 19th IFAC World Congress.
- [19] EU Commission. Strategy for reducing heavy-duty vehicles’ fuel consumption and co2 emissions, 2014.  
<http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52014DC0285>  
Accessed: 2017-03-09.
- [20] F. Farokhi and K. H. Johansson. A study of truck platooning incentives using a congestion game. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):581–595, April 2015.
- [21] H. Fritz. Longitudinal and lateral control of heavy duty trucks for automated vehicle following in mixed traffic: experimental results from the chauffeur project. volume 2, pages 1348–1352. IEEE Publishing, 1999.
- [22] O. Gehring and H. Fritz. Practical results of a longitudinal control concept for truck platooning with vehicle to vehicle communication. pages 117–122. IEEE Publishing, 1997.
- [23] C. Kammer and K. Johansson. Coordinated heavy truck platoon routing using global and locally distributed approaches, 2013.
- [24] N. Kinnear, S. Helman, C. Wallbank, and G. Grayson. An experimental study of factors associated with driver frustration and overtaking intentions. 79:221–230, June 2015.
- [25] J. Larson, K. Y. Liang, and K. H. Johansson. A distributed framework for coordinated heavy-duty vehicle platooning. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):419–429, Feb 2015.
- [26] S. Lee and D. Ahn. Design and verification of driver interfaces for adaptive cruise control systems. *Journal of Mechanical Science and Technology*, 29(6):2451–2460, June 2015.
- [27] K.-Y. Liang, S. van de Hoef, H. Terelius, V. Turri, B. Besselink, J. Mårtensson, and K. H. Johansson. Networked control challenges in collaborative road freight transport. *European Journal of Control*, 30(Supplement C):2 – 14, 2016. 15th European Control Conference, ECC16.

- 
- [28] N. Lyamin, L. Lan, and A. Vinel. Vehicle-to-vehicle communication in c-acc/platooning scenarios.(cooperative adaptive cruise control)(technical report). Technical Report 8, August 2015.
- [29] X. Ma. Towards intelligent fleet management: Local optimal speeds for fuel and emissions. pages 2201–2206. IEEE, October 2013.
- [30] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. *Control Systems Technology, IEEE Transactions on*, 8(4):695–708, July 2000.
- [31] M. Saeednia and M. Menendez. A consensus-based algorithm for truck platooning. *IEEE Transactions on Intelligent Transportation Systems*, 18(2):404–415, Feb 2017.
- [32] Socioeconomic Data and Applications Center. Population Density Grid, 2000.  
<http://sedac.ciesin.columbia.edu/downloads/maps/grump-v1/grump-v1-population-density/swedens.png>.
- [33] A. Tiganasu, C. Lazar, and C. Caruntu. Design and simulation evaluation of cooperative adaptive cruise control for a platoon of vehicles. In *2016 20th International Conference on System Theory, Control and Computing, ICSTCC 2016 - Joint Conference of SINTES 20, SACCS 16, SIMSIS 20 - Proceedings*, pages 669–674. Institute of Electrical and Electronics Engineers Inc., December 2016.
- [34] S. Tsugawa, S. Jeschke, and S. E. Shladover. A review of truck platooning projects for energy savings. *Intelligent Vehicles, IEEE Transactions on*, 1(1):68–77, March 2016.
- [35] S. van de Hoef. Fuel-efficient centralized coordination of truck platooning, 2016. QC 20160525.
- [36] S. van de Hoef, K. Johansson, and D. Dimarogonas. Fuel-efficient en route formation of truck platoons. *IEEE Transactions on Intelligent Transportation Systems*, PP(99), May 2017.
- [37] S. van de Hoef, K. H. Johansson, and D. V. Dimarogonas. Computing feasible vehicle platooning opportunities for transport assignments. *CoRR*, abs/1511.00849, 2015.

- 
- [38] S. van de Hoef, K. H. Johansson, and D. V. Dimarogonas. Coordinating truck platooning by clustering pairwise fuel-optimal plans. pages 408–415, 2015.
- [39] G. Yang, H. Xu, Z. Wang, and Z. Tian. Truck acceleration behavior study and acceleration lane length recommendations for metered on-ramps. *International Journal of Transportation Science and Technology*, 5(2):93 – 102, 2016.

TRITA 2017:169  
ISSN 1653-5146