Universiteit Utrecht

Faculty of Science

Decentral routing strategies for public transportation

Master thesis

Lisanne Waardenburg

Computing Science

Supervisors:

Dr. ir. Marjan van den Akker

Dr. Johan van Rooij

31st October 2022

Abstract

In case of large disturbances in public transportation networks, there often is little information available for central dispatching. Moreover, high frequency systems may use decentral dispatching. We consider a local dispatching strategy, where a cyclic order is assigned to the lines. This order determines which route the next vehicle to depart will take and the timetable determines when this will be.

It has been proved this system stabilises to the normal schedule, when all lines are operated with the same frequency. When looking at arbitrary frequencies, deciding whether a timetable can be operated with a certain number of vehicles is NP-complete in the strong sense.

The goal of this thesis is to consider variations, e.g. star networks or frequencies that are multiples of each other and see if these are NP-complete as well or if we can prove stabilisation results. In case of stabilisation we would also like to know how fast it stabilises. Finally we will consider the stabilised situation to see if we can deduce properties for the stabilised system.

Contents

1 Introduction

1.1 Problem description

When large disturbances happen in public transit networks, often essential information is unavailable at the central dispatcher. Because of this routes may be cancelled even though crew and material are available. Decentralised dispatching strategies might be a solution to this problem.

In [10] Van Lieshout et al. propose such a local dispatching strategy. In this paper every end station is considered separately and has the same strategy, namely assigning the next bus to arrive to the route that has last been assigned to. It then departs on the scheduled departure time or immediately if this has already passed. This policy makes sure every route is operated the same number of times. When a line is scheduled to depart, but no vehicles are available the entire schedule is updated, i.e. when a vehicle does become available it departs immediately and the next departure is based on the actual departure time, instead of the originally planned time.

In this thesis we will consider extensions to this paper. Van Lieshout et al. showed in [10] that the system stabilises in polynomial time when all lines have the same frequency. With arbitrary frequencies the stabilisation is not guaranteed to happen in polynomial time, unless $P = NP$, as was proved by Van Lieshout in [8].

We will consider the situation where frequencies are multiples of each other. Also, bus networks often are star-shaped. So we will research if the problem is solvable in polynomial time on path and star networks. Finally, we will look into the properties of a stabilised network. We will support and extend the theoretical results with simulation results.

2 Literature

Van Lieshout et al introduced a policy in [10] for decentral vehicle dispatching based on the rotor-router model. We start with some background in public transport scheduling, then we will consider the rotor-router model and known results for it and finally we will give an overview of aforementioned article.

Planning for public transportation is usually done in four steps. The first step is to make a line plan, then add the timetable, subsequently plan the material and finally plan the personnel. For this decentral approach only the line plan remains, all other plans are discarded and decisions are made per end station based on the information available there.

2.1 Rotor-router model

The rotor-router model finds its origin in physics, where it was studied as a self-organising system in [7] by Priezzhev et al. The rotor-router model can be seen as a deterministic version of a random walk. We have a graph where each vertex has a pointer to one of its outgoing edges and a cyclic order on all of the outgoing edges. Then we place an agent somewhere on this graph. In each step the agent arrives at a vertex and then moves from this vertex along the direction stored in its pointer, after that the pointer is updated to the next edge in the cyclic order.

Priezzhev et al showed in [7] that for undirected graphs, where the undirected edges are treated as two directed edges in opposite direction the agent will eventually walk an Euler circuit that is repeated forever.

Yanovski et al. extended this in [12] to a multiple agents problem in the context of patrolling a network. They proved that for one agent the Euler cycle is reached within time $O(|E|D)$, where E is the set of edges and D is the diameter of the graph. As an Euler cycle is reached, every edge is revisited in a period of $|E|$. For the multi agent case the similar result reads that for any edge $u \to v$ that was visited at time t, it must be revisited at least once again until time $t + |E| + 1$ by some agent.

The results so far do not say anything about the ratio between the most and least frequently visited edges yet. For k agents it is shown that the difference in traversals between any two edges $u_1 \to v$ and $v \to u_2$ is no more than $k + 1$ at any moment in time. The number of traversals for a vertex is defined as the minimum amount of traversals of any of its outgoing edges. With this definition, the result can be extended to any pair of vertices u, v to a difference that is no more than $d(u, v)(k + 1)$, where $d(u, v)$ denotes the length of the shortest path between u and v.

They also bound the blanket time, the first time at which the ratio between the number of traversals for any two edges does not exceed two, introduced by Winkler and Zuckerman in [11] for random walks. For k agents, at time $t \geq 2(1 + \frac{1}{k})|E|D$, for edges e_1, e_2 it holds that $\frac{z_t(e_1)}{z_t(e_2)} \leq 2$. Here $z_t(e)$ is the number of traversals for edge e until time t.

As the number of states for the algorithm is finite and the difference in traversals between edges cannot grow arbitrarily large, the system must enter a periodic motion. The length of this period must be a multiple of $\frac{|E|}{k}$, but there are no theoretical results for when this periodic motion is entered.

Chalopin et al. further investigate the behaviour of the rotor-router system towards stabilisation for multiple agents in [3]. Showing that for k agents the system stabilises in polynomial time, but the duration of the periodic motion can be superpolynomial in the number of edges.

Finally Dereniowski et al. show in [5] that the cover time, that is the time it takes until all edges are traversed, with k agents is at least $\Theta(\log k)$ times shorter than the time with one agent. For a star graph G with n nodes they bound the cover time t_c for $k \leq n$ agents by $t_c \leq 2\lceil \frac{n}{k} \rceil$ for any initialisation of the rotor and any initial positions of the agents.

2.2 A self-organizing policy for vehicle dispatching in public transit systems with multiple lines

In this paper [10] Van Lieshout et al. propose a policy for dispatching public transit vehicles in case of large disruptions in the system. This work follows on [9] and [4]. They propose a decentralised approach so not all information needs to be available at the main dispatcher, as that is often the case for large disruptions. The disadvantage of this decentralised approach is that passengers cannot rely on a schedule anymore.

Self-organising policies for public transit systems have been studied before, by Bartholdi and Eisenstein [2], Liang et al. [6] and Zhang and Lo [13] but only for single line systems to prevent bus bunching. Bartholdi and Eisenstein introduced a policy where vehicles are delayed by an amount that depends on the headway to the next vehicle and proved the headways equalise, no matter the initial position of the vehicles. Liang et al. proposed a policy where both the headway to the previous and the headway to the next vehicle are considered and showed this leads to faster equalisation of the headways. Zhang and Lo showed that for stochastic travel times the headway variance will eventually be limited.

Argote-Cabanero et al. have studied multiline systems in [1] but do not consider the possibility of dynamically assigning vehicles to lines.

The policy proposed by Van Lieshout et al. determines at each end station (we do not consider the intermediate stations) for each vehicle that arrives, what line it will operate next and at what time. All outgoing lines for the station have been added to a cyclic order and each time a bus arrives, it is assigned to the next line in this cyclic order. All lines have a headway, the desired time between two consecutive trips of that line. The departure time is set to be the previous departure time of the line plus the target headway or the current time if that has already passed.

As the policy from Van Lieshout et al. has many similarities to the rotor-router model, results are based on results from the rotor-router literature. However, the original rotor-router model does not allow waiting, so results do need to be adapted.

The results are based on the assumption that all lines have the same headway. For k the number of vehicles used in the system and k^* the minimum required number of vehicles, Van Lieshout shows that headways for all lines stabilise to the target headway H if enough vehicles are available and to an average headway of $\frac{k^*}{k}H$ if not. The maximum headway in the second case is bounded by $H+(k^*-k)H$. If the driving time of all lines is equal to the headways as well, the situation becomes even more similar to the rotor-router system, in that case the time before the system stabilises is $O(|\mathcal{L}|^2DH)$.

In [8] Van Lieshout shows that for arbitrary line frequencies determining the minimum number of vehicles required, can be reduced to 3-PARTITION, so unless $P = NP$ there will not be a guarantee that the system stabilises in polynomial time.

3 Preliminaries

3.1 Definitions

We will start by introducing most of the notation that is used throughout this article. If possible we comply with the notation introduced by Van Lieshout et al. in [10].

A public transit network is represented by a directed graph $\mathcal{G} = (\mathcal{S}, \mathcal{L})$, where $\mathcal{S}$ is the set of terminal stations and $\mathcal{L}$ is the set of lines, going from one station in $\mathcal{S}$ to another. The intermediate stations do not interfere with our algorithm and are therefore not included in the network. We assume $\mathcal{G}$ to be a connected and symmetric, as it is not connected all components may be considered separate networks and by symmetric we mean that is for every line $(s \to s') \in \mathcal{L}$, the line $(s' \to s)$ is also an element of $\mathcal{L}$.

