Vehicle Scheduling Based on a Line Plan

Rolf N. van Lieshout
Erasmus University Rotterdam
Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands.
vanlieshout@ese.eur.nl

Paul C. Bouman
Erasmus University Rotterdam
Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands.
bouman@ese.eur.nl

Abstract

We consider the following problem: given a set of lines in a public transportation network with their round trip times and frequencies, a maximum number of vehicles and a maximum number of lines that can be combined into a vehicle circulation, does there exist a set of vehicle circulations that covers all lines given the constraints. Solving this problem provides an estimate of the costs of operating a certain line plan, without having to compute a timetable first. We show that this problem is NP-hard for any restriction on the number of lines that can be combined into a circulation which is equal to or greater than three. We pay special attention to the case where at most two lines can be combined into a circulation, which is NP-hard if a single line can be covered by multiple circulations. If this is not allowed, a matching algorithm can be used to find the optimal solutions, which we show to be a $\frac{16}{15}$-approximation for the case where it is allowed. We also provide an exact algorithm that is able to exploit low tree-width of the so-called circulation graph and small numbers of vehicles required to cover single circulations.

2012 ACM Subject Classification Applied computing → Transportation, Mathematics of computing → Graph algorithms

Keywords and phrases Vehicle scheduling, integrated railway planning, (fractional) matching, treewidth.

Digital Object Identifier 10.4230/OASIcs.ATMOS.2018.15

1 Introduction

Traditionally, the planning of public transport services occurs in a number of steps. First, a *line plan* is constructed where service routes, usually referred to as lines, are selected such that high quality service is provided to the customers [5, 14]. In the second step, a *timetable* is constructed that specifies the departure and arrival times along the stops of all lines [6, 10]. In the final step, *vehicles* and possibly *human resources* are planned as they are necessary resources to execute the services [1, 8, 11]. As the individual scheduling steps are already quite challenging, the sequential planning approach is traditionally applied because an integrated approach is computationally not tractable. The disadvantage of the sequential approach is that the objectives of the subsequent steps are not taken into account when the prior steps are solved. In particular, the line plan and timetable are usually optimized based on passengers' convenience, while the vehicle schedule is optimized based on the operator costs. Therefore, the optimal solution for the combined problem is likely to be missed.

Recently, a number of authors have proposed ideas to integrate the separate planning steps. One example, the *eigenmodel* [15], replaces a fixed order with an iterative approach that takes a different route through the separate steps, controlling both the passengers'

convenience and the operator costs during the process. Another approach [12] incorporates penalties during the line planning phase for lines which can not be covered efficiently by a vehicle in a periodic timetable. That is, assuming the cycle time is 60 minutes, a line with frequency one for which a round trip takes 54 minutes (a downtime of 6 minutes) is given a low penalty, while a line with frequency one for which a round trip takes 65 minutes (a downtime of 55 minutes) is given a very high penalty.

In this paper, we consider the construction of vehicle schedules based on the line plan without the intermediate step of constructing a timetable. One goal of this is to quickly assess whether a line plan can be operated using a small number of vehicles. This allows public transport operators to detect potential inefficiencies early in the planning process, without having to compute a timetable first. The novel aspect of our approach is that we explicitly consider the possibility to combine lines into larger vehicle circulations. To illustrate, while a line that takes 65 minutes with a period of 60 minutes may seem inefficient by itself, it may be a good option if we can combine it with a line of 55 minutes. Although combinations of lines can help to reduce the number of vehicles required to operate a line plan, large and complex combinations of lines may result in greater dependencies between the operations of the different lines. Therefore, we provide a detailed examination of cases where at most two lines can be combined in a vehicle schedule.

The remainder of this paper is organized as follows. In Section 2 we introduce the vehicle circulation scheduling problem. In Section 3 we study the computational complexity of the general case. In Section 4 we study the special case where only two lines can be combined in a circulation. We conclude and discuss ideas for future research in Section 5.

2 Problem Formulation

In this paper, we assume a *line plan* is already given and want to determine the minimum number of vehicles that are required to operate the line plan *without* the intermediate step of constructing a timetable. For the line plan, we have a fixed time period denoted by T and a set of lines $\mathcal{L}$ where a line $\{v, u\} \in \mathcal{L}$ is an unordered pair of terminal stations of the line. The *line graph* $L = (V, \mathcal{L})$ has terminal stations V as vertices and the lines as edges. In the line plan each line $l \in \mathcal{L}$ has a round trip time t_l and an integer frequency f_l assigned to it. The round trip times specified by the line plan should at least include the minimum driving and dwelling times required to execute the line, but can also include some slack to make operations more robust against disruptions. The frequency defines the number of times the line service must be executed by a vehicle within each time period of length T.

If vehicles are only allowed to operate a single line, the number of vehicles required for line l equals $\left\lceil \frac{t_l}{T} f_l \right\rceil$. In order to reduce the number of vehicles required to operate the line plan, public transport operators may consider *circulations*, which are a combination of lines that can be executed by a single vehicle. Formally, a circulation $c \subseteq \mathcal{L}$ is a set of lines that can be operated by one or more vehicles. We allow lines to be contained more than once in the set, since this can be relevant if the line has a frequency greater than 1. We call the number of times a line l is contained in a circulation the *multiplicity* of l in c. Furthermore, we assume that a summation over the lines in the circulation includes a line multiple times if it has a multiplicity greater than 1. An example of a situation where we want to assign the same line to a circulation multiple times, is a line with a round trip time of $\frac{T}{2}$ and a frequency of 2. In that case, we want to consider a circulation where we execute the line twice during each period.

For this paper, we only consider circulations c such that the lines it contains form a

connected subgraph of L. As a consequence of this, the time t_c needed to perform a single round trip of a circulation c can be expressed as $t_c = \sum_{l \in c} t_l$. Furthermore, a circulation c corresponds to a directed Eulerian tour in $\overleftrightarrow{L}$, where $\overleftrightarrow{L}$ is the *directed line graph*, which is a a symmetric directed graph derived from L where each edge of L is replaced by two arcs, one for each direction. Let us now consider the correspondence between a connected subset of L and the directed cycle in $\overleftrightarrow{L}$.

