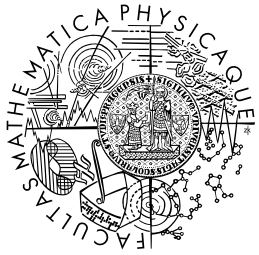


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Petra Pelikánová

Decompositions of directed and undirected graphs

Department of Applied Mathematics

Supervisor of the master thesis: prof. RNDr. Martin Loebl, CSc.
Study programme: Computer Science
Study branch: Discrete Models and Algorithms

Prague 2020

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

First of all, I would like to thank my supervisor, Martin Loebl, for gently guiding my attempts to explore the world of applied mathematics. Thank you Loebl for the creation of a research team with Jiří Fink which I have the privilege to be a part of.

I am very grateful to all members of the Road Maintenance and Operations Division in the Plzeň Region. Namely, the chief executive officer Miroslav Doležal who initiated successful cooperation between the Plzeň region and the Faculty of Mathematics and Physics. I cannot forget the support of the regional council president Josef Bernard and the head of the Department of Transportation Pavel Čížek.

I also extend my gratitude to my family for encouraging me during the years of my study, to my friend Aneta Pokorná, and finally to the coronavirus for forcing me to stay at home and finish my thesis.

Title: Decompositions of directed and undirected graphs

Author: Bc. Petra Pelikánová

Department: Department of Applied Mathematics

Supervisor: prof. RNDr. Martin Loebl, CSc.,
Department of Applied Mathematics

Abstract: Eulerian graphs have a closed walk traversing each edge exactly once. Finding such a walk is a basic arc routing problem based on a road network. Most of the problems with applications in operational research are NP-hard. We describe a formal model of a road network and vehicle routes and formulate several arc routing problems motivated by winter road maintenance in the Czech Republic.

The main part is focused on single vehicle routing problems on trees. We propose a new unfairness minimization problem for finding a vehicle route with properties that lead to a minimal number of resident complaints against unfair maintenance. Residents feel like they are skipped when the vehicle route has multiple trips and passes nearby without providing maintenance to their street. By reduction of the necklace splitting problem to the unfairness minimization problem we prove it is PPA-complete.

Further, we define a restricted arc routing problem on trees which formalize conditions given by Czech legislation. We proved the existence of a polynomial algorithm for deciding whether a single vehicle route exists when there is a single priority for roads. If multiple priorities are used, we express conditions and conjectures when the problem has polynomial complexity. Finally, a utilization of the model is illustrated by an application in winter road maintenance in the Plzeň region.

Keywords: graph, decomposition, cycle, closed Eulerian walk, algorithm

Contents

Introduction

The topic of the thesis is decompositions of graphs and digraphs into closed walks motivated by the winter road maintenance problem.

We talk mostly about *simple (undirected) graphs* $G = (V, E)$ with the set of vertices V and the set of edges E where elements $\{u, v\} \in E$ are unordered pairs of vertices and u, v are distinct. When we study *directed graphs (digraphs)* $G = (V, E)$ we talk about the set of arcs E which are ordered pairs of vertices from V.

A walk is defined as a sequence $v_0, e_1, v_1, e_2, ..., e_\ell, v_\ell$ of alternating vertices v_i and edges e_i such that for $1 \leq i \leq \ell$ the edge e_i has endpoints v_{i-1} and v_i (for directed graphs $e_i = (v_{i-1}, v_i)$). If endpoints v_0 and v_ℓ are the same, it is a *closed walk*. Moreover, it is a *cycle* if the closed walk has no repeated vertices with the exception of the end-vertices.

In the beginning I was interested in the existence of long and short cycles. This topic is discussed in Chapter 1. However, soon I turned my attention to the arc routing problems introduced in Chapter 2, and specifically to the problem of designing winter road maintenance plans.

The main results of the thesis are contained in Chapters 3 and 4. Chapter 3 is focused on the problem of how to find a single vehicle route. Chapter 4 describes an illustrative application of a theoretical model created in Chapter 3.

Chapter 3 is based on and extends the work of Fink and Loebl [1]. The basic concepts defined there are repeated or slightly modified here. I participated in introducing these basic concepts which is acknowledged in the paper.

The final chapter contains a description of an application for the winter road maintenance in the Plzeň region. This thesis initiated our collective research with Fink and Loebl. Extended results are summarized in our paper [2] about an arc routing algorithm applied to the winter road maintenance in the Czech Republic.

1. Existence of Long and Short Cycles

Graph theory examines properties of many graph classes. Eulerian graphs form one of them which shows up naturally in many applications. The theorem attributed to Euler, that a graph has an Euler closed walk if and only if it is connected and each *degree* of a vertex (number of edges incident with a vertex) is even, gave rise to graph theory.

The parameters of a graph such as the *order* (number of vertices), the *size* (number of edges) or the *girth* (length of the shortest cycle) are connected to the existence of cycles, e.g., any undirected graph G with n vertices and m edges has a subgraph with minimum degree at least m/n, and thus G also contains a cycle of length at least $m/n + 1$. [3]

Many results from undirected graphs cant be generalize for directed graphs, so special classes of graphs are studied, including Eulerian graphs.

An *Eulerian digraph* is a connected directed graph, in which for each vertex the in-degree (number of incident arcs oriented into the vertex) equals the out-degree (number of incident arcs oriented outside of the vertex). An example of an Eulerian digraph is the symmetric orientation of a graph.

A symmetric orientation of an undirected graph $G = (V, E)$ is a directed graph $G^S = (V, E^S)$ where a set of vertices is identical to the set of vertices of the original graph G and the set of arcs E^S contains for all edges $\{u, v\} \in E$ symmetric arcs $(u, v) \in E^S$ and $(v, u) \in E^S$.

A *feedback arc set* of a digraph is a set of arcs whose removal makes the digraph acyclic. We denote by $\beta(G)$ the minimum size of a feedback arc set.

Several results about the existence of cycles in directed graphs are mentioned in this section.

1.1 Eulerian graphs and long cycles

An attempt to generalize results known for undirected graphs was made by Huang, Ma, Shapira, Sudakov and Yuster [3]. They focused on several parameters of Eulerian digraps and especially studied the feedback arc sets.

Below we mention their results on relations of the order and size of an Eulerian digraph with the existence of long and short cycles.

Theorem 1 (Existence of a short cycle) *Every Eulerian digraph G with n vertices and m arcs has the length of the shortest cycle at most $6n^2/m$.*

Theorem 2 (Existence of a long cycle) *Every Eulerian digraph with n vertices and m arcs has a cycle of length at least $1 + \left\lfloor \sqrt{m/n} \right\rfloor$.*

1.2 Conjectures about short cycles

The Caccetta-Häggkvist Conjecture is one of the famous and extensively studied conjectures in the graph theory. The conjecture talks about the upper bound of

the cycle length in digraphs. Many partial results have been made [4].

Conjecture 1 (Caccetta-Häggkvist Conjecture) *Every simple n-vertex digraph with a minimum out-degree at least k has a cycle with length at most $\left\lceil \frac{n}{k} \right\rceil$.*

It has been proven only for $k \leq 5$ by Hoàng and Reed [5]. Other conjectures increase its popularity because they partially include the Caccetta-Häggkvist. There are alternative formulations related to different fields of graph theory, e.g. graph colorings (edges have assigned colors for our purpose).

A generalization of the Caccetta-Häggkvist Conjecture is the Aharoni Conjecture [7] where the orientation is transformed into colors. A *rainbow cycle* in an edge-colored graph is a cycle with all edges having distinct colors.

Conjecture 2 (Aharoni Conjecture) *Let G be an undirected graph on n vertices with the edges colored by n colors such that each color is used on at least k edges. Then there is a rainbow cycle of length at most $\leq \left\lceil \frac{n}{k} \right\rceil$ in G.*

We obtain the Caccetta-Häggkivst Conjecture from the Aharony Conjecture by coloring all outgoing edges for each vertex with one unique color. Then a rainbow cycle from the edge-colored graph is an oriented cycle in the corresponding digraph.

The Aharoni Conjecture is proven to hold for $k = 2$ in a paper about short rainbow cycles [8].

We proved, in a joint work with Sophie Spirkl and Aneta Šťastná [9], the following partial result.

Theorem 3 *Let G be a graph on n vertices with edge coloring using n colors such that each color is used on at least $(128k^3 + 74k^2 + k)^2$ edges. Then there exists a rainbow cycle of length at most $\left\lceil \frac{n}{k} \right\rceil$ in G.*

So if we have asymptotically $\mathcal{O}(k^6)$ edges of each color, the Aharoni Conjecture implies the existence of a short cycle of length at most $\left\lceil \frac{n}{k} \right\rceil$.

We are at the end of the famous Cimrman's Step-Aside and we can continue with other parts of the research.

2. Arc Routing Introduction

The following chapters are about arc routing problems and an application in winter road maintenance. In general, routing problems are based on a road network and they deal with problems related to it. Common examples are mail delivery and garbage collection. New vehicle routing problems occur with the evolution of technology. The routing of drones is one of the current interesting problems dealing with navigating an autonomous entity in areas with flight restrictions.

2.1 Arc routing

In *arc routing* problems there are demands on the edges of a graph. As an example, we include the classical *Chinese postman problem*. Suppose there is a mailman who needs to deliver mail to a certain neighborhood. The mailman is unwilling to walk far, so he wants to find the shortest route through the neighborhood that meets the following criteria: it is a closed walk (it ends at the same point it starts) and he needs to go through every street at least once.

If the graph of the road network to be traveled by the mailman has an Eulerian *circuit* (no repeated edges), this circuit is the ideal solution.

Otherwise, the problem is to find a minimum weight T-join, where T is the set of vertices of an odd degree. These vertices determine roads which have to be traversed twice. The solution can be found by a reduction to the weighted perfect matching problem.