For every line $l = (s, s') \in \mathcal{L}$ the time between departure at s and arrival at s' is referred to as the travel time, which is denoted by t_l. The travel time may be asymmetric, i.e. $t_{(s,s')} \neq t_{(s',s)}$. For every line we have a target headway, that is a desired time between two consecutive departures of this line. If we have a general target headway for every line in the network we will denote this by H. If we have a target headway for a specific line l this is denoted by h_l. We assume all travel times and headways are integer, if they are not we can scale the problem so they become integer. For the sets of lines starting and terminating at a station $s \in \mathcal{S}$ we respectively use $\delta^+(s)$ and $\delta^-(s)$. So $\delta^+(s) = \sum_{l \in \mathcal{L}(s)} f_l$ where $\mathcal{L}(s) = \{(a, b) \in \mathcal{L} \mid a = s\}$.

As we will look into situation where not all lines have the same frequency, we introduce a frequency for every line, denoted by f_l. Frequencies should have integer values, but just like with travel times we can go from fractional values to integer values by scaling. Opposed to the travel times, we do assume the frequencies to be symmetric.

We denote the minimum required number of vehicles by k^*. There is a fixed number of vehicles available in the system, which we will denote by k. For the case where all lines are driven with the same frequency it holds that

$$k^* = \left\lceil \frac{\Sigma_{l \in \mathcal{L}} t_l}{H} \right\rceil .$$

We consider the system in iterations, where we can see the iterations as the discrete moments in time where events happen. We can do this as all travel times and headways have integer values. We can set the duration of one iteration to the greatest common divisor of all travel times and headways. The events can either be arrivals or departures of vehicles or both as a vehicle may depart immediately when it arrives.

3.2 Cover and stabilisation time

We would like to analyse how long it takes for the system to stabilise, but before we can do that, we define what a stable state is and how we count how long it takes until we reach this.

Definition 3.1 (Stable state (Van Lieshout et al. [10])). We call a state $\mathcal{V}_t$ *stable* if there exists $t' > t$ such that $\mathcal{V}_{t'} = \mathcal{V}_t$.

Definition 3.2 (Stabilisation time). The *stabilisation time* is the number of iterations it takes until the first time we reach a stable state, a state that is part of a periodic motion, so the first state that is repeated.

Another concept that we will use to find the stabilisation time, is the cover time:

Definition 3.3 (Cover time). The *cover time* of a system is the maximum number of iterations from some configuration to visit all nodes at least once.

Finally we need to know when two states are considered equal. Note that a state is fully defined by:

- The number of vehicles per station, denoted by v_s for station s.
- The position of the rotor (for every node, in a star only the centre node is relevant). For a station s we can denote this by $R^s_{s'}$ if the next vehicle from s should depart to s'.
- The last departure times for each line, denoted by d_l for a line l.

If all of these are equal, the states induce the same periodic motion, so we call them equal.

3.3 Potential function and anti-vehicles

To help us derive a bound on the stabilisation time from the cover time, we use the concept of anti-vehicles as introduced by Van Lieshout et al. in [10]. In this section we consider networks where the travel time is equal for all edges. Let this travel time also be equal to the headway H so we can consider the system in iterations of duration H.

Potential function

For a formal definition let us start by introducing the charge of a station and a potential function based on this charge. Let $i_s(m)$ be the number of vehicles at station s after iteration m. Now we can define the charge of a station s after iteration m as $C_s(m) = i_s(m) - \deg(s)$. Note that the number of incoming lines equals the number of outgoing lines for every station $s \in S$ so the degree can be either the in- or the out-degree. A station can now be in three situations, positively charged if $C_s(m) > 0$, negatively charged if $C_s(m) < 0$ or neutral otherwise.

In the situation where we have $k = k^* = |\mathcal{L}|$, the number of vehicles equals the number of lines, this makes the total charge over the network 0.

In every iteration $\min\{i_s(m), \deg(s)\}$ vehicles leave station s, and at most $\deg(s)$ vehicles arrive at this station. So if we have a positively charged station, the charge can never increase. This gives us the following options for a station to switch between states: positive $\rightarrow$ neutral $\leftrightarrow$ negative.

Based on this charge we can define a potential function $\Phi(m)$ as the sum of positive charges, $\Phi(m) = \sum_{s \in S: C_s(m) > 0} C_s(m)$. As for all stations we know the positive charge cannot increase, we know it holds that $\Phi(m) \geq \Phi(m+1)$.

The system has stabilised if and only if $i_s(m) = \deg(s) \forall s \in S$. If the potential function $\Phi(m)$ reaches 0, we know that $i_s(m) \leq \deg(s) \forall s \in S$ and as the total number of vehicles equals the total number of lines, we can conclude that then $i_s(m) = \deg(s) \forall s \in S$.

Anti-vehicles

Now we can define anti-vehicles as the carriers of negative charge over a network. That is, whenever a line is not traversed by a vehicle in some iteration, we can see this as the line being traversed by an anti-vehicle.

Suppose no vehicle traverses the line from station s to station s' in some iteration j. Then in this iteration at station s less than $\deg(s)$ vehicles depart, increasing the charge of this station. At the destination station s' no vehicle arrives, decreasing the amount of vehicles and thus the charge there. So we can see this as the negative charge being moved from s to s' by the anti-vehicle.

It does not have to be that the charge of station s increases after iteration j as it may be that an anti-vehicle arrives at s from another station. When a anti-vehicle arrives at a positively charged station, the charge here is decreased making a negative charge disappear from the network and decreasing the potential function Φ. Note that an anti-vehicle can only depart from a negatively charged station, as from a positively charged station t exactly $\deg(t)$ vehicles depart.

3.4 Bounding stabilisation time

We know that once the potential function reaches 0, we have a stabilised situation, so we are interested in how quickly the potential function decreases. Suppose at some point we have some charge $\Phi(m) = p$ after iteration m, then we have p anti-vehicles in our network. These anti-vehicles travel through the network according to the same policy as the normal vehicles but with the cyclic order reversed at every station.

If we have the cover time for a network with p vehicles, we know that the potential function decreases at least once during this cover time. Call the cover time for p vehicles c_p. Then as the potential function can decrease at most p times, the stabilisation time is bound by $\Sigma_{j=1}^{p} c_p$.

4 Policies

In this section we will discuss possible policies we can use to locally dispatch vehicles and why we chose to focus on one of these policies in this document. We start by discussing the options for single frequency networks and then discuss two extensions to the chosen policy for multiple frequency networks.

We have considered the following policies to use as a local dispatching strategy. As we have to make a choice for our policy in two dimensions, line assignment and choice of departure time, where for every dimension we have two options, there are four policies to consider. Not all of them are interesting as we will see shortly. Our options are the following:

- **STAT**: Static: whenever a vehicles arrives at a station s after departing from station s', it gets assigned a departure on the line $s \to s'$ with a time that is based on our timing strategy.
- **DYN**: Dynamic: whenever a vehicles arrives at a station s after departing from another station s', it is assigned to the next line in some predefined order at station s. In our case this will be the next in the cyclic order of the lines each station has. It may be assigned to $s \to s'$, but also to another line $s \to s''$.
- **ASAP**: As soon as possible: upon arrival of a vehicle at a station, it departs immediately on the next line that gets assigned to the vehicle.
- **SYNC**: Synchronised: when a vehicle gets assigned a line $s \to s'$, the last departure time of that line is checked. If this is $h_{(s,s')}$ or more time ago, the vehicle departs immediately. Otherwise it waits till this time has past and leaves then.

These policies can be divided into pairs, where we have to choose one of each pair for a total description of the policy we are considering. The pairs are:

- STAT or DYN for what line gets assigned to the vehicle.
- ASAP or SYNC for when the vehicle has to depart.

The STAT policies can be seen as single line systems. Our goal is to find properties of multiple frequency systems so that makes STAT a less interesting policy. So we will focus on the DYN policies.

We know from the work by Van Lieshout in [10] that DYN-SYNC stabilises in polynomial time for the situation where all headways are equal. Let us consider DYN-ASAP, in this case vehicles never wait at a station. In the very simple example where we have 2 stations with 2 lines, one in each direction with a travel time of 30 minutes in both directions and a headway of 15 minutes. Now suppose we have 2 vehicles in both stations at the start. Then we would like for one of these vehicles to leave at both stations and the other to leave after 15 minutes. In that case all headways are made. But in the ASAP system, both vehicles will leave at the same time, causing a periodic motion where 2 vehicles drive at the same time, every 30 minutes. This is a stable situation as the state is repeated in a periodic motion, but it in most cases not the desired situation as the headways are not made. For this reason we have decided to focus on SYNC policies.