▶ **Lemma 1.** *A connected subset $c \subseteq \mathcal{L}$ of lines in the line graph L corresponds to a directed Eulerian sub-graph in the directed line graph $\overleftrightarrow{L}$. Thus, there always exists a directed cycle in $\overleftrightarrow{L}$ that visits all arcs that correspond to both directions of the lines in c a number of times that is exactly equal to the multiplicity of the lines in the circulation.*

Proof. A directed graph contains a directed Eulerian cycle if two conditions hold: (1) for every vertex the in-degree is equal to the out-degree, and (2) the graph is strongly connected. Since the graph $\overleftrightarrow{L}$ is symmetric, each vertex must have one outgoing arc for each incoming arc, and thus condition (1) always holds. As c is a connected subset of lines, the corresponding lines in $\overleftrightarrow{L}$ must be connected as well. Since the graph is symmetric, this implies that it is also strongly connected. ◀

Although more general concepts of circulations that do not enforce connectivity can be considered, these would require dead-heading of vehicles as part of a line plan. While public transport operators have to use dead-heading when operations start up, or frequencies of lines are changed throughout the day, it is usually avoided as much as possible by public transport operators during regular operations. As we focus on regular operations, we consider those generalized concepts of circulations beyond the scope of this paper.

In the problem we consider we do not only need to decide which circulations should be used, but we also have to decide how many vehicles must be assigned to each circulation to cover the constraints of the line plan. Since a circulation c can have a round trip time that is larger than T, we may need to assign multiple vehicles to it in order to enforce that every line in the circulation is covered during every period. We define the number of vehicles required to perform the circulation once in every period as $k_c = \lceil \frac{t_c}{T} \rceil$. If we would assign fewer vehicles than k_c to a circulation, the vehicle is not finished with its circulation when a new period starts and as a consequence no vehicle executes the circulation during some periods. Furthermore, an upper bound on the number of vehicles that can be assigned to a circulation c is given by the expression $\min_{l \in c} f_l k_c$, as assigning more vehicles to the circulation would imply that the line where the minimum was attained is executed with greater frequency than the line plan prescribes. In the special case that all frequencies are 1, k_c itself is an upper bound on the number of vehicles that can be assigned to c.

Note that we do not enforce that each line is covered by a single circulation. For example, suppose we have two lines a and b with round trip times $t_a = t_b = 30$, but different frequencies $f_a = 3$ and $f_b = 1$, with a period time of $T = 60$. The most efficient way to cover these lines is to have one vehicle circulation that executes line a two times each period, while a second vehicle alternates between line a and line b, executing both lines once each period. We investigate a strict version of the problem where a line can only be covered by a single circulation in Section 4.2.

The goal of the problem studied in this paper is to find a set of circulations and the number of vehicles assigned to each circulation such that all lines are covered. From a practical point of view it is desirable that the selected circulations do not contain too many lines, as this creates significant dependencies in the operations that make the operations extremely sensitive to minor disruptions. When a disruption occurs somewhere in a circulation, all

subsequent lines in the circulation are affected. Furthermore, very large circulations can only be operated in practice if the timetables of all the lines in the circulation are synchronized. This may lead to problems in the timetabling phase, especially if you want to offer good transfer possibilities to passengers who want to follow different paths than the vehicles.

To avoid large circulations, we consider a restriction on the maximum number of lines than can be included in a circulation. We call a circulation c an α-circulation if the number of unique lines in c is α. A 1-circulation is also referred to as a *fixed circulation* and a 2-circulation is also referred to as a *combined circulation*. We introduce an input parameter κ that restricts the number of unique lines that can be combined in a single circulation. With this concept clearly defined, we can introduce the decision variant of our problem in a formal way:

Vehicle Circulation Scheduling Problem (VCSP)
Instance: A line graph $L = (V, \mathcal{L})$, a maximum number of of unique lines that are allowed to exist in a single circulation κ and a maximum number of vehicles z
Question: Does there exist a set of circulations C, with for each circulation $c \in C$ a value assigned to the integer decision variable $\theta_c \in \mathbb{N}$ that indicates how many vehicles are assigned to circulation c, such that:
(1) the circulations cover all lines in every period, i.e. $\forall l \in \mathcal{L} : f_l = \sum_{c \in C : l \in c} \frac{\theta_c}{k_c}$,
(2) there are no α-circulations in C with $\alpha > \kappa$, and
(3) at most z vehicles are required to execute all circulations, i.e. $\sum_{c \in C} \theta_c \leq z$.

The optimization version of this problem seeks to find the smallest z for a given line graph and a given parameter κ, such that there exists a set of circulations C with integer vehicle assignments θ that satisfy the conditions.

3 Computational Complexity

First, we consider a lower bound on the number of vehicles that is needed in a given line graph. Since we are not allowed to use dead-heading, we must consider the connected components of a line graph separately, as no vehicle will be able to move from one component to another. Thus without loss of generality we assume that a line graph is connected, since if it is not we can decompose the problem into independent sub-problems. A lower bound on the number of vehicles required can be computed by dividing the total running time by the cycle time. As no fractional vehicles can be used, we can round the number of vehicles up. This gives the following necessary condition to check if the instance can possibly be a YES-instance:

$$z \geq \left\lceil \frac{\sum_{l \in \mathcal{L}} t_l \cdot f_l}{T} \right\rceil \tag{1}$$

If we have an instance of the problem where $|\mathcal{L}|$-circulations are allowed, i.e. $\kappa \geq |\mathcal{L}|$, this lower bound can be obtained by a single circulation that contains all lines as many times as their frequency dictates. Thus all such instances are YES-instances.

If we are only allowed to used fixed circulations, i.e. $\kappa = 1$, the only way to cover all lines is to use a fixed circulation for each line. In this case an instance is a YES-instance if and only if z is sufficient for the sum over all fixed circulations:

$$z \geq \sum_{l \in \mathcal{L}} \left\lceil \frac{t_l \cdot f_l}{T} \right\rceil \tag{2}$$

▶ **Theorem 2.** *Any instance with $\kappa \geq |\mathcal{L}|$ for which the condition of Equation 1 holds is a*

YES-instance. Any instance with $\kappa = 1$ is a YES-instance if and only if the condition of Equation 2 holds.