Alan Goldman of the U.S. National Bureau of Standards first coined the name "Chinese Postman Problem" for this problem, as it was originally studied by the Chinese mathematician Mei-Ku Kuan in 1960 [12].

2.2 Vehicle routing

An associated area, which will not be discussed here, is formed by the *vehicle routing* problems. On the contrary to arc routing, the vehicle routing problems have demands on vertices. A basic example of a vehicle routing problem is the famous *travelling salesman problem*.

Imagine a fleet of vehicles and a set of vertices which has to be visited for meeting the demands. We always want to minimize the cost of the routes, which may depend on the length of the routes, delivery times, the number of used vehicles or a penalty given by the quality of service, and so on.

2.3 Motivation

The arc routing problems on which we focus here are usually NP-hard[1]. This is a reason why the existing research in the arc routing area is extensively focused on heuristics and approximations.

[1]The Chinese postman problem is an exception since there exists a polynomial algorithm.

We will now specify the types of arc routing problems discussed in this thesis. My involvement started with my supervisor asking the question:

"Can we improve routing for winter road maintenance in the Czech Republic and specifically in the Plzeň region?"

In the Plzeň region, the involved experts were creating routes using their experience and good practice, but also considered public opinion and made changes accordingly. This leads to a design governed by local changes. Such a situation welcomes attempts for a new and more global approach based on modern algorithms and extensive computation.

The main task was: to create new routing for winter road maintenance vehicles while minimizing the total number of vehicles used. There are many additional conditions that need to be satisfied; in particular, conditions given by Czech legislation.

A fixed plan for an entire winter season had to be created. We do the *offline-planning* on a *tactical level* for middle length durations, i.e., months. The strategical level is in years, e.g., positioning of depots, while the operational level is for short time periods (online) planning like the work of a dispatcher.[10]

The winter road maintenance problem is a *capacitated* arc routing problem because the maintenance vehicles have a finite capacity of material which they can store, and roads have demands given by their length. Czech legislation limits the length of a working shift for drivers to eight hours including several security breaks and time for non-driving manipulations.

Another important rule imposed is that each road has to be maintained by the same vehicle in both directions. We assume that there are no one-way roads which was the case for winter road maintenance in the Plzeň region.

This is the reason why we base our models on *undirected* graphs in the next sections. In order to solve the problem of routing for a single vehicle we need a *directed* graph. For each undirected graph, we consider its *symmetric orientation* where each edge is replaced by two arcs with opposite orientations. This representation guarantees that each edge is maintained in both directions.

We do not need mixed variations of the routing problem which use graphs having both a set of undirected edges and a set of directed arcs. The mixed graphs are useful for cities with many of one-way roads.

Important operating conditions are given by defined *levels of service policies*. There are three levels in the Czech Republic. Arterial roads in a region used for long-distance travelling have the first level of priority and the highest level of service. The second priority has a medium level of service. The third priority of local roads has the lowest level of service. (Note that this hierarchy is not completely coincident with the well-known classification of roads marked as class I., II., and III.). Each priority has a time limit, e.g., the first priority roads have to be maintained within 3 hours.

Furthermore, each road has one of three *methods of maintenance* assigned to it. Most of the roads are maintained using chemicals (sodium chloride solution) because this strategy is inexpensive and does not require supplementary operations. This deicing method also prevents ice formation and it is preferred for roads with a large traffic volume.

The usage of the *chemical method* is not allowed in nature parks or in protected areas around water sources. In these cases abrasive materials are spread on the roads (*inert method*).

The mechanical method of *snowplowing* is used for minor parts of the network especially in the mountains where vehicles have to use snow chains.

The maintenance vehicles employed in the Plzeň region are specialized for one type of maintenance, either chemical or inert. Each vehicle is able to snowplow.

The plan for the next chapters is as follows. The main part of the research in Section 3 is based on the work of Loebl and Fink [1]. I extend it in several ways. I focus on the arc routing problem of one vehicle with multiple trips, which is not defined in the article.

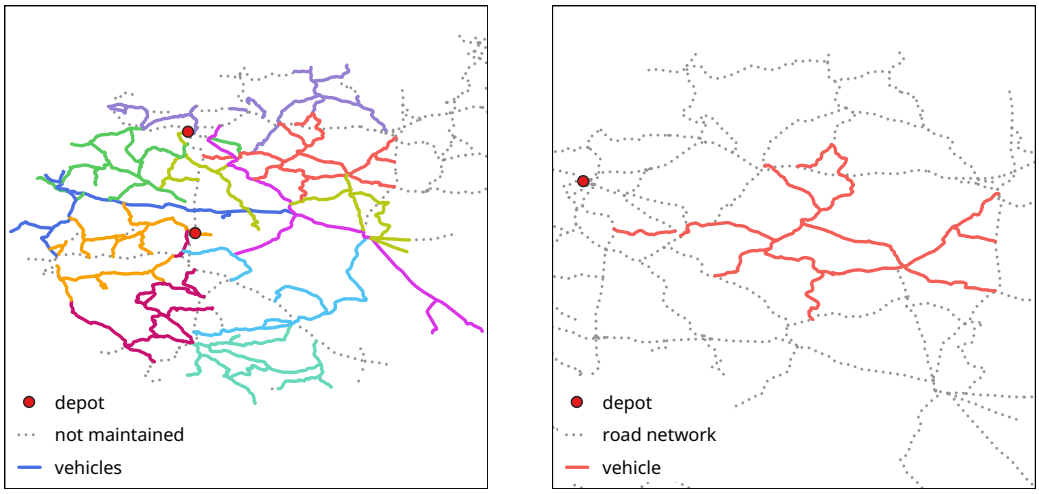
In Section 4, I describe how we (with Martin Loebl and Jiří Fink) solved the winter road maintenance problem for the Plzeň and Liberec regions. This chapter follows our joint paper [2]. My contribution to the paper is equally proportional to the other co-authors.

We take a road network and create a graph where each edge has attributes given by its priority and the type of maintenance for the corresponding road segment.

Some vertices serve as depots. Each such vertex has defined types of material which it can store.

The goal is to assign for each edge a vehicle which will maintain the corresponding road and to minimize the number of vehicles used as well as the length of the roads traversed without maintenance *(deadhead)*.

Figure 2.1a is the west half of a road network in Liberec region covered by ten maintenance vehicles. The next Figure 2.1b is a map of roads maintained by one vehicle. Deadhead is the sum of the shortest path to the depot and the shortest possible passes between disconnected parts.



(a) Coverage by maintenance vehicles (b) Roads maintained by one vehicle

Figure 2.1: Example of a solution of an arc routing problem of winter road maintenance, the west part of road network in Liberec region

This problem is difficult in practice for such a big graph, as it encompasses an entire region of the Czech Republic.

3. Arc Routing for Trees

We can define a *tree* $T = (V, E)$ as a graph in which any two vertices are connected by exactly one path. If we single out one vertex r of a tree which we call a *root* we call it a *rooted tree*, rooted in vertex r. For any other vertex $v \neq r$ we denote the unique path from root r to the vertex v by $P(v)$. Moreover, we denote by $p(v)$ the *parent* of vertex v which is a *neighbor* (i.e., they are connected by an edge $\{v, p(v)\} \in E$) of v on the unique path $P(v)$. When $p(v)$ is the parent of v we call v a *child*, or a *son*, of $p(v)$.

A graph G' with the set of vertices $V(G')$ and the set of edges $E(G')$ is a *subgraph* of a graph $G = (V(G), E(G))$ if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$.

Let T be a rooted tree in r and $v \neq r$ be another vertex. When we delete the edge $\{v, p(v)\}$ from T, we obtain a graph which we call a *forest* (disjoint union of trees). We say that the component containing the vertex v is the *branch* $B(v)$ and we consider $B(v)$ a rooted tree with the root v.

Trees are graphs that are useful for representing a vehicle route in winter road maintenance. Even if the set of edges maintained by a single vehicle is not a tree, we can represent it as a tree obtained from the Depth first search (DFS) algorithm. A symmetric orientation of a tree is always an Eulerian graph and thus a natural vehicle route exists if the only goal is to visit all arcs exactly once. But there are more conditions to be satisfied and thus a more complex model is introduced below.

Arc routing is quite interesting and not easy even for one vehicle and a tree. We show links between this problem and Number Partitioning from Computer Science as well as Necklace Splitting and Sums Packing from Discrete Mathematics.

The definitions of Section 3.1 follow the paper by Fink and Loebl [1]. I am not a coauthor of the paper, but I participated in understanding how to define basic concepts which is acknowledged in the paper. We collaborated on the model formulation which is also used in our collective work [2].

3.1 Definitions

For facilitation, in the rest of the thesis we will use the terms defined below. It helps us to talk formally about the model of winter road maintenance. We will be able to formally describe the problems and their solutions.

In practice we have a given road network and we create a corresponding abstract graph structure. Then we formulate an arc routing problem as a problem of searching for a cover by subgraphs which correspond to parts of the network maintained by single vehicles.

Definition 1 (Graph of a road network) *Let $G = (V, E)$ be a graph representing a* road network*. Vertices represent crossroads (and dead ends) and edges represent roads among them. Let $z \geq 1$ denote the number of priority classes of roads and let M denote the set of types of road maintenance. We associate several functions with G:*

- $\alpha\colon E \to \mathbb{R}^+$ *assigns a non-negative length to edges,*
- $p\colon E \to \{1, \ldots, z\}$ *a priority level,*
- $m\colon E \to M$ *a type of maintenance.*

Let $D \subset V$ be a set of depots. For $d \in D$ we denote by $m(d) \in 2^M$ stored materials at the depot d.

This graph structure is used as the input for many of the following problems. A potential candidate for a solution can be described as a set of subgraphs. The subgraphs maintained by one vehicle are called pre-maintenance plans.