Now we have an extra choice to make if there are multiple frequencies involved. What will we adhere more to, the time or the rotor order? We will first discuss the time based case (TIME). In this case vehicles are assigned to a line based on the earliest desired departure time. An advantage of this system is that perturbations one line do not immediately cascade onto the other lines. A possible disadvantage may be that in case of delays (or a shortage of vehicles) the frequencies are not adhered to. I.e. if one line should be driven twice per hour, and another once per hour but only one vehicle arrives every hour, then whenever the half-hour line departs, the single frequency line has departed an hour ago, so its next desired departure time is now. This is smaller than the next desired departure time of the double frequency line, which is in half an hour. So both lines are driven the same amount of times. Let us illustrate this with a concrete example:

Suppose we have a network with three stations, s, s' and s'', one vehicles and two bidirectional lines $s \to s', s \to s'', s' \to s$ and $s'' \to s$, where for any of these lines the travel time is 30 minutes. This network is drawn in Figure 1. We want the line (s, s') to depart every 60 minutes and the line (s, s'') to depart every 30 minutes, so $f_{(s,s')} = 1$ and $f_{(s,s'')} = 2$. The only vehicle starts at station s. In Table 1 the events from this example and the desired departure times can be found.

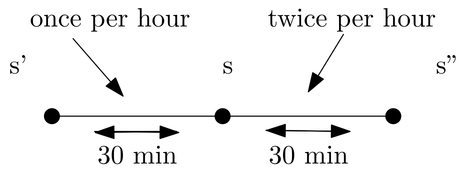

Figure 1: The example network

We can see here that $s \to s'$ and $s \to s''$ are served alternatingly, resulting in both lines being operated once every two hours even though based on frequency the line $s \to s''$ should be operated twice as many times.

One of the primary results obtained by Van Lieshout et al. in [10] is a bound on the difference between how many times two lines have been driven in total. For a double frequency line, the equivalent would be bounding the difference between twice the single frequency line and the double frequency line, as we want the double frequency line to drive twice as many times. But as we can see in the previous example this may grow arbitrary large. That is why we consider the rotor based approach where this result does hold, as we will see later.

Timestamp	Event	Next $s \to s'$	Next $s' \to s$	Next $s \to s''$	Next $s'' \to s$
8:00	Departure $s \to s''$	8:00	8:00	8:30	8:00
8:30	Arrival at s''	8:00	8:00	8:30	8:00
8:30	Departure $s'' \to s$	8:00	8:00	8:30	9:00
9:00	Arrival at s	8:00	8:00	8:30	9:00
9:00	Departure $s \to s'$	10:00	8:00	8:30	9:00
9:30	Arrival at s'	10:00	8:00	8:30	9:00
9:30	Departure $s' \to s$	10:00	10:30	8:30	9:00
10:00	Arrival at s	10:00	10:30	8:30	9:00
10:00	Departure $s \to s''$	10:00	10:30	10:30	9:00
10:30	Arrival at s''	10:00	10:30	10:30	9:00
10:30	Departure $s'' \to s$	10:00	10:30	10:30	11:00
11:00	Arrival at s	10:00	10:30	10:30	11:00
11:00	Departure $s \to s'$	12:00	10:30	10:30	11:00
11:30	Arrival at s'	12:00	10:30	10:30	11:00
11:30	Departure $s' \to s$	12:00	12:30	10:30	11:00
12:00	Arrival at s	12:00	12:30	10:30	11:00
12:00	Departure $s \to s''$	12:00	12:30	12:30	11:00

Table 1: Time based example

Let us look at the rotor based policy (RR). We will start by discussing how we model the higher frequency lines in this case. Note that we will still use one base value for both frequencies and travel time. We will call this base value H. Where a line with frequency 1 has a headway H and the travel times are multiples of $\frac{H}{f_x}$, where $f_x = \operatorname{lcm} F$, with F the set of all frequencies that appear in the system. The headway of a line is directly connected to the frequency. Consider a line l_2 with frequency f_{l_2}, where $f_{l_2} \in \mathbb{N}$, then the headway for l_2 is $\frac{H}{f_{l_2}}$.

We can now consider the system in iterations of $\frac{H}{f_x}$ time units. Every event will happen at a time that is a multiple of $\frac{H}{f_x}$ So if a vehicle departs in iteration i, over a line that has travel time $k \cdot H$ for some $k \in \mathbb{N}$ it will arrive (and may also depart again) in iteration $i + k \cdot f_x$. Suppose we have $H = 60$ minutes, in a network with three lines, that have frequencies 2, 3 and 4. Then we will consider the system in iterations of 5 minutes, and the headways for the lines will be $30, 20$ and 15 minutes respectively.

Now we can model the rotor in a station where a higher frequency line departs as a cyclic order that contains a line with frequency f, f times. Note that the order in which the lines are added to the cyclic order does influence the behaviour of the system. We will see this in more detail later.

A higher frequency line l, with frequency f_l now needs to depart after its predecessor in the rotor order, but also at least $\frac{H}{f_l}$ after previous departure of this line. So the next departure is updated for the next appearance of line l in the rotor order. When the actual departure time for this next departure of l is known, the next appearance of l after that in the rotor order is updated et cetera.

We will illustrate this with an example in the same system as for the previous policy, that can be found in Figure 1. So suppose we have a network with three stations, s, s' and s'' and two bidirectional lines $s \to s'$, $s \to s''$, $s' \to s$ and $s'' \to s$, where for any of these lines the travel time is 30 minutes. We want the line (s, s') to depart every 60 minutes and the line (s, s'') to depart every 30 minutes, so $f_{(s,s')} = 1$ and $f_{(s,s'')} = 2$. There is only one vehicle in the network, which starts at station s.

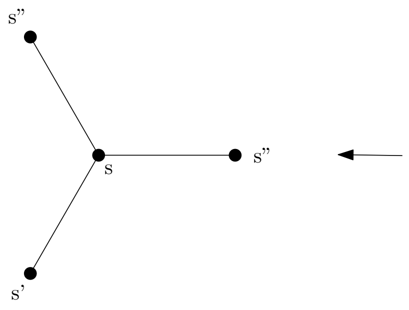

Figure 2: Rotor for the rotor based example

The rotor for this case will contain two edges to s'' and one edge to s', as illustrated in 2. The arrow indicates the starting position of the rotor. With every departure the rotor moves one node in clockwise order. In Table 2 the events from this example and the desired departure times can be found.

In this case every three hours the line $s \to s''$ is served twice, while the line $s \to s'$ is served once. So the system adheres to the frequencies we have given the lines.

Adding higher frequency lines more often to the cyclic order and then using this order to decide which lines will be operated, allows us to derive results about the total number of times certain line have been driven up to some time. We still know for all outgoing edges of a node that the difference in the number of times they are traversed is bounded if we include the frequency. The question that we still need to answer is how to order the edges, and does this make a difference?

Suppose we have a station s which has lines to three other stations, a, b and c. The line $s \to a$ has frequency 4, the line $s \to b$ has frequency 2 and the line $s \to c$ has frequency 1, then we can order these in different ways, as is shown in Figure 3. In section 7 we will consider different rotor configurations for star graphs.

Concluding, in the SYNC-DYN case for multiple frequency systems we have to make a choice in how we prioritise the assignment of vehicles. The options we have are:

- **RR**: Rotor based: when multiple lines have their last departure more than their headway in the past, the next line that will be served is selected based on the rotor order.
- **TIME**: Time based: when multiple lines have their last departure more than their headway in the past, the next line that will be served is selected based on for what line this happened earliest. Note that this is only relevant in multiple frequency systems as otherwise the rotor order will always be the same as the time based order. Ties can be broken by using the rotor order.

Timestamp	Event	Next $s \to s'$	Next $s' \to s$	Next $s \to s''$	Next $s'' \to s$
8:00	Departure $s \to s''$	8:00	8:00	8:30	8:00
8:30	Arrival at s''	8:00	8:00	8:30	8:00
8:30	Departure $s'' \to s$	8:00	8:00	8:30	9:00
9:00	Arrival at s	8:00	8:00	8:30	9:00
9:00	Departure $s \to s'$	10:00	8:00	8:30	9:00
9:30	Arrival at s'	10:00	8:00	8:30	9:00
9:30	Departure $s' \to s$	10:00	10:30	8:30	9:00
10:00	Arrival at s	10:00	10:30	8:30	9:00
10:00	Departure $s \to s''$	10:00	10:30	10:30	9:00
10:30	Arrival at s''	10:00	10:30	10:30	9:00
10:30	Departure $s'' \to s$	10:00	10:30	10:30	11:00
11:00	Arrival at s	10:00	10:30	10:30	11:00
11:00	Departure $s \to s''$	10:00	10:30	11:30	11:00
11:30	Arrival at s''	10:00	10:30	11:30	11:00
11:30	Departure $s'' \to s$	10:00	10:30	11:30	12:00
12:00	Arrival at s	10:00	10:30	11:30	12:00
12:00	Departure $s \to s'$	13:00	10:30	11:30	12:00
12:30	Arrival at s'	13:00	10:30	11:30	12:00
12:30	Departure $s' \to s$	13:00	13:30	11:30	12:00
13:00	Arrival at s	10:00	13:30	11:30	12:00
13:00	Departure $s \to s''$	13:00	13:30	13:30	12:00