As we noted earlier, circulations that do not contain too many lines are preferred as they have many advantages. However, the number of vehicles required with fixed circulations can be significantly greater than when any circulation is allowed. We can see this via application of the following general identity for sums over ceiling functions. Suppose we have a sequence $a_1, \dots a_n$ of n numbers with $n \geq 2$, then it is straightforward to show that

$$\sum_{i=1}^{n} \lceil a_i \rceil - \left\lceil \sum_{i=1}^{n} a_i \right\rceil \leq n - 1 \tag{3}$$

Thus, we can see that the difference between the lower bound of Equation 2 and Equation 1 can become as large as $|\mathcal{L}| - 1$. If we allow 2-circulations, we are able to halve the number of terms in Equation 2 which halves the worst case gap. It thus can be very beneficial to consider restrictions on the circulation κ that are small but strictly greater than one. Unfortunately, for any case with a fixed $\kappa \geq 3$, we can show that the resulting problem becomes NP-hard.

▶ **Theorem 3.** *For any fixed $\kappa \geq 3$, the Vehicle Circulation Scheduling Problem is* NP*-hard.*

Proof. We show this by reduction from 3-partition. Let $S = \{s_1, s_2, \dots, s_m\}$ be a set of integers such that $\sum_{i=1}^{m} s_i = \frac{m}{3}B$ and $\forall 1 \leq i \leq m : \frac{B}{4} < s_i < \frac{B}{2}$. A 3-partition instance is a YES-instance if it is possible to partition S into $n = \frac{m}{3}$ triplets $S_1, S_2, \dots S_n$ such that each triplet sums to B.

For the case where $\kappa = 3$, we reduce the 3-partition instance to a line graph $L = (V, \mathcal{L})$ where we have a central hub station $v_0 \in V$ and m external stations $v_1, \dots, v_m \in V$. This line plan must be periodically operated in a period of $T = B$ time units. Furthermore we have a set of m lines where $l_i \in \mathcal{L} = \{v_0, v_i\}$, with frequency $f_i = 1$ and a round trip and total time equal to s_i. As the sum of the round trip times is exactly mB, the only way to execute this line plan with m vehicles is to have every circulation take B time. Otherwise, at least one circulation will require two vehicles. Thus we set $z = m$. If the 3-partition instance is a YES-instance, we can use the triplets to create 3-circulations with this precise property. If the 3-partition instance is a NO-instance, we can not nicely divide the lines over 3-circulations and thus need at least one additional vehicle.

For cases where $\kappa > 3$, we scale the 3-partition instance by setting $s'_i = 4 \cdot \kappa \cdot s_i$ and $B' = (\kappa - 3) + 4 \cdot \kappa \cdot B$. This way we make sure that $\frac{B'}{4} > \kappa - 3$. We now introduce a set of $\kappa \cdot n$ lines where lines $l_1, \dots, l_m$ have round trip times $s'_1, \dots, s'_m$ and lines $l_{m+1}, \dots, l_{\kappa \cdot n}$ all have round trip time of 1. We define $1 + \kappa$ terminal stations and let lines i connect terminal stations $\{v_0, v_i\}$ in the same structure as the $\kappa = 3$ case. By construction, only circulations that consist of $\kappa - 3$ lines with round trip time 1 and three other lines can sum up to B'. This way it is enforced that it is a YES-instance if and only if the 3-partition instance is a YES-instance. ◀

4 Fixed and Combined Circulations

Of special interest is the case of $\kappa = 2$ where we are only allowed to have fixed and combined circulations, as this gives us some flexibility to decrease the number of vehicles required to operate the line plan, while we still keep the number of lines in a circulation low. Since all finite cases with $\kappa > 2$ are NP-hard, the $\kappa = 2$ case is also of particular interest from a theoretical perspective.

For the remainder of this section, we restrict ourselves to instances of the vehicle circulation scheduling problem where the frequency of each line is 1. Although this restriction may seem unrealistic, we can approximate instances with higher frequencies either by splitting up a line l with a frequency higher than 1 into f_l lines with frequency 1 and the same characteristics or by increasing the round trip times as a function of the frequencies. The straightforward approach increases the round trip times based on the frequency, i.e. the new round trip time becomes $f_l \cdot t_l$. More sophisticated approaches can add slack to model the periodicity in more detail in order to increase the probability that a regular timetable exists, possibly at the cost of requiring more vehicles to execute the line plan.

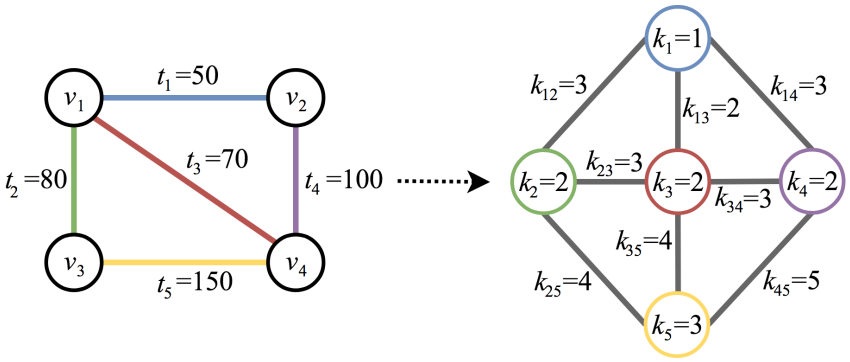

Figure 1 Example of a line graph with round trip times and the corresponding circulation graph. The cycle time T is 60.

In case $\kappa = 2$ and $f_l = 1$ for all lines, we can represent an instance of the VCSP by a *circulation graph* $G = (\mathcal{L}, E)$. In this graph, the lines are the vertices and the edges are the circulations. The set of edges consists of edges for the fixed circulations E_1 and of the 2-circulations E_2, thus $E = E_1 \cup E_2$. The set E_1 contains a self loop for every line $l_i \in \mathcal{L}$. The set E_2 contains an edge between two lines l_i and l_j if they have a common terminal station. To ease notation, we denote a circulation $\{l_i\} \in E_1$ simply as l_i.