Definition 2 (Single vehicle pre-maintenance plan) *A* Pre-maintenance plan *for a vehicle is a tuple (G, P, d, α, z, p) where:*

1. *$G = (V, E)$ is a graph of a road network,*
2. *$P \subseteq E$ is the set of maintained edges,*
3. *$d \in D$ is called the depot, $D \subset V$,*
4. *$\alpha\colon E \to \mathbb{Z}^+$ assigns a non-negative integer length to edges,*
5. *$p\colon E \to \{1, \ldots, z\}$ defines the priorities of edges.*

The pre-maintenance plan has a crucial role in the routing of one vehicle. For our algorithmic purposes, we usually use a special case where the graph G is a tree rooted in the depot d.

For the graphs from the previous definition, we describe a walk which creates a vehicle route if all necessary conditions hold.

Definition 3 (Vehicle route) *For a given pre-maintenance plan (G, P, d, α, z, p), length limit L, capacity c, limits associated with priorities given by function $t : \{1, \ldots, z\} \to \mathbb{R}$ and function $f : E \to \mathbb{Z}^+$ we define a L, c, t, f-*vehicle route *as a closed walk w in graph G of a road network if the conditions below hold.*

A vehicle route is sequence of arcs $w = (e_1, \ldots, e_\ell)$ where e_i of w is an element of the symmetric orientation $G^S = (V, E^S)$ of graph G, i.e., E^S contains arcs (u, v) and (v, u) for every edge $\{u, v\} \in E$.

A trip *of a vehicle is a segment T_j of route w which starts and ends at depot d and the interior of T_j (ending vertices are excluded) does not contain the depot. A vehicle route is thus a sequence of trips $w = (T_j)_1^k$.*

The following conditions have to be satisfied in a vehicle route:

1. *(all edges are maintained) w starts and ends at d and each arc of symmetric orientation P^S is traversed at least once by w and at most $f(e)$ times,*
2. *(total length) $\sum_{q \le \ell} \alpha(e_q) \le L$,*
3. *(priority) for all pairs $i < j$ such that $e_i = e_j \in P^S$ and $e_i \neq e_k$ for $i < k < j$, $\sum_{q \ge i}^{j-1} \alpha(e_q) \le t(p(e_i))L$, taken cyclically,*
4. *(capacity) for all trips $T_j\colon \sum_{e \in T_j \cap P^S} \alpha(e) \le cL$.*

The necessity of conditions defining vehicle routs are self-explanatory except possibly for the last one. The last condition describes the constraint given by the limited storage of a vehicle. The condition expresses that after maintenance length at most cL, the vehicle must return to its depot for the reloading of maintenance material.

A set W of vehicle routes is a solution to the winter road maintenance arc routing problem if the sets P of the elements of W form a partition of the set of all the maintained edges in the given road network.

We first observe that it is a hard problem to find a vehicle route, even if the network is a tree.

Observation: *It is a NP-complete problem to decide if a pre-maintenance plan admits a L, c, t, f-vehicle route.*

Proof. The problem is hard even if G is a star rooted at its vertex r of degree one. All edges are maintained $P = E$, all edges have the same priority, i.e., $z = 1$, the lengths of edges are integers $\alpha : E \to \mathbb{N}$, the total limit L for length of vehicle route is $2\alpha(e_0) + \sum_{e \in E} 2\alpha(e)$ where e_0 is the edge associated with r, the associated functions $p(e) = t(p(e)) = 1$ are constant, $f(e_0) = 2$ and is otherwise equal to 1 and $c = 1/2$ does not depend on the tree. Such an input tree admits a vehicle route if and only if the edges not incident with r can be divided into two parts with equal sums of edge lengths. This problem is called *Partitioning* and it is NP-complete. □

The number partitioning problem is a fundamental problem for many proofs of NP-completeness. The computational complexity is very interesting, which is why the problem is called The Easiest Hard Problem [13]. The exact formulation is in the next definition.

Definition 4 (Number Partitioning Problem) *Let $a_1, \ldots, a_n$ be a list of positive integers.* The number partitioning problem *is to find a subset $A \subseteq \{1, \ldots, n\}$ with minimal discrepancy*

$$d(A) = \left| \sum_{i \in A} a_i - \sum_{i \notin A} a_i \right|.$$

A partition with $d = 0$ is called a perfect partition.

We can assume a less general definition of a pre-meaintaing plan. The concepts can then be explained in a more clear way.

Definition 5 (maintenance plan) *Any pre-maintenance plan (G, P, d, α, z, p) is in addition a* maintenance plan*, denoted by (G, d, z, p), if it satisfies that:*

1. *$\forall e \in E(G) \colon \alpha(e) = 1$ all edges have unit length and*
2. *$P = E$ all edges are maintained.*

Our motivation for formulating the first condition stems from the fact that in practice each length (in meters) is bounded by a constant and we can always

subdivide edges. The second condition is justified by our computational experiments where there is typically only negligible length of deadhead in proportion to the length of a vehicle route.

If we have a maintenance plan, we still don't have a vehicle route and we know it can be hard to decide if such feasible route exists with respect to all conditions in the Definition 3.

Definition 6 (Admissible plan) *A maintenance plan* (G, d, z, p) *is* admissible *if and only if it admits a* L, c, t, f*-vehicle route.*

The last definition in this section formulates a *monotonic property* which enables the service hierarchy to be reflected in the solution of arc routing. The constraint expresses an intuitive necessity of having edges of higher importance nearer to the depot. This means the priority function is non-decreasing on each path from the depot to a border of the plan. In order that such a property can be correctly formulated, we need to consider trees only as network graphs.

Definition 7 (Monotonic property) *A maintenance plan* (T, d, z, p) *has a* monotonic property *if the graph* T *is a tree rooted in depot* d *and the following condition holds: for each path* $W = (e_1, \ldots, e_\ell)$ *from the depot* d *to a leaf, the priority is non-decreasing*

$$\forall e_i, e_j \in W,\, i \leq j \colon p(e_i) \leq p(e_j).$$

This property of a maintenance plan is very useful in times of a calamity. If there is not enough resources for maintaining the whole network, the plans can be reduced by less frequent maintenance of edges near leaves of the tree.

3.2 Arc routing problems specification

The arc routing problem on trees is to find a vehicle route for a single car with a given maintenance plan. It is a subproblem of finding a feasible solution to the winter road maintenance problem.

3.2.1 General arc routing problem

Definition 8 (Arc routing problem for trees) *The* arc routing problem on a tree *is to find a vehicle route for a given maintenance plan* (T, d, z, p) *where* T *is a graph of a road network which is a tree.*

The decision version of the problem can be used as a subroutine of algorithms for searching for a solution of problems with multiple vehicles. It answers the question whether a given maintenance plan is admissible. The following theorem is proved by Fink and Loebl [1].

Theorem 4 (Fink, Loebl) *Let* z, Δ, F *be integer constants and let* (T, d, z, p) *be a maintenance plan where* T *is a tree rooted at* d *with maximum degree* Δ*. Let* $f : E \to N$ *satisfies for each* $e \in E$, $f(e) \leq F$*. Then there is a polynomial algorithm to decide if a* L, c, t, f*-vehicle route on* T *exists.*

The algorithm given in the proof of theorem 4 (Fink, Loebl) is based on dynamic programming. Even though the complexity is polynomial for practical implementation, it is too high. In view of this fact, we formulate a restricted arc routing problem for trees.

3.2.2 Restricted arc routing problem

The following additional conditions for the graph of a road network and the vehicle routes are based on our experience with winter road maintenance in the Czech Republic.

First of all, we require the *monotonic property* (Definition 7) for the maintenance plans. In a catastrophic scenario of heavy snowfall, the vehicles can simply restrict the maintenance operations to important parts of the network.

Czech law distinguishes three priority classes with time limits for their maintenance. According to these time limits, we define *priority constraints* by function $t(p)$.

Definition 9 (Restricted arc routing problem for trees) *Let T be a graph of a road network which is a tree, and let $(T, d, 3, p)$ be a maintenance plan with the monotonic property. The* restricted arc routing problem *is to find a L, c, t, f-vehicle route where $t(p) = 1/(4-p)$ and f is not specified.*

This is the interesting case for the winter road maintenance in the Czech Republic.

Definition 10 (Constant capacity property) *We say that a L, c, t, f-vehicle route for a given maintenance plan has a* constant capacity property *if the capacity c is a constant which does not depend on the given maintenance plan.*

In the context of the winter road maintenance, it is natural to think of vehicle capacity as being constant and identical for all the vehicles in a fleet.

But, in a theoretical study of the existence of a single vehicle route in a given maintenance plan, this is not useful.

Vehicle capacity is defined as cL for a given length L. Here both c and L may depend on the input tree T.

In practice it make sense to assume that L depends on T, in fact usually L is at least $2|E(T)|$ if each edge has length at least 1.

Not only the limit L can be a function of the input tree but also the derived vehicle capacity. Most often we can assume that the capacity c is a fixed fraction, e.g., $c = 1/c'$ where c' equals to 2.

We will consider both fixed and general capacity cL. The reason is that considering general capacity allows us to connect the existence of a vehicle route on a tree with the interesting mathematical concepts of necklace splitting, sum packing, Sidon sets, and the PPA complexity class.

Another parameter which impacts the complexity is the *strong capacity property* of a vehicle route.

Definition 11 (Strong capacity property) *Let $w = (T_j)_1^k$ be a L, c, t, f-vehicle route for a maintenance plan (T, d, z, p) where $T_1, \ldots, T_k$ are the trips and T is a tree with a given fixed order of children of each vertex. The vehicle route has the* strong capacity property *if k is a constant independent of the input and each T_j is a closed trail (each arc is traversed exactly once) where the sequence of arcs in each T_j is fixed by the given order of children of each vertex.*

In this section we study sufficient conditions, algorithms, and further aspects associated with the existence of a L, c, t, f-vehicle route on a given tree T not covered by Theorem 4.