Table 2: Rotor based example

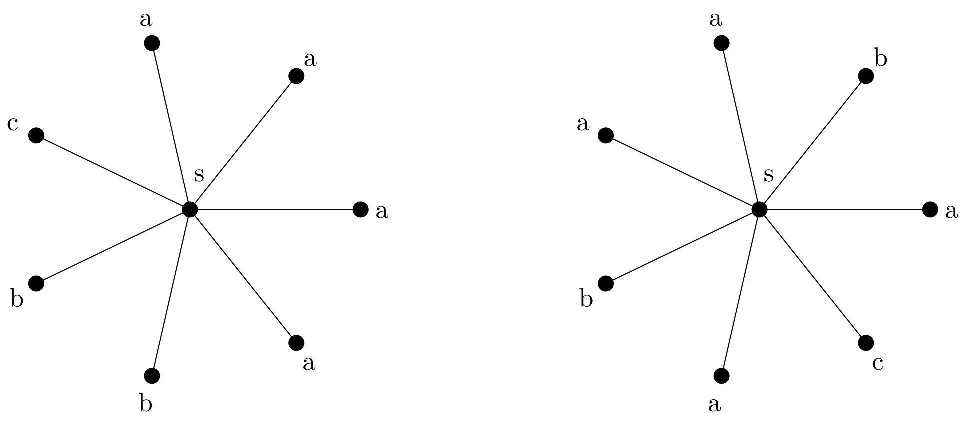

Figure 3: Examples of different rotor orders for the same network.

As the RR policy gives us more insight in how many times a certain line has been driven, we choose to continue with this policy, so from now on we will be using the SYNC-DYN-RR policy unless indicated otherwise.

5 Multiple frequency networks

Some of the intermediate results obtained by Van Lieshout et al. in [10] can be shown to still hold in the case where not all lines have the same frequency. In this section we will discuss these results. Let $g(s \to s')$ denote the number of departures of line $s \to s' \in \mathcal{L}$ up to time t.

As the lines are served in a round robin fashion which takes the frequencies into account we can see that for any pair of lines $s \to s'$ and $s \to s''$ originating at the same station it holds that $|f_{s\to s''} \cdot g(s \to s') - f_{s \to s'} \cdot g(s \to s'')| \leq \max\{f_{s\to s'}, f_{s \to s''}\}$. As it may be that one of the pair has already had all its departures of the current cycle and the other still needs all of them. Note that in many cases this bound can be tighter if we 'nicely' spread the lines in the cyclic order.

Let $\mathcal{S}_1, \mathcal{S}_2$ be a partition of the set of stations $\mathcal{S}$. Then it still holds that a vehicle cannot cross from $\mathcal{S}_1$ to $\mathcal{S}_2$ twice without crossing back, so we have that $|g(\mathcal{S}_1 \to \mathcal{S}_2) - g(\mathcal{S}_2 \to \mathcal{S}_1)| \leq k$

Lemma 5.1. For every line $s \to s' \in \mathcal{L}$, it holds that $|g(s \to s') - g(s' \to s)| \leq k \cdot f_{s \to s'}$.

Proof. We define $g(s) = \min\{\frac{g(s \to s')}{f_{s\to s'}} | s \to s' \in \delta^+(s)\}$ for a station $s \in \mathcal{S}$, denoting the minimum number of departures for any line leaving s scaled by the frequency of the line. As the lines are served in round-robin fashion and by scaling with the frequency all lines are served once per round, we have that $0 \leq \frac{g(s\to s')}{f_{s \to s'}} - g(s) \leq 1$.

Now suppose the lemma is not true, i.e. there exists a pair of opposite lines $s \to s'$ and $s' \to s$ with $\frac{g(s\to s')}{f_{s\to s'}} = j$ and $\frac{g(s' \to s)}{f_{s' \to s}} \leq j - k - 1$. Note that $f_{s \to s'} = f_{s' \to s}$ as frequencies are symmetric.

By definition, it holds that $g(s) \geq j - 1$ and $g(s') \leq j - k - 1$. If we now consider the partition $\mathcal{S}_1, \mathcal{S}_2$ where $\mathcal{S}_1 = \{s \in \mathcal{S} | g(s) \geq j - k\}$ and $\mathcal{S}_2 = \{s \in \mathcal{S} | g(s) \leq j - k - 1\}$. We have $s \in \mathcal{S}_1$ and $s' \in \mathcal{S}_2$. Let the set of lines starting in $\mathcal{S}_1$ and ending in $\mathcal{S}_2$ be M. We know the flow from s to s' is $j \cdot f_{s\to s'}$ and for all other lines from $\mathcal{S}_1$ to $\mathcal{S}_2$ the flow is at least $j - k$ times the frequency of the line. We find that

$$g(\mathcal{S}_1 \to \mathcal{S}_2) \geq \Sigma_{l \in M} f_l (j-k) + j \cdot f_{s \to s'}.$$

We can also bound the flow from $\mathcal{S}_2$ to $\mathcal{S}_1$ by

$$g(\mathcal{S}_2 \to \mathcal{S}_1) \leq \Sigma_{l \in M} f_l (j-k) + (j-k-1) \cdot f_{s \to s'}.$$

Therefore it follows that

$$g(\mathcal{S}_1 \to \mathcal{S}_2) - g(\mathcal{S}_2 \to \mathcal{S}_1) \geq (k+1) f_{s \to s'}.$$

And as frequencies are at least 1 this contradicts our previous observation. So by contradiction the lemma must be true. □

Theorem 5.2. For two lines $s_1 \to s_1', s_2 \to s_2' \in \mathcal{L}$, it holds at any time that $|g(s_1 \to s_1') - g(s_2 \to s_2')| \leq (\text{lines}(s_1', s_2) + 1)(f_{\max} + 1)$. Where by $\text{lines}(u, v)$ we denote the minimum number of edges that has to be traversed to get from u to v and where $f_{\max}$ is the maximum frequency that appears in the system.

Proof. The proof from Van Lieshout et al. in [10] still holds with the adapted result from lemma 5.1. □

Lemma 5.3. After a certain finite time, the system enters a periodic motion. In every cycle, every line is performed the same number of times.

Proof. As the system can still only be in a finite number of states and as by the adapted version of Theorem 5.2 the number of times two lines are served cannot differ too much, the proof from Van Lieshout et al. in [10] for Lemma 4 still holds here. □

6 Path networks

In this section we will consider networks which consist of n vertices which we can order $v_1, \ldots, v_n$ and $n-1$ bidirectional edges such that there is an edges between vertices v_i and v_{i+1} for $i = 1, \ldots, n-1$.

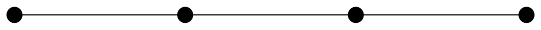

Figure 4: A path graph of length 4.

6.1 Theoretical bounds

We already knew from Van Lieshout et al. in [10] that the general bound on the single frequency system where all headways and travel times equal H is $O(|\mathcal{L}^2|DH)$ time for stabilisation. As in the case of a path both $\mathcal{L} = O(n)$ and $D = O(n)$, we can see this as $O(n^3)$ for cases where $H = 1$.

As the last anti-vehicle can take $O(n^2)$ iterations before arriving at a positively charged station, we know that for path graphs stabilisation will take $\Omega(n^2)$ iterations.

6.2 Simulation results

We have ran some simulations for path graphs of different lengths to see how long it takes for them to reach a stable state if all vehicles start in one end node. For a path graph of size n where all edges have travel time 1, there are $2n - 2$ vehicles in the network. The results can be found in Figure 5.

The stable situation for path graphs has two vehicles in all nodes except for the end nodes, there only is one vehicle. All edges are driven each iteration. From Figure 5 we cannot conclude anything more than we already had from the theoretical results.

7 Star networks

A star graph is a graph with one central node surrounded by all other nodes. We will call the central node the centre node and the other nodes end nodes. These end nodes only have one adjacent (bidirectional) edge connecting them to the centre. Star graphs are very relevant to public transportation as public transport networks often are (similar to) stars as this can be seen as a all lines departing from a central station.

We will start by considering the rotor-router procedure on star graphs. After that we will extend this with more conditions and give a cover time and stabilisation time where we have found those.