An example of a VCSP instance represented by a circulation graph is given in Figure 1. On the circulation graph, we also depict the k_c value of each circulation c. For the self loops, these values are depicted inside the nodes to make the graph more clear. For example, line 3 has a round trip time of 70 minutes, so $k_3 = \lceil \frac{70}{60} \rceil = 2$. Line 4 also needs 2 vehicles if it is performed in a fixed circulation, but if lines 3 and 4 are combined only 3 vehicles are required to operate both lines since $k_{34} = \lceil \frac{70+100}{60} \rceil = 3$.

Using the circulation graph, we can formulate the following optimization problem for the VCSP:

$$\nu(\mathcal{L}) = \min_{\theta} \quad \sum_{c \in E} \theta_c \tag{4}$$

$$\text{s.t.} \quad \frac{1}{k_l}\theta_l + \sum_{c \in E_2 | l \in c} \frac{1}{k_c}\theta_c = 1 \quad \forall l \in \mathcal{L}, \tag{5}$$

$$\theta_c \geq 0 \text{ and integer} \qquad \forall c \in E, \tag{6}$$

The objective is to minimize the number of used vehicles. We now consider how to rewrite this problem to a maximization problem where the goal is to maximize the number of *saving* circulations. First note that we can rewrite the summation in the objective of Equation 4 by splitting the summation over E into summations over E_1 and E_2, and that by definition a

summation over E_1 is equal to a summation over $\mathcal{L}$. We rewrite the objective as follows:

$$\min_{\theta} \sum_{l \in \mathcal{L}} \theta_l + \sum_{c \in E_2} \theta_c \tag{7}$$

In the next step we make use of Constraints 5, which state that a line is included in a sufficient number of circulations. As these constraints imply that $\theta_l = k_l - \sum_{c \in E_2 | l \in c} \frac{k_l}{k_c} \theta_c$, we can substitute the left term to obtain

$$\min_{\theta} \sum_{l \in \mathcal{L}} \left(k_l - \sum_{c \in E_2 | l \in c} \frac{k_l}{k_c} \theta_c \right) + \sum_{c \in E_2} \theta_c \tag{8}$$

Note that the double summation over $l \in \mathcal{L}$ and $c \in E_2 | l \in c$ can also be written as a double summation over $c \in E_2$ and $l \in c$. Rewriting the double summation and reshuffling some terms gives:

$$\sum_{l \in \mathcal{L}} k_l + \min_{\theta} \sum_{c \in E_2} \left(\theta_c - \sum_{l \in c} \frac{k_l}{k_c} \theta_c \right) \tag{9}$$

In the last step we factor out $\frac{\theta_c}{k_c}$ and Equations 4 – 6 are written as:

$$\nu(\mathcal{L}) = \sum_{l \in \mathcal{L}} k_l + \min \left\{ \sum_{c \in E_2} \left[k_c - \sum_{l \in c} k_l \right] \frac{\theta_c}{k_c}, \text{ s.t. } (5) - (6) \right\} \tag{10}$$

If we now apply Equation 3, it can be seen that $\left[k_c - \sum_{l \in c} k_l\right]$ equals either -1 or 0. If the term is -1, we call the circulation *saving*, otherwise we call it *non-saving*. For example, in Figure 1, circulation $\{3, 4\}$ is saving, while circulation $\{1, 2\}$ is non-saving. If we let the set of all saving circulations be denoted as E_2^S, we have that $\nu(\mathcal{L}) = \sum_{l \in \mathcal{L}} k_l - \sigma(\mathcal{L})$ where $\sigma(\mathcal{L})$ is the *savings problem* defined as follows:

$$\sigma(\mathcal{L}) = \max_{\theta} \left\{ \sum_{c \in E_2^S} \frac{\theta_c}{k_c} \text{ s.t. } (5) - (6) \right\} \tag{11}$$

As such, it can be observed that minimizing the number of vehicles is equivalent with maximizing the savings over a vehicle schedule that only uses fixed circulations. In the remainder of this section, we use this observation to give a proof of the NP-hardness of the VCSP with $\kappa = 2$ and $f_l = 1$ for all lines, and to develop an exact algorithm and an $\frac{16}{15}$-approximation algorithm.

4.1 NP-hardness

Our proof depends on the fact that we can construct an arbitrary circulation graph from the line graph if it is accompanied by auxiliary restrictions on which 2-circulations are allowed, which are not part of the formal input to the VCSP. This can be seen from the fact that a star-shaped line graph translates to a complete circulation graph, since we can combine all pairs of lines. If we can provide auxiliary restrictions on which combinations can be combined into circulations and which not, we have complete control over the structure of the circulation graph. In practice, such restrictions are realistic, since the possibility to combine lines into circulations does not only depend on the lines having a shared terminal station, but also on the precise layout of the infrastructure at the terminal station, and whether there exists a type of rolling stock that is able to operate both lines.

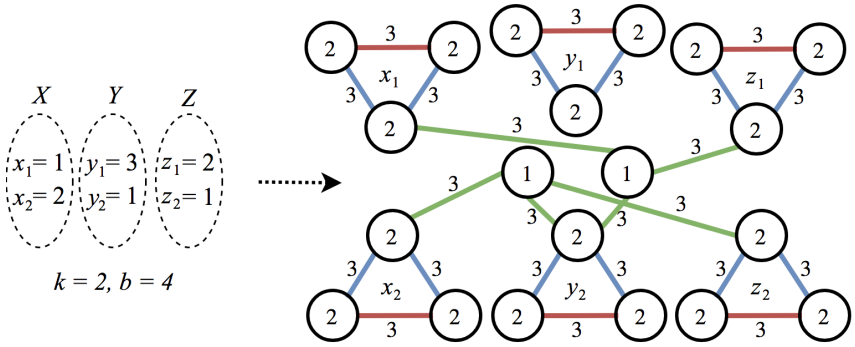

Figure 2 Transformation of a N3DM instance to a VCSP instance represented by a circulation graph.