3.2.3 Examples

These illustrative examples of maintenance plans are from the paper by Fink and Loebl [1].

Example 1: First, let T consist of a path P of $L/8$ vertices rooted at an end-vertex where each edge has priority 1, and $L/8$ leaves where each leaf is attached to a different vertex of P by an edge of priority 3. Let $t(p) = 1/(4-p)$. It is not difficult to see that this maintenance tree is admissible. There exists a vehicle route which consists of three trips, each of them contains the whole path P and one third of the edges of priority three, so $L/4 + L/12 = L/3$.

Example 2: Secondly let T be a path of $L/4$ edges rooted at an end-vertex, where the initial (from the root) part of length $L/8$ has priority 1 and the remaining part has priority 3. It is clear that this maintenance tree is not admissible.

We note that in both examples, the number of edges of T is $L/4$. Hence, the number of edges is not by itself a sufficient parameter for deciding admissibility. Some necessary and sufficient conditions for the existence of a vehicle route can be found in the proof of Theorem 8.

The next section examines a very important aspect of the winter road maintenance, which may be called *public complaints*. This is clearly a political issue and some public dissatisfaction with winter road service can be avoided by public relations actions. This concept is related to interesting mathematics and complexity fields.

3.3 Unfairness of vehicle route

We want to illustrate that the vehicle routes on trees studied in this section, i.e., with natural restrictions given by winter road maintenance in the Czech Republic, have interesting unexpected connections with several well studied problems. The particular important aspect studied here is *public complaints*. This is an important issue for everyone involved in this business globally.

Services of winter road maintenance are generally of higher quality than what the constraints of legislation require. Experts who design winter road maintenance plans pay close attention to the satisfaction of residents. The number of complaints from the public against winter road maintenance is a quantitative measure of the quality of service which is focused on, e.g., in the media or in election campaigns.

3.3.1 Public complaint

Residents make a complaint of insufficient service if they think that they are skipped in the service or an adjacent district is maintained more frequently. They only see a vehicle going around and feel a sense of unfairness.

The next definitions formalize the terms of neighborhood and unfairness. Figure 3.1 illustrates the notation of the definition of neighborhood.

Definition 12 *(Next-door neighbor, previous-door neighbor) Let $T = (V, E)$ be a tree rooted in r with given fixed linear orders of children of each vertex. Let E^S*

be the set of the arcs of the symmetric orientation T^S. We define a relationship for arcs $e_1, e_2 \in E^S$. We say $e_1 = (u_1, v_1)$ has a next-door neighbor $e_2 = (u_2, v_2)$ *and e_2 has a* previous-door neighbor e_1 *if they satisfy conditions below:*

1. *$u_1 = p(v_1)$ and $u_2 = p(v_2)$,*
2. *u_1 and u_2 are vertices in unique path $P(v_2)$ from the root r to v_2,*
3. *u_1 is closer to the root than u_2, i.e., $|P(u_1)| \leq |P(u_2)|$,*
4. *the interior vertices in the subpath of $P(v_2)$ from u_1 to u_2 all have degree 2 (the length of the subpath can be zero then $u_1 = u_2$),*
5. *v_1 is not in the path $P(v_2)$,*
6. *in the fixed order of children of u_1, the vertex v_1 is the child immediately preceding the vertex (denoted by v_1') of the subpath of $P(v_2)$ from u_1 to v_2.*

We also say that vertex v_1 has a next-door neighbor vertex v_2 and vertex v_2 has a previous-door neighbor v_1.

We can look at path $P(u_2)$ and then we can see that u_1, u_2 are the closest vertices on the path with crossroads (they have a degree greater than two) and v_1, v_2 are turn-offs. In case $u_1 = u_2$ the vertices v_1, v_2 are just consequential children of the vertex u_1.

We note that each arc has at most one previous-door neighbor and may have no next-door neighbors or more than one next-door neighbor. The next-door neighbors form a directed path and the arcs on this path are naturally directed from the root. In the Figure 3.1 the arc e_1 has next-door neighbors $(u_1, v_1'), (v_1', u_2), (u_2, v_2)$.

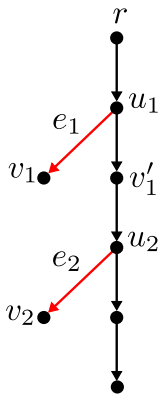


Figure 3.1: A tree where each vertex has children ordered from left to right.

3.3.2 Unfairness minimisation problem

The number of potential public complaints can be deduced from the structure of a vehicle route. We call this number the *unfairness index* of a vehicle route, and denote it by $\mathrm{Uf}(w)$.

Definition 13 *(Complaint, Unfairness) Let (T, d, z, p) be a maintenance plan and $w = (w_1, \ldots, w_\ell)$ a vehicle route on the tree T. Let v be a non-root vertex of T. Let there be i such that the arc $(p(v), v)$ does not belong to $(w_1, \ldots, w_i)$.*

We say that there is a forward complaint *at v if all next-door neighbors of $(p(v), v)$ belong to $(w_1, \ldots, w_i)$.*

We say that there is a backward complaint *at v if there is $j < i$ such that w_j is the previous-door neighbor of the arc $(p(v), v)$ and all the next-door neighbors of w_j preceding arc $(p(v), v)$ belong to $(w_1, \ldots, w_i)$.*

We define the unfairness index Uf(v) of a vertex v to be a number from $\{0, 1, 2\}$ according to the existence of a forward and/or a backward complaint at v.

The unfairness index *of the route w, denoted by Uf(w), is the sum of unfairness indices Uf(v) of all vertices in the maintenance plan.*

An example of a vehicle route with an unfairness index $\mathrm{Uf}(w) = 2$ is in Figure 3.2. There is only one vertex v_2 with a nonzero unfairness index.

Observe that every vertex has an unfairness index of at most two even if there are multiple trips in the route which are the same. It is normal to make only one complaint for each problem regardless of the number of occurrences.

Naturally we obtain an optimisation problem, the goal of which is to find a vehicle route with the minimal number of complaints.

Definition 14 *(Unfairness minimisation problem) Let (T, d, z, p) be a maintenance plan, T is a tree.* Unfairness minimisation problem *is to find a L, c, t, f-vehicle route w with a minimal unfairness index.*

Next we show that this innocent looking problem includes the extensively studied *necklace splitting problem.*

3.3.3 Necklace splitting problem

In 1987 Noga Allon [14] studied an interesting problem in combinatorics, which has a curious interpretation about how to divide a stolen necklace fairly between thieves.

Definition 15 *(k-splitting) Let N be an open necklace, a sequence of $k \cdot n$ beads, chosen from s different colors. There are $k \cdot a_i$ beads of color i, $1 \leq i \leq s$. A k-*splitting *of the necklace is a partition of the necklace into k parts, each consisting of a finite number of nonoverlapping intervals of beads whose union contains precisely a_i beads of color i, $1 \leq i \leq s$. The* size *of the k-splitting is the number of cuts that forms the intervals of the splitting.*

We can imagine that there are precious metal links between the beads and we want to destroy only a small number of them.

Definition 16 *(Necklace splitting problem) Let N be a necklace.* The necklace splitting problem *is to find, for given number k, a k-splitting of necklace N of minimal size.*

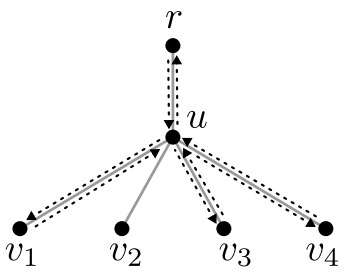


Figure 3.2: The vertex v_2 has both forward and backward complaints in the vehicle route $w = ((r, u), (u, v_1), (v_1, u), (u, v_3), (v_3, u), (u, v_4), (v_4, u), (u, r))$.

If the beads of each color appear contiguously, then at least $k-1$ cuts between the beads of each color are necessary. The following theorem says that it is sufficient for all k-splittings.

Theorem 5 (Noga Alon) *Every necklace with $k{\cdot}a_i$ beads of color i, $1 \leq i \leq s$, has a k-splitting of size at most $(k-1)\cdot s$. The number $(k-1)\cdot s$ is the best possible.*

This theorem has only a topological non-constructive proof so far; Alon's proof uses a transformation of the discrete problem to a continuous coloring of the unit interval.

Complexity of the necklace splitting

The algorithmic complexity of the necklace splitting has been intensively studied. Since the proof for the existence of the splitting of size $(k-1)s$ is not constructive, consequent research has been directed towards constructively finding the splitting.

The following question had been open for a long time: *Can one efficiently find the splitting guaranteed by Theorem* 5? This was finally answered negatively in 2019 by Filos-Ratsikas and Goldberg [15]. We explain this result but first we introduce an algorithmic problem *LEAF*.

Definition 17 (LEAF problem) *An instance of the problem called* LEAF *consists of a graph G of maximum degree* 2*, whose 2^n vertices are represented by* $0,1$ *sequences of length n; G is given by an algorithm that takes as input a vertex and outputs its neighbors. Moreover, the vertex* 0 *has degree* 1*. The goal is to output another vertex of degree* 1*.*

We say that a problem is *PPA-complete* if it is polynomial time equivalent to the LEAF problem. Those problems are a subclass of problems which guarantee the existence of the solution, and its correctness can be checked in polynomial time. The result of Filos-Ratsikas and Goldberg is that necklace splitting guaranteed by Theorem 5 is PPA-complete even for $k = 2$, and thus it is hard to find an optimal solution even though we know it exists.

Another open problem was: *Determine the algorithmic complexity of feasible splitting with the smallest number of cuts.* This is proven to be NP-complete even for 2-splitting and two beads of each color by Bonsman, Eppig and Hochstättler [17]. An alternative proof was made by Meunier [18].