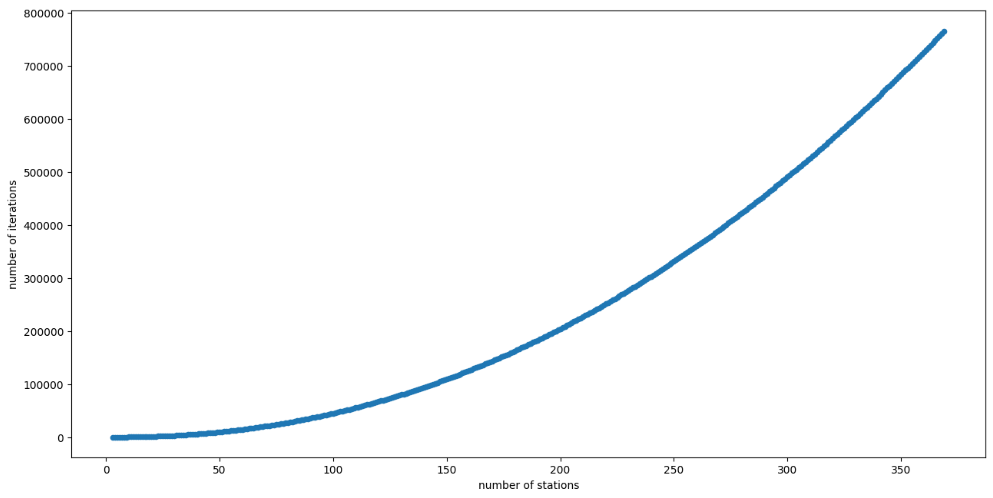

Figure 5: Number of iterations until stable state for path graphs.

For every station in our graph we have a rotor order, which is a cyclic order of all lines departing from that station. If we have the cyclic order fixed we can call all possible rotor positions 1 to $\sum_{s' \in \delta^+(s)} f_{s'}$ for station s as we add an higher frequency line l to the cyclic order f_l times. For a star graph, the rotor position for the end nodes will always be the same and thus is not interesting to mention. So in a star graph we use r to indicate the rotor position, which indicates the next line a vehicle will be assigned to, for the centre node.

7.1 Rotor-router star

In the first policy we examine we have a star graph of $n + 1$ nodes, where the travel time for all edges is 1. We will use the original rotor-router policy. I.e. all vehicles will depart every iteration, there is no headway to take into account, so vehicles never have any reason to wait. This may cause multiple vehicles to drive the same edge in the same iteration.

We define 3 starting situations:

Centre	All vehicles start in the centre node.
End	All vehicles start in the same end node.
Scattered	The vehicles are scattered around the network at the start, which specifically means not all vehicles are in the same node.

Suppose we have k vehicles. In the first (Centre) situation, in the first iteration all k vehicles drive from the centre to an end node. They return in the next iteration and then depart to the next k end nodes in the rotor order in the third iteration. So every two iterations k new nodes are visited, giving us a cover time of $2\lceil \frac{n}{k} \rceil$.

In situation two (End) the first iteration is spent travelling to the centre for all vehicles. Then we arrive at situation 1 again. This yields a cover time of $2\lceil \frac{n}{k} \rceil + 1$.

In situation three (scattered) we get two groups of vehicles which alternatingly visit the centre, one group of vehicles that starts in the centre and the other group of vehicles that starts in some end node. Still a new node is visited per vehicle every two iterations, yielding a cover time of $2\lceil \frac{n}{k} \rceil + 1$ as it may be that all vehicles start in end nodes.

In all three starting situations described above we have that a set of end nodes is visited each iteration. This set of visited nodes shifts by k every two iterations. After $2\,\mathrm{lcm}(n, k)$ iterations, the same set of nodes will be visited (and the rotor will be in the same position, as the last visited node is the same) so we have the exact same situation. We have a constant stabilisation time as the first situation is repeated already.

7.2 Single frequency network

Now we extend the situation by demanding that all edges are travelled at most once per iteration. So if there are more than n vehicles in the centre, some will remain there until the next iteration.

In this situation where we have $n + 1$ nodes, we need $2n$ vehicles to allow all edges to be driven at the same time.

We again consider the same three starting situations as in the rotor router case. In the first situation n vehicles can depart from the centre, one to each end node, on the first iteration. In the next iteration the other n vehicles can drive from the centre to the end nodes, while the group that departed in the first iteration returns to the centre. Giving us a stable situation where n vehicles drive from the centre and n vehicles drive to the centre every iteration. The cover time and stabilisation time both are one iteration.

We consider the second situation, where all vehicles start in the same end node, we will call this node the stock node. In this case only one vehicle can depart in the first iteration, going towards the centre. Then in the second iteration this vehicles goes to an end node, while another vehicle goes towards the centre. If the first vehicle went back to the stock node, we have one vehicle drive towards the centre in the third iteration again. Then in the fourth iteration, two vehicles drive towards the centre, so in the iteration after that two vehicles visit new edges. As long as the stock node is not visited again, every two iteration one more vehicle is visited per iteration. So the number of end nodes visited in iteration i is $\lfloor \frac{i}{2} \rfloor$. So after i iterations $\sum_{j=0}^{i} \lfloor \frac{j}{2} \rfloor = \frac{i}{2}(1 + \lfloor \frac{i}{2} \rfloor)$ new end nodes have been visited. So to find the cover time we need an i such that $\frac{i}{2}(1 + \lfloor \frac{i}{2} \rfloor) \geq n$, so after $\mathrm{O}(\sqrt{n})$ iterations all end nodes have been visited. From the work of Van Lieshout [10] we know that at least once per cover time the potential function decreases, so this gives us a stabilisation time of $\mathrm{O}(n\sqrt{n})$.

Note that the number of vehicles that can visit new nodes in one iteration can never grow larger than n, but after $\mathrm{O}(\sqrt{n})$ iterations, $\mathrm{O}(\sqrt{n})$ vehicles will be visiting new nodes which is less than n for sufficiently large n.

For the third situation we can say that both cover time and stabilisation time will be at least as good as the second situation, so the same bounds will still hold.

7.3 Double frequency line

Next we consider the situation where all lines have frequency 1 except for one line that has frequency 2. This means all frequency 1 lines may depart every two iterations and we model the frequency 2 line as having two positions in the router order, which can both depart once every two iterations, and not in the same iteration. The rotor can be found in Figure 6.

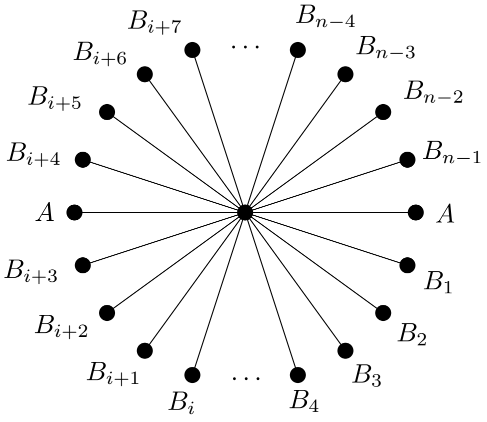

Figure 6: The double frequency line rotor

We consider this network both for the case where all edges have travel time 1 and for the case where all edges have travel time 2.

7.3.1 Travel time 1

We will consider the second starting situation, where all vehicles start in the same end node, as this clearly is the worst case scenario. For the travel time 1 scenario in a star network of $n+1$ nodes, we have a total distance of $2n$ that should be travelled every two iterations, so n vehicles can suffice to make all the headways.

In the situation where not all vehicles have left the stock yet, in every iteration there is at least one edge that should be driven based on headway, but it is not as there are not enough vehicles available, we sat these edges that are not driven are skipped. We use the rotor based system, so we can bound the difference of the number of times different edges have been driven. We again use $g(s \to s')$ to denote the number of times edge $s \to s'$ has been driven up to some time t. For frequency 1 lines l_1 and l_2 and frequency 2 line l_3 we have that

$$|g(l_1) - g(l_2)| \leq 1$$

and

$$|2 \cdot g(l_1) - g(l_3)| \leq 2.$$

Because of this bound we know that all edges are skipped some time. Whenever the edge towards the stock node is skipped, this allows the charge as defined in chapter 3 to decrease. Eventually leading to all vehicles leaving the stock (as long as there are not too many vehicles in the system).

Suppose we have a situation where all vehicles are driving around, so the stock is empty and there is no edge that contains more than one vehicle. We can now divide the vehicles over three groups, in every iteration.

x_1^i This contains the vehicles that are leaving from the centre in iteration i.

x_2^i This contains the vehicles that will remain in the centre in iteration i.

x_3^i This contains the vehicles that depart from the end vertices in iteration i.

We will use $|x_j^i|$ to indicate the size of group j in iteration i.

We will call the first iteration we consider iteration 1. Then in the first iteration $|x_1^1|$ vehicles will travel from the centre to the endpoints, we will distinguish two cases.

Case 1: in this case one of the double frequency lines is travelled by these $|x_1^1|$ vehicles. In the next iteration all $|x_2^1| + |x_3^1|$ vehicles can travel from the centre to an endpoint as this includes as most one of the double frequency lines. The $|x_1^1|$ vehicles that departed in the first iteration will return during this iteration and will be all allowed to leave in the next iteration. This gives us a pattern of alternatingly $|x_1^1|$ and $|x_2^1|+|x_3^1|$ end nodes being visited, where the two double frequency lines are in different groups.

Case 2: in this case all $|x_1^1|$ vehicles depart from the centre to endpoints, but none of these endpoints belong to the double frequency line. Then in the next iteration one of the double frequency lines will be served as all other lines can be served then based on the amount of vehicles. From this point on we can go back to the situation of case 1.