▶ **Theorem 4.** *The vehicle circulation scheduling problem with $\kappa = 2$ and auxiliary restrictions on which 2-circulations are allowed is* NP*-hard.*

Proof. Our proof is based on a reduction from the NP-complete NUMERICAL 3-DIMENSIONAL MATCHING problem [9] to an instance of the vehicle circulation scheduling problem expressed by means of a circulation graph. Since each circulation graph can be generated based on a line graph with auxiliary restrictions, this is sufficient to prove the theorem.

The inputs to the N3DM are three multisets of integers X, Y, Z, each containing k elements, and a bound b. An N3DM instance is a YES-instance if there exist k disjoint triples (x, y, z) such that $x + y + z = b$ holds for every triple.

We transform this to the following instance of the VCSP. For every element in X, Y and Z we create three lines in $\mathcal{L}$, all with $k_l = 2$. The three lines can be combined in saving circulations ($k_c = 3$) such that they form a triangle. One of the three lines serves as the *connect line*, the other lines are referred to as *dummy lines*. For every triple (x, y, z) that sums up to b (all such triples can be found in polynomial time), a *triple line* is created with $k_l = 1$, which can be combined in non-saving circulations ($k_c = 3$) with the connect-lines corresponding to x, y and z. Letting μ denote the number of triples that sum up to b, the resulting VCSP instance has $9k + \mu$ lines. We call the generated instance a YES-instance, if the number of required vehicles is at most $14k + \mu$, equivalent to a saving $\sigma(\mathcal{L})$ of at least $4k$. In Figure 2, we visualized this transformation for a small N3DM instance.

We claim that the constructed VCSP instance is a YES-instance if and only if the N3DM instance is a YES-instance.

(if) Each disjoint triple (x, y, z) can be used to generate a saving of 4 by assigning 1 vehicle to each of the 3 circulations combining the triple line with the connect lines (green in Figure 2, 1 vehicle to each of the 6 circulations combining the triples lines with the dummy lines (blue) and 2 vehicles to each of the circulations combining dummy lines (red). In every triangle, this gives a saving of $\frac{4}{3}$, so the total saving generated by every disjoint triple equals 4. As such, if there are k disjoint triples, the total saving is $4k$ and the VCSP instance is indeed a YES-instance.

(only if) Since only the combined circulations in the triangle are saving, if the instance is a YES-instance, every triangle must generate a saving of $\frac{4}{3}$. This implies that in every triangle, the circulations combining the triple lines and dummy lines are assigned 1 vehicle (blue in Figure 2) and the circulations combining dummy lines are assigned 2 vehicles (red). As a consequence, for every connect line, one of the circulations connecting the line with a triple

line must be assigned 1 vehicle (otherwise the connect line would not be covered entirely). Next, note that a triple line cannot be partially performed by combined circulations. Hence, if one of the circulations combining a certain triple line and a connect-line is assigned a vehicle, all three such circulations must be assigned a vehicle. So, as every connect line is included in exactly one circulation with a triple line, and as every triple line is included in either zero or three combined circulations, there must be k triple lines that are connected with 3 connect lines. Clearly, the associated triples in the N3DM instance must be disjoint, hence the N3DM instance must also be a YES-instance. ◀

4.2 The Strict Vehicle Circulation Scheduling Problem

We define the *strict* version of the VCSP to state that each circulation c is either not executed at all, or executed by exactly k_c vehicles. We will now show that the strict version with $\kappa = 2$ and all frequencies equal to 1 can be solved exactly using an approach based on matching.

As in the non strict version, we have the relation that the minimum number of vehicles required under the strictness assumption, denoted as $\bar{\nu}(\mathcal{L})$, equals $\sum_{l\in\mathcal{L}} k_l - \bar{\sigma}(\mathcal{L})$, where $\bar{\sigma}(\mathcal{L})$ denotes the strict savings problem, obtained by replacing $\frac{\theta_c}{k_c}$ with the binary variable γ_c in the regular savings problem $\sigma(\mathcal{L})$:

$$\bar{\sigma}(\mathcal{L}) = \max_{\gamma} \quad \sum_{c\in E_2^S} \gamma_c \tag{12}$$

$$\text{s.t. } \gamma_l + \sum_{c\in E_2 | l\in c} \gamma_c = 1 \quad \forall l \in \mathcal{L}, \tag{13}$$

$$\gamma_c \in \{0,1\} \qquad \forall c \in E, \tag{14}$$

Constraints 13 now state that a line is either operated with a fixed circulation, or using one of the combined circulations. Since the objective does not contain the γ_c variables for the fixed circulations anymore, the γ-variables for these circulations can be viewed as slack variables for the Constraints 13. Furthermore, since the non-saving circulations have zero contribution to the objective, there always exists an optimal solution that does not contain any non-saving circulations. As a consequence, Constraints 13 can be rewritten as:

$$\sum_{c\in E_2^S | l\in c} \gamma_c \leq 1 \quad \forall l \in \mathcal{L} \tag{15}$$

Since the circulations in E_2^S contain precisely two lines, the resulting formulation is equivalent to a matching problem where we have to maximize the number of selected saving circulations. Thus, we can compute $\bar{\nu}(\mathcal{L})$ by computing the maximum matching in the graph that only contains the edges from E_2^S.

▶ **Theorem 5.** *The strict Vehicle Circulation Scheduling Problem with $\kappa = 2$, $f_l = 1$ for each line $l \in \mathcal{L}$ is solvable in polynomial time.*

4.3 The Matching Approximation

Since the strict vehicle circulation scheduling problem provides solutions that are also feasible for the regular problem, it can be applied as a heuristic. In this section we derive an approximation guarantee for this heuristic. Our approximation results are based on the observation that the savings problem is a maximization problem and that the linear programming relaxations of the savings problem and its strict version, denoted as $\sigma^{\text{LP}}(\mathcal{L})$