Reduction of necklace splitting to the unfairness minimisation problem

After a short discussion of the complexity of necklace splitting, we prove in the following theorem that the unfairness minimisation problem is at least as complex as necklace splitting.

Theorem 6 *There exists a polynomial reduction of the necklace splitting problem to the weighted unfairness minimisation arc routing problem for trees.*

Proof. We have given an instance of the necklace splitting problem. The necklace N of length nk has to be partitioned into k parts each containing a_i beads of color i, $1 \leq i \leq s$. The number of colors is s.

We describe a construction of an instance of the unfairness minimisation problem whose solution can be transformed into a solution of the original necklace splitting problem and vice versa. We construct a maintenance plan $(T, d, 1, p)$ and define parameters L, c, t, f of a vehicle route in the way that the optimal vehicle route with a minimal unfairness index will consist of k trips from depot corresponding to k parts of the necklace.

The graph of the network is a tree T. We have a constant priority function $p(e)$, $e \in E(T)$ and also the function of time limits $t(p(e)) = 1$ is constant for all edges. The number of traverses of each arc has the upper bound equal to k, i.e., $f(e) \leq k$ for each edge e.

First, we create a path $P = (v_1, \ldots, v_{kn})$ and we assign the depot d to be the first vertex v_1. To each vertex v_i, $1 \leq i \leq kn$ we attach new vertex u_i by edge $e_i = \{v_i, u_i\}$ of length appropriate to the color of the i-th bead of the necklace.

For each color r, $1 \leq r \leq s$, we define the appropriate length M_r recursively:

$$M_1 = 1 + nk$$
$$M_{i+1} = 1 + nk \sum_{q \leq i} M_q$$

The total limit L for the length of the vehicle route and the capacity constant c remain to be defined.

$$L = 2k \cdot (nk + \sum_{r=1}^{s} a_r M_r)$$
$$c = \frac{1}{k}$$

The capacity constant determines the length of each trip to be at most $2(nk + \sum_{r=1}^{s} a_r M_r)$. Multiplication by two means each edge is traversed in both directions. The first term in the brackets is the capacity for traversing the whole path P and the second term in the brackets is the capacity for traversing exactly those edges e_i corresponding to beads of one part of the feasible splitting.

A solution to the unfairness minimisation problem is vehicle route $w = (T_j)_1^k$. We observe that, since it minimises the unfairness index, we can assume without loss of generality that the complaints are only at vertices of $\{u_1, \ldots, u_{nk}\}$. The complaints at vertices u_i correspond to the splits of the necklace. If u_i has the forward complaint, there must be a split after the i-th bead of the necklace. The backward complain is equivalent to the split before the i-th bead. Hence the sum of the complaints at the vertices is equal to the minimum size of the necklace splitting.

□

As a consequence, a weighted variant of the unfairness minimisation problem is PPA-complete.

3.3.4 Sum packing problem

As a follow-up to the proof of Theorem 6, we show a connection between the unfairness minimisation problem and the sum packing problem of Erdös [19].

We note that in the construction of the proof of Theorem 6, the tree T has at least $(nk)^s$ vertices.

Question 1 *It is not clear if a construction exists with T which has a polynomial number of vertices (in the product nks) and no weights.*

As described above, the necklace splitting is, from the complexity point of view, already very interesting for the number of thieves $k = 2$, and each $a_i = 1$.

Relating to this special case of the unfairness index and Question 1, in the construction of a tree T with the desired properties in the proof of Theorem 6 the next question came into our view.

Question 2 *Is there a set of natural numbers $M_1, \ldots, M_s$ such that*

- *for each $i \in \{1, \ldots, s\}$ is M_i bounded by a fixed power of s and*
- *all partial sums of M_i's are pairwise distinct.*

Clearly, when the number of thieves $k = 2$ and each $a_i = 1$ then the proof of Theorem 6 works for any set of numbers $M_1, \ldots, M_s$ with the property in Question 2.

It turns out that the answer to Question 2 is negative. This is related to a very nice part of the combinatorial number theory which we now explain.

Definition 18 *(Set with distinct subset sums) A set S of positive integers has* distinct subset sums *if the set $\{\sum_{x \in X} x \colon X \subset S\}$ has $2^{|S|}$ distinct elements.*

For example, any set of distinct powers of number 2 has the distinct subset sums property. More examples of sets with distinct subset sums are $\{3, 5, 6, 7\}$ and $\{6, 9, 11, 12, 13\}$. We mention a lower bound for the value of the maximum in the sets with the distinct subset sums property.

Definition 19 *Let $f(n) = \min\{\max S \colon |S| = n$ and S has distinct subset sums$\}$*

Paul Erdös conjectured in 1931 that for some constant c

$$f(n) \geq c2^n.$$

The best known lower bound, up to the constant, has been proven by Erdös and Moser [20] in 1955,

$$f(n) \geq 2^n/(10\sqrt{n}).$$

This negatively resolves Question 2 and we observe that for a given number of colors s in the necklace splitting, the sizes of M_i's in the proof of Theorem 6 are not bounded by a polynomial function in s.

The case where only partial sums of any two elements need to be different is also studied extensively. Such sets of natural numbers are called *Sidon sets* or *Sidon sequences* [21].

Definition 20 *(Sidon sequence) Let $a_1 < a_2 < \ldots$ be a sequence of positive integers, and suppose that the sums $a_i + a_j$ (where $i < j$) are all different. Such a sequence is called a* Sidon sequence.

There are Sidon sets in $\{1, \ldots, n\}$ of $\sqrt{n}$ elements (see e.g. [21]).

3.3.5 Further work

In this closing part of the unfairness minimization problem, we write down conjectures about several variations of the problem.

Question 3 *Is there a polynomial algorithm to solve the unweighted unfairness minimization problem, even for a given instance of a restricted arc routing problem of a maintenance plan (T, d, z, p) with parameters of vehicle route L and c?*

We note that, since the answer to Question 2 is negative, we cannot directly use the results on the complexity of necklace splitting to argue that the unfairness minimization problem is NP-complete. We conjecture the following.

Conjecture 3 *The unfairness minimization problem is NP-complete even if the constant capacity property is required, i.e., capacity c does not depend on input tree T.*

The next conjecture assumes that we want to find a vehicle route composed of a fixed number of trips and the edges of each trip are traversed in a particular order.

Conjecture 4 *The existence of a vehicle route and the unfairness minimization problem become polynomial if we require the strong capacity property of the vehicle route.*

In the last part of this chapter we design algorithms for the problem of the existence of a restricted vehicle route on a tree. We first introduce an auxiliary problem which we call the *tree cutting problem*.

3.4 Tree cutting problem

Our arc routing problem is to find a vehicle route with multiple trips. Below, we introduce a problem which we employ to solve the restricted arc routing problem specified in Section 3.2 for a given maintenance plan with a constant priority function.

Definition 21 (Tree cutting problem) *The tree cutting problem is to find, for a given tree T rooted in r and set of numbers $t_1, \ldots, t_k$, a cover of the graph T by subtrees $T_1, \ldots, T_k$ rooted in r of sizes $t_1, \ldots, t_k$.*

3.4.1 Algorithms

First, we discuss an algorithm for the tree cutting problem with $k = 2$, i.e., finding two rooted subtrees of given sizes covering T. Afterwards, we introduce the algorithm for general k and then a pseudocode, for $k = 2$, illustrates the idea of general algorithm in a shorter way.

We note that if T has bounded degrees, $k = 2$ and $t_1 = t_2$, a polynomial algorithm for the Tree Cutting Problem follows directly from Theorem 4 applied to $F = 2$, all edges have the same priority, and $t_1 = t_2 = 1/2\, cL$. However, its implementation is not usable in practice.

Greedy algorithm

A natural approach is to use a greedy algorithm which constructs trees of suitable sizes. Greedy algorithms can often be used as a heuristic in practice. Below we define an algorithm of greedy type and we create a counterexample for which the naive steps of such an approach do not lead to an optimal solution.

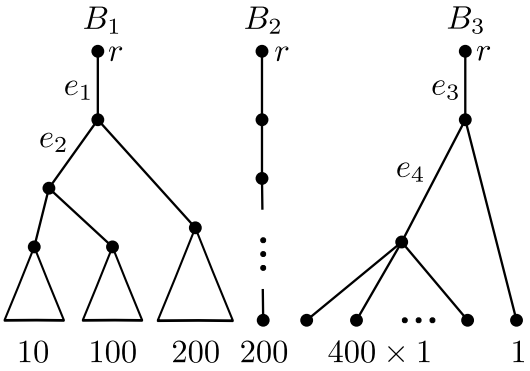


Figure 3.3: Counterexample for correctness of the greedy algorithm. The graph T is a union of three branches of sizes 315, 200, 403. For $R = \{e_1\}$ the minimal discrepancy is $d_1 = 111$ and $A_1^1 = \{113, 201\}, A_2^1 = \{200, 403\}$. For $R = \{e_3\}$ there is the best discrepancy $d_3 = 113$, $A_1^3 = \{315, 200\}, A_2^3 = \{401, 1\}$.

We have given a tree T rooted in r and t_1, t_2 sizes of trees T_1, T_2. We try to find *"cutting vertices"* and a partition of their children into two sets. The branches rooted in the vertices of the first, set we put into T_1 and the branches rooted in the vertices of the second set, we put into T_2.

Observe that the path from the *cutting vertex* to the root of T is in both trees T_1, T_2. We always denote by R the common subtree of T_1 and T_2. In every step of the greedy algorithm, we update the subtree R by adding one edge. At the beginning, R contains only root r.

Let $v_1, \ldots, v_m$ be vertices which are not in R but their parents (cutting vertices) are in R. We define branches $B_i = B(v_i) \cup \{v_i, p(v_i)\}$, $1 \leq i \leq m$. We denote by b_i the size of the branch B_i.