So in either two or three iterations the system is stable. From this we can conclude that the system with a double frequency line stabilises as well.

7.3.2 Travel time 2

We have a star network with n vertices, one centre vertex and $n-1$ end vertices. One of the lines from centre node to end node has a double frequency compared to the other lines. So if we set the headway for all other lines to be 2, the headway for this line is 1. We model this by adding the double frequency line to the rotor order twice. The double frequency lines may be anywhere in the order, close together or far apart. The only restriction is that they may not be driven during the same iteration. This situation is illustrated in Figure 6.

To compare to the situation where all lines have frequency 1 and driving time 1, where the vehicles arrive in the end when the next one is allowed to depart from the centre, we set the travel time to be 2 for all lines in all directions in this scenario.

Now we have $2n$ (n from centre to end and n from end to centre) lines of length 2 in the rotor order that need to be driven every 2 iterations, driving any one lines takes 2 iterations, so we need $2n$ vehicles in our network.

We start in the situation where all vehicles have spread out throughout the network, so no end vertex has more than 1 vehicle (and no vertex has a vehicle and one almost arriving). We divide all vehicles over five groups, in each iteration.

x_1^i This contains the vehicles that are leaving from the centre in iteration i.

x_2^i This contains the vehicles that arrive in the centre in iteration i.

x_3^i This contains the vehicles that depart from the end vertices in iteration i.

x_4^i This contains the vehicles that arrive in the end vertices in iteration i.

x_5^i This contains the vehicles that will remain in the centre in iteration i.

A vehicle that departs from the centre in iteration i, arrives at an end node in iteration $i+2$ and can then also depart from this end node in iteration $i+2$. The situation is illustrated in Figure 7. We will use $|x_j^i|$ to indicate the size of group j in iteration i.

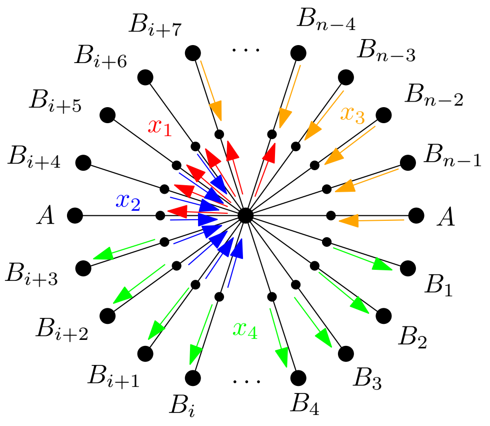

Figure 7: The spreading of the groups over the rotor network.

We will call the first iteration we consider iteration 1. Let us first make some observations. $|x_4^1| + |x_3^1| \leq n$, this follows from our assumption that the vehicles have spread over the network. Each edge can only have a vehicle arriving at the end station or departing from it, as vehicles never depart to the same end node within two iterations (for this we consider the double frequency line to have two end nodes that cannot be served in the same iteration). We also know that $|x_1^1| + |x_4^1| \leq n$ as otherwise more than n vehicles have left in two consecutive iterations.

Every iteration we have the following equalities:

$$\begin{aligned} x_1^{i+1} + x_5^{i+1} &= x_2^i + x_5^i \\ x_2^{i+1} &= x_3^i \\ x_3^{i+1} &= x_4^i \\ x_4^{i+1} &= x_1^i \end{aligned}$$

In the first iteration $|x_1^1|$ vehicles will depart from the centre. Without loss of generality we assume one of the double frequency lines is served by these vehicles. If this is not the case, we can consider the next iteration to be our first iteration. We have to do this at most four times, as in four iterations all vehicles have arrived at the centre once, so at least one of the double frequency lines is served.

We separate two cases: $|x_1^1| + |x_5^1| + |x_2^1| \leq n$ and $|x_1^1| + |x_5^1| + |x_2^1| > n$.

Case 1: In the first case we can conclude that $|x_1^1| + |x_5^1| + |x_2^1| = n$, as we know that $|x_4^1| + |x_3^1| \leq n$, so here we have $|x_4^1| + |x_3^1| = n$ as well.

As one of the double frequency lines is served in the first iteration, we know that in the second iteration at most $n - |x_1^1|$ vehicles may depart from the centre, which is exactly $|x_2^1| + |x_5^1|$ vehicles. So every line in the rotor is served exactly once after two iterations.

In the third iteration at most $|x_1^1|$ lines can be served as all others still need to wait their headway. $|x_3^1|$ is the number of vehicles that is in the centre at the start of iteration 3 and can depart. We will now show that $|x_3^1| \leq |x_1^1|$. We know that $|x_4^1| + |x_1^1| \leq n$ as they have departed in two consecutive iterations in the past. We also have that $|x_4^1|+|x_3^1| = n$ which gives us $|x_3^1| \geq |x_1^1|$.

As at most $|x_1^1|$ vehicles may depart and as $|x_3^1| \geq |x_1^1|$ we can say that exactly $|x_1^3| = |x_1^1|$ vehicles depart on lines which contain exactly one of the double frequency lines. $|x_5^3| = |x_3^1| - |x_1^1|$ vehicles remain in the centre. Then in the fourth iteration $|x_4^1|$ additional vehicles have arrived at the centre, so there are $|x_4^1| + |x_5^3|$ vehicles in the centre now. As $|x_4^1| + |x_3^1| = n$, we have that $|x_4^1| + |x_5^3| = n - |x_3^1| + |x_5^3| = n - |x_1^1|$ and thus all $|x_4^1| + |x_5^3|$ vehicles that are in the centre can depart in the fourth iteration.

So after four iterations all lines have been served exactly twice. We now have a repeating pattern of two iterations where in the first iteration $|x_1^1|$ vehicles depart and in the second iteration $|x_2^1| + |x_5^1|$ vehicles depart.

Case 2: For the second case we have that $|x_2^1| + |x_5^1| + |x_1^1| > n$. In the first iteration we still have that $|x_1^1|$ vehicles depart and $|x_5^1|$ vehicles remain in the centre. These $|x_1^1|$ vehicles cover one of the double frequency lines. Then in the second iteration we have that $n - |x_1^1| = |x_1^2|$ vehicles are allowed to depart. Leaving $|x_5^2| > 0$ vehicles in the centre.

Now all edges have been visited exactly once. Giving us $|x_1^1| + |x_2^1| + |x_5^1| - |x_5^2| = n$ but then we also know that $|x_3^1| + |x_4^1| + |x_5^2| = n$ as we have $2n$ vehicles in total. Now if in the third iteration the $\min\{|x_1^1|, |x_3^1| + |x_5^2|\}$ vehicles that are allowed to depart serve one of the double frequency lines, we have the same situation as in case 1, only two iterations shifted.

If it does not serve any of the double frequency lines, we know all vehicles are allowed to depart. If we name the lines according to their rotor order, where the double frequency line has two places in the order, lines 0 to $|x_3^1| + |x_5^2|$ are served in the third iteration. Then in the fourth iteration there are enough vehicles to serve all lines from $|x_3^1| + |x_5^2|$ to 0, but this includes both of the double frequency lines, so we split the group of vehicles with size $|x_4^1|$ in the centre into two groups of size $|x_1^4|$ and $|x_5^4|$ that respectively depart and stay behind. Now lines $|x_3^1| + |x_5^2|$ to $n - |x_5^4|$ are served.

Then in the fifth iteration we have $|x_1^1| + |x_5^4|$ vehicles in the centre, as we know that $|x_1^1| >$

$|x_3^1| + |x_5^2|$, not all of these vehicles are allowed to leave based on their headway. We can serve exactly all lines that were not served in the previous iteration, serving lines $n - |x_5^4|$ to $|x_3^1| + |x_5^2|$ which contains exactly one of the double frequency lines. We say that $|x_1^5|$ vehicles have departed and $|x_5^5|$ remain in the centre.

Finally in iteration six, we now have $|x_1^2| + |x_5^5|$ vehicles start in the centre and at most $|x_1^4|$ are allowed to depart. We can show that $|x_1^2| + |x_5^5| \geq |x_1^4|$, so exactly $|x_1^4|$ vehicles depart, serving exactly one of the double frequency lines. Because of that in the next iteration the final part of the $2n$ vehicles in total can depart, serving $n - |x_1^4|$ lines, repeating the pattern of alternatingly serving $|x_1^4|$ and $n - |x_1^4|$ lines indefinitely.

Now for showing that $|x_1^2| + |x_5^5| \geq |x_1^4|$, we have that $|x_1^4| + |x_1^5| = n = |x_5^5| + |x_1^3| + |x_1^2|$. Where we know that $|x_1^5| = n - |x_1^4|$ and $|x_1^3| \leq n - |x_1^4|$ as they depart in consecutive iterations, so from this we can conclude that $|x_1^2| + |x_5^5| \geq |x_1^4|$.