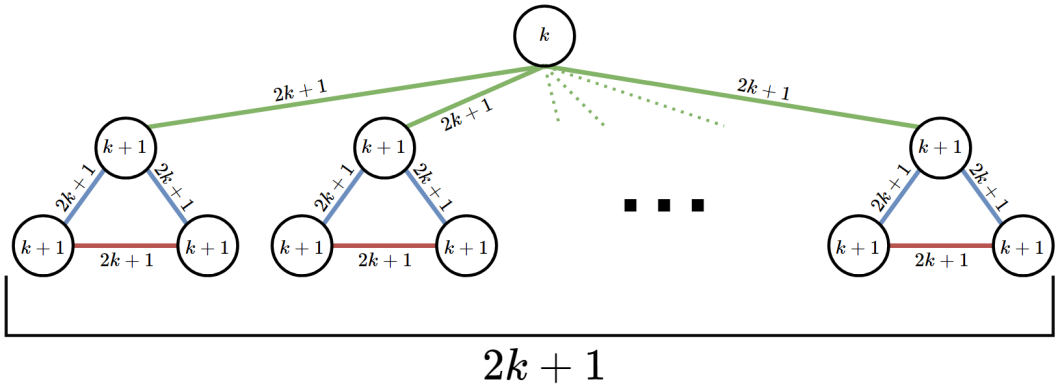

Figure 3 Circulation graph of the instance used to show that the bound of Theorem 6 is tight.

and $\bar{\sigma}^{\text{LP}}(\mathcal{L})$ respectively, are equal. This easily follows from the fact that the strict version is obtained from the non strict version by performing a linear variable substitution, which does not influence the value of the linear relaxation.

▶ **Theorem 6.** *For any line plan it holds that* $\bar{\nu}(\mathcal{L}) - \nu(\mathcal{L}) \leq \left\lfloor \frac{|\mathcal{L}|}{6} \right\rfloor$. *Furthermore, there exists an instance that attains this bound.*

Proof. Note that we can consider the difference between the savings instead of the difference between the number of vehicles, as $\bar{\nu}(\mathcal{L}) - \nu(\mathcal{L}) = \sum_{l \in \mathcal{L}} k_l - \bar{\sigma}(\mathcal{L}) - \sum_{l \in \mathcal{L}} k_l + \sigma(\mathcal{L}) = \sigma(\mathcal{L}) - \bar{\sigma}(\mathcal{L})$. For any graph it holds that the difference between the value of the maximum fractional matching and the value of the maximum matching is at most $\frac{n}{6}$, with n the number of nodes [7]. This implies that $\bar{\sigma}^{\text{LP}}(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) \leq \frac{|\mathcal{L}|}{6}$. Since the linear programming relaxations of the savings problem and its strict version are equal, it follows that $\sigma(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) \leq \sigma^{\text{LP}}(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) = \bar{\sigma}^{\text{LP}}(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) \leq \frac{|\mathcal{L}|}{6}$. Furthermore, the right hand side of this equation can be rounded down since the difference between savings must be integral.

To show that this bound is tight, consider the circulation graph depicted in Figure 3. The example contains $2k + 1$ triangles, where k is a positive integer, and one central node connected to all triangles. The circulations between the lines in the triangles are saving. The value $\bar{\sigma}(\mathcal{L})$ is equal to the size of the maximum matching in the graph induced by all saving circulations, i.e. the graph with only the $2k + 1$ triangles. Since we can pick only one circulation in every triangle, we have that $\bar{\sigma}(\mathcal{L}) = 2k + 1$. The optimal unrestricted solution is as follows. We can assign 1 vehicle to all green circulations, k vehicles to all blue circulations and $k + 1$ vehicles to all red circulations. The objective attained with this solution equals $\sigma(\mathcal{L}) = \sum_{c \in E_2^S} \frac{\theta_c}{k_c} = (2k + 1)(\frac{k+k+k+1}{2k+1}) = 3k + 1$.

Comparing the two objectives, we have that $\sigma(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) = k$. As the bound equals $\left\lfloor \frac{|\mathcal{L}|}{6} \right\rfloor = \left\lfloor \frac{3(2k+1)+1}{6} \right\rfloor = \lfloor k + \frac{4}{6} \rfloor = k$, this circulation graph attains the bound for every k. ◀

▶ **Lemma 7.** *If* $\sigma(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) = k$, *the circulation graph contains at least* $2k + 1$ *disjoint odd cycles of saving circulations.*

Proof. Every vertex x of the fractional matching polytope is half-integral, i.e. $x_e \in \{0, \frac{1}{2}, 1\}$ [2]. Moreover, the edges with $x_e = 1$ form a matching and the set of edges with $x_e = \frac{1}{2}$ form a set of disjoint odd cycles. If the optimal solution to the fractional matching problem contains ω such odd cycles, the difference between the size of the fractional matching and the size of the matching equals $\frac{\omega}{2}$, as a fractional matching in a single odd cycle can improve

the objective only by $\frac{1}{2}$ compared to an integer matching in the same cycle. As such, if $\sigma(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) = k$, we certainly have that $\bar{\sigma}^{\text{LP}}(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) \geq k$, implying that the circulation graph contains at least $2k$ disjoint odd cycles of saving circulations.

We prove that the number of odd cycles of saving circulations should be one more than $2k$ by contradiction. As we have already established that the number of odd cycles is at least $2k$, we assume that $\sigma(\mathcal{L}) - \bar{\sigma}(\mathcal{L}) = k$ while there are exactly $2k$ odd cycles. Note that an odd cycle cannot contribute strictly more than $\frac{1}{2}$ to the difference between $\sigma(\mathcal{L})$ and $\bar{\sigma}(\mathcal{L})$, as this violates the fact that fractional matching is the relaxation of the savings problem. Hence, it must hold that every odd cycle contributes exactly $\frac{1}{2}$ to the difference between savings.