We have a solution to the tree cutting problem if there exists a partition of integers b_i into two sets A_1, A_2 so that $\sum_{b_i \in A_1} b_i = t_1 - |E(R)|$ and $\sum_{b_i \in A_2} b_i = t_2 - |E(R)|$. The solution is formed by trees T_1, T_2 obtained as a union of R with the corresponding branches B_i such that b_i is in A_1, in A_2 respectively.

If there is a prefect partition of branches, we are done. Otherwise, for each edge $e_i = \{v_i, p(v_i)\}$ we calculate a new partition A_1^i, A_2^i. The discrepancy of the partition is

$$d_i = \left| |E(R)| + \sum_{b_j \in A_1^i} b_j - t_1 \right| + \left| |E(R)| + \sum_{b_j \in A_2^i} b_j - t_2 \right|.$$

For the minimal discrepancy d_i we add into R the edge e_i and continue.

Assume we want two subtrees T_1, T_2 of the same size for graph G in Figure 3.3. We can see that the optimal solution is given by $R = \{e_3, e_4\}$ (it create a lot

of small branches) but the greedy algorithm will add in the first step edge e_1 and next the e_2. The greedy approach never retracts wrong decisions and it never comes to the optimal solution with trees of sizes $t_1 = t_2 = 460$ and a discrepancy equal to zero.

Polynomial algorithm

Next, we show a polynomial algorithm for the tree cutting problem based on dynamic programming. The algorithm works for general trees and thus it lifts the restriction of a bounded degree imposed by Theorem 4.

Theorem 7 *Let k be any fixed number. There is a polynomial algorithm to solve the Tree Cutting Problem.*

Proof. We are given a tree T rooted in a vertex r and integers $s_1, \ldots, s_k$. Let us denote by n the number of vertices of T.

We need to cover the graph T with trees of given sizes. To do that we will construct a cover for all possible sizes of trees. This set of covers is denoted by $F(v)$.

Formally, we proceed in two steps. First, define for each $v \in V(T)$ a set $F'(v)$; the elements of $F'(v)$ are all k-tuples of trees $(T_1, \ldots, T_k)$ with root v such that:

$$B(v) = \bigcup_{i=1}^{k} T_i.$$

Each k-tuple in $F'(v)$ is a cover of the branch $B(v)$ by trees $T_1, \ldots, T_k$.

We define the size of $(T_1, \ldots, T_k)$ as the vector $(t_1, \ldots, t_k)$ where $t_j = |E(T_j)|$, $1 \leq j \leq k$.

We define an equivalence on $F'(v)$: $(T_1, \ldots, T_k)$ and $(T'_1, \ldots, T'_k)$ are equivalent if $\forall i \in \{1, \ldots, k\} : t_i = t'_i$.

Secondly, for each vertex v we define the set $F(v)$ to be the set of all representatives of this equivalence.

We denote by $S(v)$ a set of the sizes of the elements in $F(v)$. We note that $|S(v)| \leq n^k$ because each tree has at most n edges and there are k trees in every k-tuple.

We will construct $F(v)$ for all $v \in T$ recursively.

- Let v be a leaf: $F(v) := \{(\emptyset, \ldots, \emptyset)\}$.
- Let v be a parent of vertices $v_1, \ldots, v_m$ and we assume for all $i = 1, \ldots, m$ $F(v_i)$ are determined.

  Later, we will define for v and $1 \leq i \leq m$ a k-tuple $F_{v_i}(v)$ and we let

  $$F(v) := F_{v_m}(v).$$

  Each $F_{v_i}(v)$ contains representatives of covers of the subtree induced by the vertex v and branches rooted in its children $v_1, \ldots, v_i$.

Now we describe an augmenting step used for the construction of $F(v)$. We call it Additive step.

Additive step: We construct from $(T_1, \ldots, T_k)$ k-tuples $(T'_1, \ldots, T'_k)$ by addition of the edge $\{v_i, v\}$ in every possible way. There are two cases dependent on the size of the cover.

1. Case $t_i = 0$ for each $1 \leq i \leq k$:
   we add the edge for every nonempty subset of indices

$$\forall I \subset \{1, \ldots, k\}, |I| \geq 1$$
$$j \in I : T'_j = T_j \cup \{v, v_i\}$$
$$j \notin I : T'_j = \emptyset$$

2. Case $\exists\, i \in \{1, \ldots, k\} : t_i \neq 0$
   Let $J = \{i \mid t_i = 0\}$, we add the edge for all nonempty trees and every subset of empty trees

$$\forall I \subset J$$
$$j \notin J : T'_j = T_j \cup \{v, v_i\}$$
$$j \in I : T'_j = T_j \cup \{v, v_i\}$$
$$j \notin J \setminus I : T'_j = \emptyset$$

This finishes the description of the Additive step.

For the construction of $F_{v_1}(v)$ we only need the additive step. For the construction of $F_{v_{i+1}}(v)$ we need to merge $F_{v_i}(v)$ with $F(v_{i+1})$. This is described below.

- Construction of $\boldsymbol{F_{v_1}(v)}$**:**

  We start with $F_{v_1}(v) = \emptyset$. For each $(T_1, \ldots, T_k) \in F(v_1)$ we construct by the additive step a set of k-tuples $(T'_1, \ldots, T'_k)$ which we add into $F_{v_1}(v)$.

- Construction of $\boldsymbol{F_{v_{i+1}}(v)}$**:**

  In this case we proceed in two steps.

  First we construct $F'(v_{i+1})$ by additive steps applied to $F(v_{i+1})$. Specifically, we start with $F'(v_{i+1}) = \emptyset$. For each $(T_1, \ldots, T_k) \in F(v_{i+1})$ we construct by the additive step a set of k-tuples $(T'_1, \ldots, T'_k)$ which we add into $F'(v_{i+1})$.

  Secondly, we merge $F_{v_i}(v)$ and $F'(v_{i+1})$ again, in two steps as follows.

  First, for all $(T_{11}, \ldots, T_{1k}) \in F_{v_i}(v)$ and all $(T_{21}, \ldots, T_{2k}) \in F'(v_{i+1})$ we create a k-tuple $(T'_1, \ldots, T'_k)$ defined element-wise

$$\forall i \in \{1, \ldots, k\} : T'_i = T_{1i} \cup T_{2i}.$$

  We add into $F_{v_{i+1}}(v)$ the created k-tuple.

  Finally, we clean the set $F_{v_{i+1}}(v)$ by keeping only the representatives of equivalence classes.

The described construction of the set $F(r)$ determines the set $S(r)$ of sizes. We have a solution to the tree cutting problem if and only if the k-tuple $(s_1, \ldots, s_k)$ is in $S(r)$.

By analyzing the above procedure, we need at most two times $n^k \times n^k$ steps for the addition of one edge. Hence the complexity of the algorithm is asymptotically n^{2k+1} because there are at most n edges. So there exists a polynomial algorithm for the tree cutting problem with fixed k. □

A shorter representation of the idea from the general proof is in the pseudocode below describing a recursive algorithm for testing the existence of a tree cutting for $k = 2$. The pseudocode works with the sizes of the pairs of trees which cover the input graph. It is straightforward to adapt the algorithm to store the trees also.

An input of the `TreeCut()` algorithm is a pointer `root` to the structure of a graph which we want to cover by trees of sizes `T1` and `T2`. For each vertex of the graph a set of pairs is created`subtrees_sizes[]` (no duplications inside). This set stores all sizes of trees which cover the input graph.

```
 1 TreeCut(root, T1, T2):
 2 |  CountSizes(root)
 3 |
 4 |  if (T1, T2) is in root.subtrees_sizes:
 5 |     print "There exists a tree cutting"
 6 |  else
 7 |     print "There does not exist a tree cutting"
 8
 9 CountSizes(root):
10    root.subtree_sizes <- [(0,0)]
11    for each son of root do:
12    |  if son is not a leaf:
13    |     CountSizes(son)
14    |
15    |  if son is a leaf:
16    |  |  new_sizes <- []
17    |  |  for (t1, t2) in root.subtree_sizes do:
18    |  |     new_sizes.Add(t1 + 1, t2 + 1))
19    |  |     new_sizes.Add(t1 + 0, t2 + 1))
20    |  |     new_sizes.Add(t1 + 1, t2 + 0))
21    |  |  root.subtree_sizes = new_sizes
22    |  else:
23    |  |  for (s1, s2) in son.subtrees_sizes do:
24    |  |  |  new_sizes <- []
25    |  |  |  for (t1, t2) in root.subtree_sizes do:
26    |  |  |     new_sizes.Add(t1 + s1 + 1, t2 + s2 + 1)
27    |  |  |     if s1 == 0:
28    |  |  |        new_sizes.Add(t1, t2 + s2 + 1)
29    |  |  |     if s2 == 0:
30    |  |  |         new_sizes.Add(t1 + s1 + 1, t2)
31    |  |  |  root.subtree_sizes = new_sizes
```

The time complexity can be deduced from the `else` branch in the procedure `CountSizes()`. There are two `for` cycles going over sets of size at most n^2 so at most n^4 steps are needed. We do those steps for every son of every vertex. Hence at most n-times. We obtain time complexity of the algorithm $\mathcal{O}(n^5)$.

For space complexity, in each vertex we store a list of length n^2. So the space complexity is $\mathcal{O}(n^3)$. If we consider a trivial remembering of trees, the needed space increases to $\mathcal{O}(n^4)$.

We are finally ready to construct and analyse the algorithm for the restricted routing problem on trees.

3.4.2 Restricted arc routing problem on trees with a constant priority function

We recall that in the restricted arc routing problems, we assume:

- the monotonic property holds (= priorities are non-decreasing on every path from the depot to a leaf),
- $z = 3$,
- $t(p) = 1/(4 - p)$ and
- f is not specified.

This is the interesting case for the winter road maintenance in the Czech Republic. The assumptions reduce the parameters of the problem to

1. total length L,
2. priority p,
3. capacity c.