So also in this case stabilisation occurs after a constant number of iterations, from which we can conclude that the scenario where all lines have travel time 2 and there is one line with frequency two and all other lines have frequency one stabilises in constant time after the stock is empty.

7.4 Other frequencies

As we have already mentioned in Section 4, it can be that the order of the edges matters for the performance. In this section we will illustrate this with two examples.

For the situation where we have one centre node and four end nodes, three lines with frequency 1 and then one line with frequency 3 and all travel times are 1, the order of the lines in the cyclic order starts to matter. Per periodic motion we have 12 units that need to be driven every 3 iterations. So 4 vehicles should suffice. However if we put all three of the triple frequency line are next to each other for example, we have that the middle one of these will be departing alone in an iteration, as it cannot be in the same as its predecessor and successor. Because of this there is an iteration of the three in which at least three vehicles need to depart, but that means the iteration before and after it at most one vehicle can depart. Letting in total only five vehicles depart instead of the necessary six. So for this cyclic order there is no periodic motion in which all headways are made.

Whereas when we put the triple and single frequency lines alternatingly in the cyclic order, it can be that every iteration two vehicles depart, making four vehicles enough to make all the desired headways. However, the system does enter a periodic motion of four iterations. This periodic motions is illustrated in Figure 8.

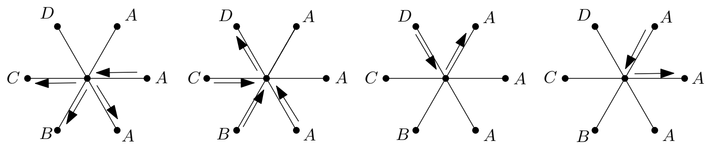

Figure 8: The periodic motion for all frequency 3 edges next to each other.

We can also consider the situation where we have one line with frequency 8, one line with frequency 4 and one line with frequency 1. If there are enough vehicles available, one would say that we can serve all lines in eight iterations. However, the configuration matters here as well. If we have the situation where all eight frequency 8 lines follow each other in the rotor order as is shown in 9, each of them has their own iteration, implying that the six that are enclosed by the others are the only lines served in that iteration. But then if we also have the four frequency 4 lines grouped together, the middle two of these also have their own iteration, so we need at least ten iterations to serve all edges.

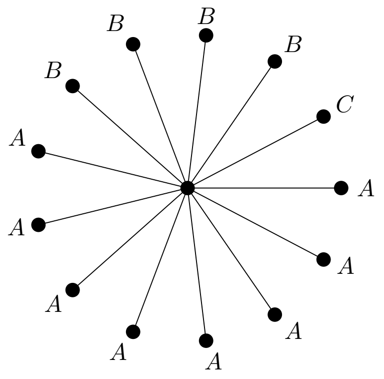

Figure 9: Rotor for frequencies 8, 4 and 1.

7.5 Simulation results

To see how these results hold in practice we have run some simulations on star networks.

First we consider the situation where we have one central station and $n + 1$ end stations for $n \in \{2, \dots, 1000\}$. Here n end stations have an edge with frequency 1 and the corresponding headway is n, the other end station has frequency n and headway 1. The travel times for all edges are 1. As we have already seen for $n = 3$ the cyclic order of the edges matters for the performance. We start by considering the situation where we alternatingly have a frequency 1 and a high frequency line. The rotor for frequency 2, 3 and 4 is described in Figure 10, 11 and 12. All four vehicles start in a frequency 1 end node. Note that if we have one frequency n edge and n frequency 1 edge, four vehicles can be exactly enough to drive all of those in n iterations.

If we recall the definition of a stable situation as being in a periodic motion which is repeated, we perform the algorithm in iterations until we reach such a stable situation. In the stable situation the stock node is empty for $n - 1$ iterations. So we stop the simulation when the stock node has been empty for $n - 1$ iterations without interruptions. As when the stock has been empty that long it may immediately send a vehicle that arrives back to the centre. After that the same pattern can be followed as the one we had after the previous departure from the stock node. In Figure 13 we can see how many iterations it takes before we reach this stable situation. Here the value of n is shown on the x-axis and the number of iterations on the y-axis.

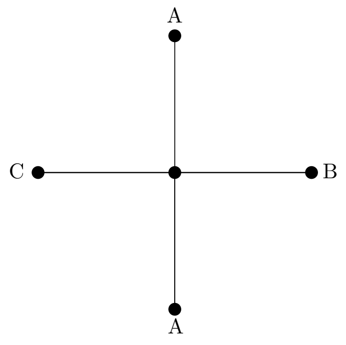

Figure 10: Frequency 2

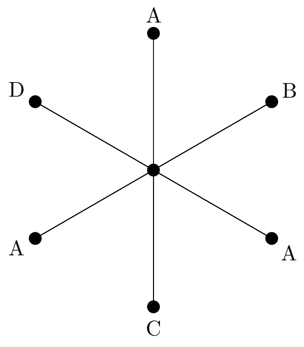

Figure 11: Frequency 3

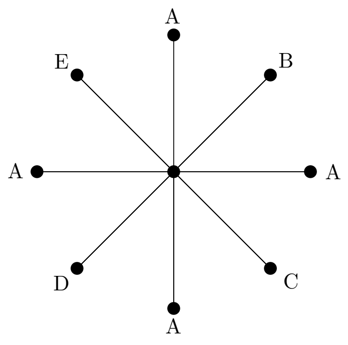

Figure 12: Frequency 4

We are also interested in what the stable situation looks like. For $n \geq 3$ there is a difference in behaviour between odd and even n. Note that in all cases vehicles that depart from the centre, will return in the next iteration. For even n we see that alternatingly one and three vehicles depart from the centre. As for even n the number of edges is a multiple of 4 this nicely forms a repeating motion, resulting in a periodic motion of n iterations. For n is odd two vehicles leave the centre in every iteration. Also resulting in a periodic motion of n iterations.

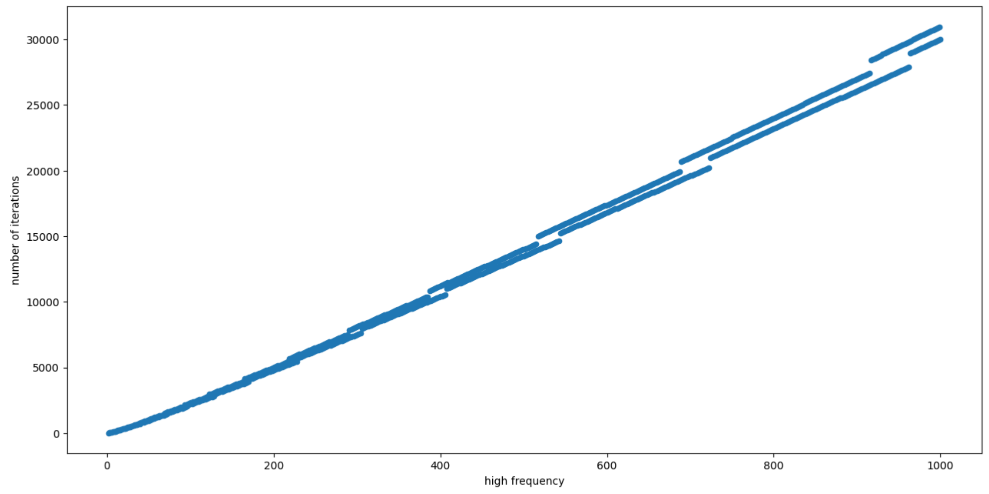

Figure 13: Number of iterations until stable state for alternating high and low frequency.

As we can see in Figure 13 the values are divided into two lines as well, when separating the even and odd values for the frequencies we can see that one of the lines comes from the even values and the other from the odd values.

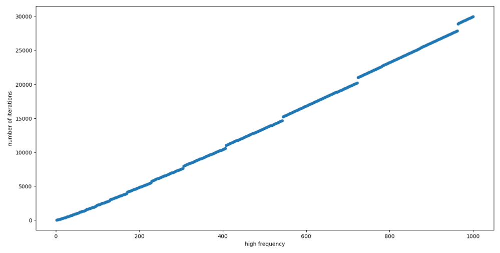

Figure 14: Even values from Figure 13

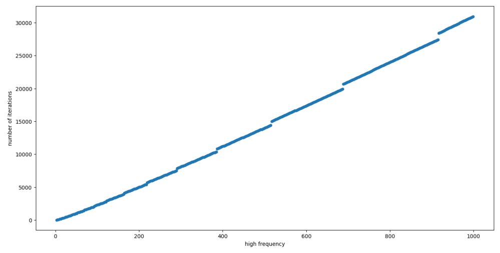

Figure 15: Odd values from Figure 13

Next we consider the same case but with the edges in a different rotor order. Now all the high frequency edges follow each other in the rotor order. Examples of this for frequencies 2, 3 and 4 can be found in Figures 16, 17 and 18. All vehicles start in the frequency 1 end node that follows the high frequency nodes in the rotor order. Again we performed the algorithm until we have reached a stable state. The results are shown in Figure 19. The high frequency n can be found on the x-axis and number of iterations until a stable state is reached can be found on the y-axis.