However, we will now show that for the VCSP, every odd cycle can increase the difference between $\sigma(\mathcal{L})$ and $\bar{\sigma}(\mathcal{L})$ with strictly less than $\frac{1}{2}$. This is the case since it is not possible to select all circulations in an odd cycle of saving circulations in the circulation graph with value $\frac{1}{2}$. To see this, note that if circulation $c = \{l, m\}$ contributes $\frac{1}{2}$ to the objective of the VCSP, this implies that k_c is even (e.g. $k_c = 4$ and $\theta_c = 2$). Furthermore, since $k_c = k_l + k_m - 1$ (the circulation is saving), it must hold that k_l and k_m have a different parity (one of them is odd, the other even). As such, if we do have an odd cycle in which every circulation is selected with value 1/2, there must exists a 2-coloring of the vertices of the cycle. Since this is clearly not possible for an odd cycle, we reach a contradiction. ◀

▶ **Theorem 8.** *For any line plan it holds that* $\frac{\bar{\nu}(\mathcal{L}) - \nu(\mathcal{L})}{\nu(\mathcal{L})} \leq \frac{1}{15}$. *Furthermore, there exists an instance where this bound is attained. This implies that* $\bar{\nu}(\mathcal{L})$ *is a* $\frac{16}{15}$*-approximation algorithm for the VCSP with* $\kappa = 2$ *and all frequencies* 1.

Proof. First note that

$$\max \frac{\bar{\nu}(\mathcal{L}) - \nu(\mathcal{L})}{\nu(\mathcal{L})} = \max \frac{\sigma(\mathcal{L}) - \bar{\sigma}(\mathcal{L})}{\sum_{l \in \mathcal{L}} k_l - \sigma(\mathcal{L})}. \tag{16}$$

It follows from Lemma 7 that for a given value of $\bar{\nu}(\mathcal{L}) - \nu(\mathcal{L}) = k$, the worst case ratio must be attained by using $2k+1$ cycles of 3 vertices (more or larger cycles only lead to larger values in the denominator). Next to that, for a fixed numerator it is easily seen that the denominator of the ratio is minimized by letting a single node connect all the cycles. This implies that for a given value of $\bar{\nu}(\mathcal{L}) - \nu(\mathcal{L}) = k$, the instance in Figure 3 gives the worst case ration. Maximizing over k gives

$$\max_{k \in \mathbb{N}} \frac{k}{(2k+1)(3k+3) + k - (3k+1)} = \frac{1}{15}, \tag{17}$$

with the maximum being attained at $k = 1$. ◀

4.4 An exact algorithm for bounded treewidth

In this section we consider how to solve the VCSP exactly with $\kappa = 2$ where the circulation graph has a low *treewidth*. Treewidth is a graph property that was introduced by Robertson and Seymour [13] that, informally, indicates how "similar to a tree" the graph is. Many problems that are NP-hard on general graphs, such as independent and dominating set, are solvable in polynomial time if the treewidth of the input graph is bounded by a constant.

Formally, the treewidth of a graph G is the smallest *width* for which there a exists *tree-decomposition* of G with that width. A tree-decomposition of an undirected graph $G = (V, E)$ is a tree $\mathcal{T}$, where each node $n \in \mathcal{T}$ is associated with a bag $X_n \subseteq V$ and these two properties hold: (1) the endpoints of each edge should occur simultaneously in at least

one bag, i.e. for each edge $\{v, w\} \in E$ there is a node $n \in \mathcal{T}$ such that both $v \in X_n$ and $w \in X_n$, and (2) for each vertex $v \in V$, all nodes n for which the associated bag contains v, i.e. $v \in X_n$, are a connected subtree of $\mathcal{T}$. The width of a tree decomposition $\mathcal{T}$ is equal to $\max_{n \in \mathcal{T}} |X_n| - 1$. Although finding a tree-decomposition of the treewidth of a graph is NP-hard, there is a linear time algorithm [3] for any fixed width. Furthermore, there are algorithms that are able to efficiently find good tree decompositions in practice, e.g. [16].

One versatile approach in the design of algorithms that exploit bounded treewidth is to perform *dynamic programming* on the tree-decomposition. Central to this idea is the interpretation of every bag X_n in the tree-decomposition as a *graph separator*, which means that if we remove the nodes in the bag from the graph, the graph splits up in different parts. By moving up the the tree of the tree-decomposition, the algorithm looks at the current bag of vertices of the graph, which separates the part of the graph that is already processed by the algorithm from the part still needs to be processed, with the invariant that all connections between the processed and unprocessed parts of the graph must go through the current bag. In each state of the algorithm a state table is constructed for (combinations of) vertices in the bag associated with the current node in the tree, under the assumption that optimal decisions were made for the processed part of the graph.

A helpful way to design a dynamic programming algorithm based on the tree-decomposition is to assume it is a *nice* tree-decomposition [4]. Such a tree-decomposition has a root and as a consequence the order in which the dynamic programming algorithm visits the nodes of the tree is fixed: we start at the leaves and moves up to the root. In our description of the algorithm, we say that the algorithm moves from *parent* nodes to *child* nodes. A nice tree-decomposition distinguishes four types of nodes: *create* nodes which corresponds to leaves in the tree that only have a single vertex in their bag, *introduce* nodes which introduce a single new vertex into the bag of their parent, *forget* nodes which remove a single vertex from the bag of their parents and *join* nodes which have the same bag as their two parents. Thus in a nice tree-decomposition join nodes have two parents, leaf nodes have no parents and the other nodes have a single parent. There exists a linear time algorithm that converts any tree-decomposition into a nice tree-decomposition with $O(|V|)$ nodes and the same width [4].

Our algorithm adopts this approach. For each bag a state table is constructed for all partial covers of the lines in the current bag, based on the possible combinations of values of the left hand sides of Constraints 5. In every step we are only allowed to increase the θ_c values and thus the coverage of each line. Since each circulation $c \in E$ can only be selected an integer number of times, there is a finite number of fractions $\sum_{c \in \delta(l)} \frac{\theta_c}{k_c}$ that lie in the range $[0, 1]$ for each line l. The total number of combinations of values of the left hand sides of Constraints 5 for a particular set of lines is at most the product of the possible number of values for the individual constraints. This gives us an upper bound on the number of states we need to maintain in a state table when we enumerate the optimal partial covers for that bag. An upper bound on the number of possible fractional values for the left hand side of the constraint of a line l is denoted by ρ_l. One (crude) upper bound for this can be computed as $1 + \prod_{c \in \delta(l)} k_c$. Note that if the circulations are short enough compared to T, which they often are in practice, this number will typically be small.