An additional assumption of this section is a constant priority function. We denote the value $t := t(p(e))$.

In the case that the constant capacity property is not required, i.e., the capacity $c = c(T)$ depends on the tree T, we do not know how to find a restricted vehicle route efficiently.

Question 4 *Is there a polynomial algorithm for finding a vehicle route when the priority is constant but the capacity may depend on the input tree?*

The case of the fixed capacity c already admits a polynomial algorithm.

Theorem 8 *There is a polynomial algorithm for finding the solution to the restricted arc routing problem for trees, a L, c, t, f-vehicle route, on a maintenance plan $(T, d, 3, p)$ when the priority is constant and the capacity is a constant fraction not depending on the input tree.*

Proof. We distinguish several cases.

1. $t \geq 1$, $c \geq 1$:
   - priority condition and capacity condition always hold
   - necessary and sufficient condition for existence of L, c, t, f-vehicle route is: $|E(T)| \leq \frac{1}{2} L$
2. $t \geq 1$, $c < 1$:
   - priority condition always holds

- we want to construct subtrees $T_1, \ldots, T_k$ rooted at r such that $k := \left\lceil \frac{1}{c} \right\rceil$,

$$\forall i \in [k] \colon |E(T_i)| \leq cL,$$

$$T = T_1 \cup \ldots T_k \text{ and}$$

$$\sum_{i \leq k} 2\,|E(T_i)| \leq L.$$

This is achieved by the algorithm for the tree cutting problem described in the proof of Theorem 7. A solution is any collection of k trees with sizes t_i satisfying $t_i \leq cL$ for each $i \leq k$, and

$$\sum_{i \leq k} 2t_i \leq L.$$

3. $t < 1$, $t \leq c$:

   - the capacity condition holds if the priority condition is satisfied
   - the necessary and sufficient condition for existence of L, c, t, f-vehicle route is: $2\,|E(T)| \leq tL$

4. $t < 1$, $c < t$:

   - this case is equivalent to Case 2 for $L' := tL$, $t' = 1$ and capacity $c' := \frac{c}{t}$

Solution of cases 1 and 3 can be a DFS order of a tree if the necessary condition holds, otherwise there is no solution. Case 4 is reduced to Case 3. The proof of Theorem 7 gives us a polynomial algorithm for Case 2. This proves that we have a polynomial algorithm for all cases.

□

3.4.3 Restricted arc routing on trees with more priorities

When the priority function $p \colon E \to \{1, \ldots, z\}$ is not constant, for a given graph of a road network, we can not easily generalize results written above. Presently, we do not know if a polynomial algorithm exists even for the very restricted case when the capacity c is a fixed fraction not depending on the input.

However, we believe (and express this believe in Conjecture 4) that everything about restricted routes is polynomial when the strong capacity property holds.

4. Application in Winter Road Maintenance

The routing for winter road maintenance vehicles is especially complex. It is not possible to solve it with one universal algorithm. Every city, region, or state has different conditions based on geography, meteorology, economics, technology, etc. [11]. Even if the conditions are the same, they often have different importance.

Algorithms for finding an optimal solution for the rural postman problem are studied but they are not useful for large graphs that are needed in real world applications.

We created a model of a road network and an algorithm which is used for planning vehicle routes in the Czech Republic. It has been used in the Plzeň region for the first time and was later used in the Liberec region. This section describes our results which I also presented in the Workshop on Arc Routing Problems [6]. Wider context and more details can be found in our joint paper with Fink and Loebl [2].

We describe the model and show the results of the optimization in the Plzeň region. We explain the output of the algorithm in the context of the input graph structure.

4.1 Graph of the road network

A preprocessing of data creates a graph of the road network. During this process road exchanges with neighboring regions are handled, road segments with vertices of degree two are contracted and other details are managed.

In compliance with Definition 1 of *road network*, we obtain a graph $G = (V, E)$ on 1,719 vertices and 2,280 edges.

Edges have a length given in meters by function $\alpha\colon E \to \mathbb{Z}^+$, the total length of the road network is $\sum_{e \in E} \alpha(e) = 4,860$ km.

The priority function uses a service hierarchy defined by the legislation based on traffic volume (see figure 4.1a). It partitions the roads into four classes $p : E \to \{1, 2, 3, 4\}$ and so the number of categories is $z = 4$. Roads of the first priority have the best service and the shortest time limit, which specifies the frequency of maintenance. The fourth class contains roads that are not maintained during the winter season. Time limits associated with road priorities are given in Table 4.1. The fourth priority roads have no time limit for maintenance by law, but they can be used for deadheading. We say that those edges are not maintained, but in practice these edges can be maintained by other companies.

The methods of the winter road maintenance are: chemical, inert, and snow-

priority	1	2	3	4
time limit	3 h	6 h	12 h	∞

Table 4.1: Time limits of maintenance

plow, $M = \{c, i, s\}$. Each edge is assigned a type of maintenance by function $m : E \to M$. There are two types of vehicles used in the Plzeň region. The first type spreads chemicals and snowplows. The second type spreads inert materials and snowplows. (Mixed types of vehicles are rare and we do not use them in this model.)

There is a set of 34 depots $D \subset V$. Chemical vehicles can have their base in 28 of them which have salt storage. Inert vehicles can use 30 depots with storage for inert materials.

4.2 Structure of the road network

Service hierarchy

Our road network hierarchy is shown in Figure 4.1a. First priority edges (19%, Table 4.2) form a crucial part of the network. This subset of edges is a connected component. Priorities two (24%) and three (57%) make the network graph more connected, but on their own, they form disconnected graphs. This structure provides a reason for why the monotonic property in Definition 7 is formulated, and also a reason for why a route of one vehicle corresponds to a tree rather than a circuit. Less important roads (edges) are far away from a depot, and closer to the leaves of a subgraph maintained by one vehicle. In a time of crisis, maintenance efforts can focus on the essential part of the network, which is still connected even if some third priority routes are temporarily excluded.

type of maintenance by priority	**1**	**2**	**3**	sum
chemical	840	1080	1472	3392 (70 %)
inert	42	65	569	676 (14 %)
snowplow	8	15	724	747 (16 %)
sum	**890** (19 %)	**1160** (24 %)	**2765** (57 %)	**4815**

Table 4.2: Road lengths [km] of each priority and maintenance method

Priorities also give us a lower bound for kilometers driven. A working shift is 8 hours. Time limits in Table 4.1 imply the frequency of maintenance and the multiplicity of edges traversed in any feasible solution. When we multiply the length and multiplicity of each arc and sum these, we obtain a lower bound for the total length of routes traversed by all vehicles: 7,755 km (see Table 4.3). This is not tight and is related to the distribution of depots and the structure of the network graph explained in the following paragraphs.

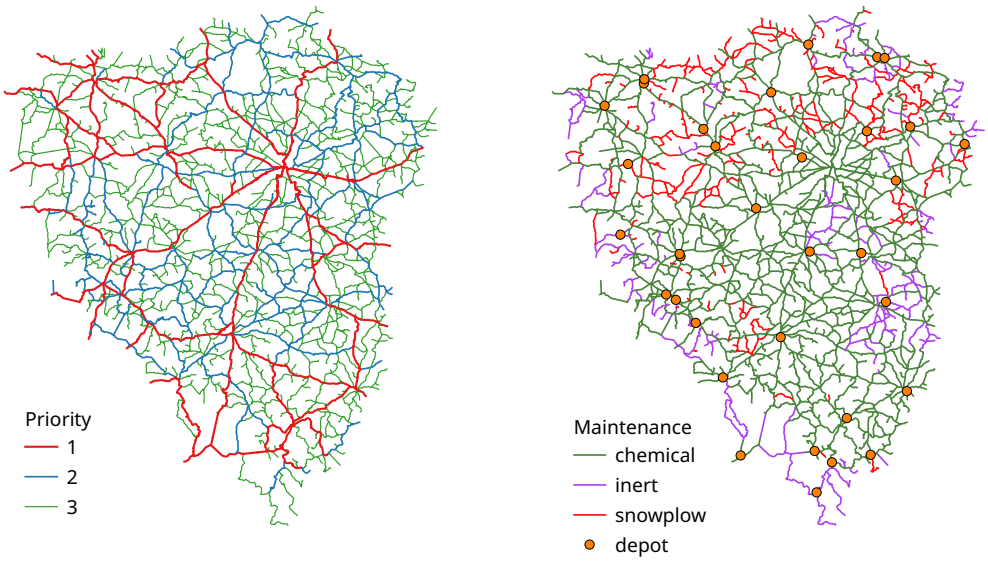
Types of maintenance

Positions of depots are in Figure 4.1b. In the same figure, we can see that a great amount of edges are treated by chemicals (70 %, Table 4.2). It is a trend to increase the number of roads maintained by chemicals. But there are smaller areas where it is not allowed. We can see inert roads (14 %) forming small disconnected components of the road network. Since depots are not placed in each of these

priority	1	2	3	**sum**
length	890	1160	2765	4815
multiplicity	3	2	1	
multiplied length	2670	2320	2765	**7755**

Table 4.3: Length of roads including frequency of maintenance [in kilometers]

components, there has to be some deadhead. The goal of the optimization is to minimize this, but it is clear that vehicles with the inert type of maintenance have to traverse some chemically treated roads to reach their place of operation. According to the structure of the network, it is necessary to use at least 143 km of deadhead by vehicles of inert type and 7 km by vehicles of chemical type. It is 150 km in total, this result was obtained using the linear programming model from the paper about arc routing in winter road maintenance [2].