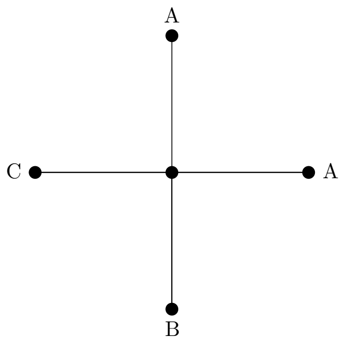

Figure 16: Frequency 2

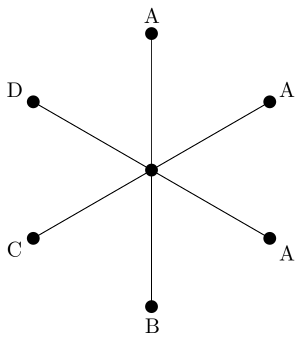

Figure 17: Frequency 3

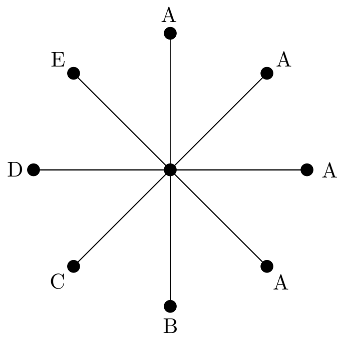

Figure 18: Frequency 4

For the case where all higher frequency lines follow each other the stable situation is harder to describe. There are two general things that stand out. The first being that n iterations is not enough to serve all nodes, as often only one vehicles can depart if the rotor is in the middle of the high frequency edges part. The second thing is that never more than three vehicles depart at the same time, so every iteration there is at least one vehicle departing from the centre.

Finally we have also the considered the case with alternatingly frequency n and frequency 1 lines, with travel time n per edge. The results can be found in 20.

In this case we need $4n$ vehicles to serve all lines in n iterations instead of 4 vehicles. So it makes sense that when the results for travel time 1 were that it takes slightly more than linear in the number of stations iterations before we reach a stable state, that for travel time n this is sightly worse than quadratic as we have n times as many vehicles.

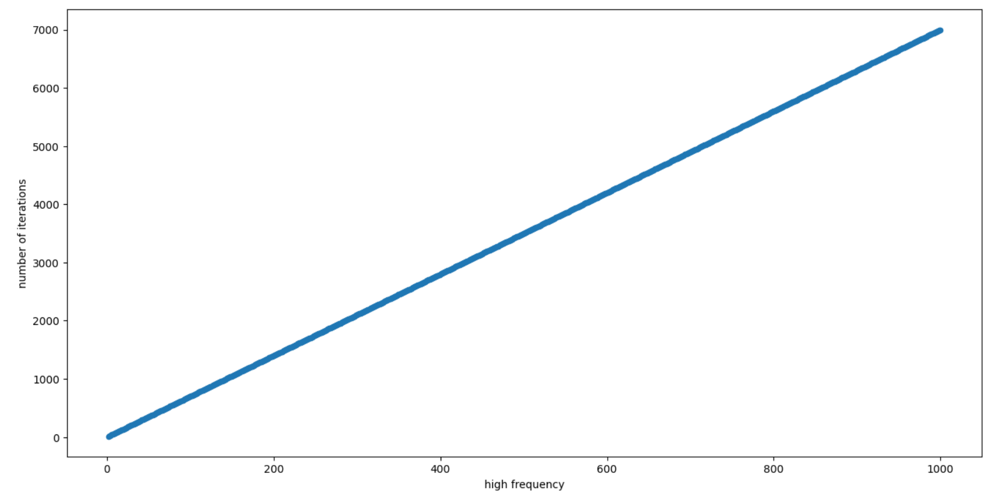

Figure 19: Number of iterations until stable state for all high frequency lines together.

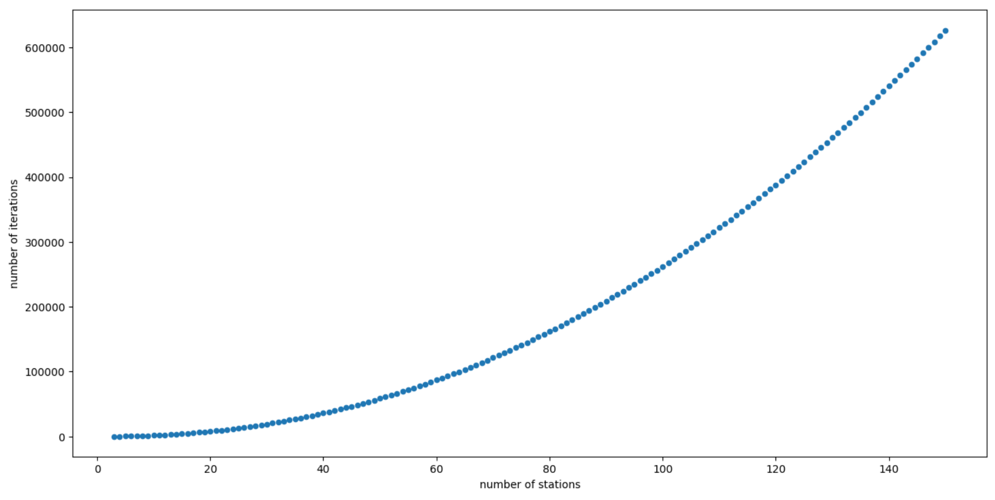

Figure 20: Number of iterations until stable state for travel time equal to the high frequency.

8 Conclusion

For the path graph we have not been able to improve on the generally known $O(n^3)$ bound. However for the star graph stabilisation occurs in $O(n\sqrt{n})$ in the single frequency case. When we add one double frequency line, stabilisation occurs in constant time after all vehicles have left the stock node. Where we also know that at least once per cover time a vehicle will leave the stock.

The simulation results for the path graph confirm what we already got from the theoretical results. For the star graph the most remarkable result is the case where all high frequency lines are grouped together and the travel time is 1 for all edges, here we see that a stable state is reached in time linear in the number of stations in the network.

There are many more cases we could have simulated, for example with combinations of multiple frequencies. It is also interesting to see if we can prove the results we have seen in the simulations using theory, especially the linear result for when all high frequency lines are grouped together.

References

[1] Juan Argote-Cabanero, Carlos F Daganzo, and Jacob W Lynn. Dynamic control of complex transit systems. *Transportation Research Part B: Methodological*, 81:146–160, 2015.

[2] John J Bartholdi III and Donald D Eisenstein. A self-coördinating bus route to resist bus bunching. *Transportation Research Part B: Methodological*, 46(4):481–491, 2012.

[3] Jérémie Chalopin, Shantanu Das, Paweł Gawrychowski, Adrian Kosowski, Arnaud Labourel, and Przemysław Uznański. Limit behavior of the multi-agent rotor-router system. In *International Symposium on Distributed Computing*, pages 123–139. Springer, 2015.

[4] Mark M Dekker, Rolf N van Lieshout, et al. A next step in disruption management: Combining operations research and complexity science. *Public Transport*, pages 1–22, 2021.

[5] Dariusz Dereniowski, Adrian Kosowski, Dominik Pajak, and Przemysław Uznański. Bounds on the cover time of parallel rotor walks. *Journal of Computer and System Sciences*, 82(5):802–816, 2016.

[6] Shidong Liang, Shuzhi Zhao, Chunxiu Lu, and Minghui Ma. A self-adaptive method to equalize headways: Numerical analysis and comparison. *Transportation Research Part B: Methodological*, 87:33–43, 2016.

[7] Vyatcheslav B Priezzhev, Deepak Dhar, Abhishek Dhar, and Supriya Krishnamurthy. Eulerian walkers as a model of self-organized criticality. *Physical Review Letters*, 77(25):5079, 1996.

[8] Rolf N Van Lieshout. Integrated periodic timetabling and vehicle circulation scheduling. *Transportation Science*, 55(3):768–790, 2021.

[9] Rolf N van Lieshout, Paul C Bouman, and Dennis Huisman. Determining and evaluating alternative line plans in out-of-control situations. *Transportation science*, 54(3):740–761, 2020.

[10] Rolf N van Lieshout, Paul C Bouman, Marjan van den Akker, and Dennis Huisman. A self-organizing policy for vehicle dispatching in public transit systems with multiple lines. *Transportation Research Part B: Methodological*, 152:46–64, 2021.

[11] Peter Winkler and David Zuckerman. Multiple cover time. *Random Structures & Algorithms*, 9(4):403–411, 1996.

[12] Vladimir Yanovski, Israel A Wagner, and Alfred M Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.

[13] Shuyang Zhang and Hong K Lo. Two-way-looking self-equalizing headway control for bus operations. *Transportation research part B: methodological*, 110:280–301, 2018.