A single state in the state table for a bag X_n assigns a fraction q_l to each line $l \in X_n$ where we have $q_l \in \{0, \frac{1}{\rho_l}, \ldots, \frac{\rho_l - 1}{\rho_l}, 1\}$. The state table for a node $n \in \mathcal{T}$ maps each state to the minimum number of vehicles required to reach this state. If the algorithm generates the same state multiple times, it is sufficient to store only the state for which the minimum number of vehicles was required to reach the state. If we introduce ρ as the maximum ρ_l for all lines $l \in \mathcal{L}$, the size of the table is for a single bag is $O(\rho^{w+1})$.

To conclude the description of the algorithm, we describe how the state table for each of the four types of nodes in the tree-decomposition can be computed based on the state tables of the parent(s). In a *start* node, only a single line l is introduced and thus only a fixed circulation is considered, modeled by a self loop. This loop can be used at most ρ_l times and may not be used at all. These state tables can be generated in $O(\rho)$ time. In an *introduce* node, a new line l is introduced to the state table of the parent. Due to this introduction, we have to expand all states in the parent table with all possible multiplicities of the circulations in $\delta(l)$, including multiplicities of zero. As the state table of the parent contains $O(\rho^w)$ states and there are $O(w)$ circulations that connect the new line l to lines in the current bag, each of which can be used at most ρ times in the expansion, we can construct the state table for an introduce node in $O(w \cdot \rho^{w+1})$ time. In a *forget* node, a line l is removed from the state table of parent. This means that from this step onward, that particular line is in the set of lines that have been processed by the algorithm and thus we must make sure that it is fully covered. This can be achieved by removing all states from the parent table where the q_l of this line is not equal to 1, as the removal of line l implies that these states are infeasible. This can be done in $O(\rho^{w+1})$ time as we only have to filter the state table of the parent. Finally, in a *join* node we have two parent nodes with the same bag, but potentially different states. We construct the new state table by either taking the state and its associated number of vehicles from one of the two parents' state table or by taking a state from one table and combining it with a state from the second table, by adding up the q_l's of both states and adding up the number of vehicles of the two states. These combinations are only worth considering if none of the resulting q_l's exceeds 1. As both parent tables can be of size ρ^{w+1}, there are ρ^{2w+2}, combinations that can be explored in this step. This means the table for a join node can be computed in $O(\rho^{2w+2})$ time.

When the algorithm is done, we can find the optimal solution to the VCSP in the root node at the state where all q_l's are equal to one. Recall that the size of the tree-decomposition is $O(|\mathcal{L}|)$ and each node in this decomposition can be processed in $O(\rho^{2w+2})$ time.

▶ **Theorem 9.** *The VCSP with $\kappa = 2$ and auxiliary constraints on the allowed circulations can be solved in $O(n\rho^{2w+2})$ time where $n = |\mathcal{L}|$, w is the tree-width of the circulation graph and $\rho = \max_{l \in \mathcal{L}} \prod_{x \in \{k_c | c \in \delta(l)\}} x$.*

5 Conclusions and further research

We have shown that the Vehicle Circulation Scheduling Problem is NP-hard for any finite restriction on the number of lines that can be included in a circulation (κ) greater than two. For the $\kappa = 2$ case we need to make the (realistic) additional assumption that we have auxiliary restrictions on which lines can be combined in order to prove NP-hardness. For the $\kappa = 2$ case we show that if we can cover each line by at most one unique circulation, a matching algorithm yields the optimal solution. This solution provides a $\frac{16}{15}$-approximation in case multiple circulations can be used. We also provide an exact algorithm that can exploit low treewidth of the circulation graph, and a low number of vehicles required per circulation. For future research, it makes sense to combine these algorithms with the line planning process to see if it can help to make line plans that allow better vehicle schedules. Furthermore, it is interesting to consider whether algorithms exist that are useful for cases where κ is small, but greater than two. Finally, it is still an open question whether the $\kappa = 2$ case without auxiliary restrictions is NP-hard.

References

1 Erwin JW Abbink, Luis Albino, Twan Dollevoet, Dennis Huisman, Jorge Roussado, and Ricardo L Saldanha. Solving large scale crew scheduling problems in practice. *Public Transport*, 3(2):149–164, 2011.

2 Michel Louis Balinski. Integer programming: methods, uses, computations. *Management science*, 12(3):253–313, 1965.

3 Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

4 Hans L Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

5 Ralf Borndörfer, Martin Grötschel, and Marc E Pfetsch. A column-generation approach to line planning in public transport. *Transportation Science*, 41(1):123–132, 2007.

6 Gabrio Caimi, Leo Kroon, and Christian Liebchen. Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning & Management*, 6(4):285–312, 2017.

7 Ilkyoo Choi, Jaehoon Kim, and O Suil. The difference and ratio of the fractional matching number and the matching number of graphs. *Discrete Mathematics*, 339(4):1382–1386, 2016.

8 Pieter-Jan Fioole, Leo Kroon, Gábor Maróti, and Alexander Schrijver. A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174(2):1281–1297, 2006.

9 Michael R Garey and David S Johnson. Computers and intractability. a guide to the theory of NP-completeness. a series of books in the mathematical sciences, 1979.

10 Omar J Ibarra-Rojas, Fernando López-Irarragorri, and Yasmin A Rios-Solis. Multiperiod bus timetabling. *Transportation Science*, 50(3):805–822, 2015.

11 Natalia Kliewer, Bastian Amberg, and Boris Amberg. Multiple depot vehicle and crew scheduling with time windows for scheduled trips. *Public Transport*, 3(3):213–244, 2012.

12 Julius Pätzold, Alexander Schiewe, Philine Schiewe, and Anita Schöbel. Look-ahead approaches for integrated planning in public transportation. In *OASIcs-OpenAccess Series in Informatics*, volume 59. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

13 Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

14 Anita Schöbel. Line planning in public transportation: models and methods. *OR spectrum*, 34(3):491–510, 2012.

15 Anita Schöbel. An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research Part C: Emerging Technologies*, 74:348–365, 2017.

16 Hisao Tamaki. Positive-Instance Driven Dynamic Programming for Treewidth. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7880`, `doi:10.4230/LIPIcs.ESA.2017.68`.