(a) Priority levels (b) Method of maintenance

Figure 4.1: Priority levels and maintenance methods of the road network of Plzeň region

4.3 Optimization problem

We have a graph of a road network. A solution to the winter road maintenance problem is a cover of the set of maintained edges, i.e., $p(e) \neq 4$, by subsets $C = \{C_1, C_2, \ldots, C_k\}$. These subsets correspond to vehicles of winter road maintenance and there is a pre-maintenance plan $(C_i, P_i, d_i, \alpha, 4, p)$ for all $i \in \{1, \ldots, k\}$. The following conditions have to be satisfied:

- The total length of the plan is at most 90 km.

$$\sum_{e \in C_i} \alpha(e) \leq 90,000 \text{ m}$$

- All maintained edges are maintained by a vehicle of one type (chemical/inert).

$$\forall e \in P_i \colon m(e) \in \{c, s\} \text{ xor } m(e) \in \{i, s\}$$

- The depot stores the appropriate maintenance material.

$$(\{c, i\} \cap \{m(e) \colon e \in P_i\}) \subseteq m(d_i)$$

- The monotonic property (Definition 7) holds.

There are two objective functions. In the first place we minimize the number of vehicles, and then we minimize the number of deadhead kilometers.

4.4 Algorithm

At the beginning we find an initial solution by creating a set of pre-maintenance plans $\{(C_i, P_i, d_i, \alpha, 4, p)\}_i$, corresponding to the set of vehicles.

Each plan i is created by choosing any edge e that has not been maintained yet $e \in E \setminus \bigcup_{j<i} P_j$ and defining $P_i := \{e\}$.

Then we add more edges by greedy steps until the next edge exceeds the maximum size of the plan, 90 km.

Chosen edges added into P_i have the type of maintenance $m(e) \in \{c, s\}$ or $m(e) \in \{i, s\}$. The size of a plan is $\sum_{e \in C_i \setminus P_i} \alpha(e) + \sum_{e \in P_i} \alpha(e) \cdot (4 - p(e))$ and the set of edges C_i is created by using a heuristic for finding a Steiner tree including maintained edges P_i and closest depot d_i with storage of appropriate material type.

The heuristics must keep the monotonic property.

After the initialization, we move on and repeat the reducing steps while we improve our solution. There are two ways to eliminate a vehicle.

The first one is a union of two small plans by creating a new one $P_i := P_i \cup P_j$.

The second is by distributing any plan into other plans in its neighborhood. To enable this distributing step, we first need to create some 'space' in the neighboring plans.

That is the reason for creating a layered graph, where layers correspond to the distance from the considered plan. For the layer which is the furthest, we use precisely defined switching rules and we attempt to fill up its plans with branches of the plans in nearer layers, with the help of a Bin packing heuristic; the bins are the sets P_i. Then we continue with closer layers until ultimately we can make the considered plan empty.

4.5 Results

Our model of winter road maintenance was used for creating plans of vehicles for the winter season at the turn of the years 2019 and 2020 in the Plzeň region.

The output of our model was adjusted by experts with background knowledge of the local conditions.

The plans for winter 2018-2019 (before our optimization) had 113 vehicles. We optimized these plans and our solution had 93 vehicles. After evaluation of the risks related to such a huge reduction, 102 routes have been used in the winter of 2019-2020. The additional routes were added in Plzeň city and mountain areas with a larger volume of snow during winter.

Our model reduced the number of required maintenance vehicles by 17 % (19 vehicles). After the experts' adjustment, 10 % (11 vehicles) have been removed from service.

An overview of our solution is described in Table 4.4. The value of our first objective function is 93 vehicles. From the structure of the road network (Section 4.2), we know that the number of driven kilometers is at least 7755 + 150 = 7,905 km. The limit for one vehicle is 90 km. This means that the theoretic lower bound is that at least 88 vehicles are needed. The second objective function is the number of deadhead kilometers. The solution includes 361 km of deadheading and the theoretical lower bound based on linear programming proposes at least 150 km.

method	chemical		inert		**sum**	
	km	%	km	%	km	%
number of cars	78		15		**93**	
total limit (90 km per car)	7020		1350		8370	
maintenance (% of limit)	6709	(95.57)	1046	(77.48)	**7755**	(92.65)
deadhead (% of limit)	155	(2.21)	206	(15.26)	**361**	(4.31)
unused (% of limit)	156	(2.22)	98	(7.26)	**254**	(3.03)

Table 4.4: Summary of the solution

The table shows that inert vehicles have a larger amount of deadheading. That corresponds to a high lower bound of 143 km needed to reach the parts of the network maintained by the inert method. Other causes of deadhead can be small degrees of depot vertices, i.e., multiple vehicles traverse the same road and only one maintains it, the monotonic property, and suboptimality of the solution.

The last row of Table 4.4 contains an unused reserve of the vehicles. If we consider an ideal utilization of maintenance capacity, 93 vehicles can service 8,370 km of roads. Hence, our solution is efficient because only 3 % of operating capacity is not used.

Conclusion

We built up basic definitions and formulated arc routing problems for winter road maintenance in the language of graph theory. We introduced terms such as a graph of a road network, a maintenance plan, and a vehicle route. The problem of whether or not a vehicle route exists is a NP-complete problem as can be seen from a reduction of the number partition problem to it.

A perfect solution for the problem of winter road maintenance could be a decomposition of a graph (of a road network) to closed Eulerian walks. Due to a small degree of vertices with depots, such a perfect solution does not exist in practice. Consequently, we look for a cover of a graph by subgraphs maintained by a single vehicle. While we look at a subgraph of a network maintained by one vehicle, there are a small number of cycles. Moreover each edge (road) has to be maintained in both directions so we can naturally represent the subgraph as a tree obtained from the depth first search algorithm.

We formulated a general *arc routing problem on trees*. We then specified restrictions used in the Czech winter road maintenance model and a few more conditions which may affect the complexity of the problems.

A new *unfairness minimisation problem* was introduced. There is a formalization of the properties of a vehicle route which typically precede a situation when residents feel skipped in a vehicle route and make a complaint about inappropriate maintenance. Because the number of complaints is a measurement of the quality of service, it is important to prevent complaints.

We proved The model reduced the number of required maintenance vehicles by 17 % (19 vehicles). After the experts' adjustment, 10 % (11 vehicles) have been removed from service.

that the unfairness minimisation problem is PPA-complete by reduction of the necklace splitting problem to it. There are also connections between the unfairness minimisation problem, Sidon sequences (sets with distinct sums of pairs of elements), and the sum packing problem (sets with distinct all subset sums).

This thesis extends the research of Fink and Loebl [1]. They proved a theorem which states there is a polynomial algorithm to determine the existence of a vehicle route for a given road network with a bounded degree of vertices. The proof is usable neither for general trees with an unbounded degree nor for the construction of a vehicle route in practice. For these aims we formulated a *tree cutting problem* and described a polynomial algorithm based on dynamic programming.

We used the algorithm of tree cutting for solving a *restricted arc routing problem* on trees. We proved that the existence of a vehicle route for a network with a single priority of roads is polynomial. The decision can be made by checking a linear condition or by reducing it to the tree cutting problem.

Finally we illustrate the theoretic model with application in winter road maintenance in the Czech Republic. We explain the representation of a road network of the Plzeň region by the graph model. We describe the structure of the graph which motivated the formulation of the restricted arc routing problem on trees. Because this thesis is focused on single vehicle routing, there is only a brief de-

scription of the algorithm for the multiple vehicle routing problem and more details can be found in our collective work with Fink and Loebl [2]. In the Czech Republic there is a low rate of unemployment and a large number of drivers are going into retirement. The suggested solution shown solved the problem of having to find new drivers and resulted in a reduction in the number vehicles used for winter road maintenance by ten percent.

The model reduced the number of required maintenance vehicles by 17 % (19 vehicles). After the experts' adjustment, 10 % (11 vehicles) have been removed from service.

Bibliography

[1] Fink, Jiří, Loebl, Martin. *Arc-routing for winter road maintenance* [online]. preprint, 2019 [accessed 2020-05-25]. URL: https://arxiv.org/abs/2001.08416v1

[2] Fink, Jiří, Loebl, Martin, Pelikánová, Petra. *A New Arc-Routing Algorithm Applied to Winter Road Maintenance* [online]. preprint, 2019 [accessed 2020-05-25]. URL: https://kam.mff.cuni.cz/~loebl/clanky/snow_packing.pdf

[3] Huang, Hao, Ma, Jie, Shapira, Asaf, Sudakov, Benny, Yuster, Raphael. *Large feedback arc sets, high minimum degree subgraphs, and long cycles in Eulerian digraphs*. Combinatorics, Probability and Computing [online]. 2013, vol. 22, no. 6, p. 859–873 [accessed 2020-05-25]. DOI: 10.1017/s0963548313000394

[4] Sullivan, Blair D. *A summary of results and problems related to the Caccetta-Häggkvist conjecture*. preprint, 2006 [accessed 2020-05-25]. URL: http://arxiv.org/abs/math/0605646v1

[5] Hoàng, Chính, Reed, Bruce. *A note on short cycles in diagraphs*. Discrete mathematics [online]. 1987, vol. 66, no. 1-2, p. 103–107 [accessed 2020-05-25]. DOI: 10.1016/0012-365x(87)90122-1

[6] Pelikánová, Petra. *Winter road maintenance in Czech Republic*. Workshop on Arc Routing Problems. Pizzo. Presentation. 2019.

[7] Aharoni, Ron, DeVos, Matthew, Holzman, Ron. *Rainbow triangles and the Caccetta-Häggkvist conjecture*. Journal of Graph Theory [online]. 2019, vol. 92, no. 4, p. 347-360 [accessed 2020-05-25]. DOI: 10.1002/jgt.22457

[8] DeVos, Matt, Drescher, Matthew, Funk, Daryl, Hermosillo de la Maza, Sebastián González, Guo, Krystal, Huynh, Tony, Mohar, Bojan, Montejano, Amanda. *Short rainbow cycles in graphs and matroids* [online]. preprint, 2018 [accessed 2019-05-20]. URL: https://arxiv.org/pdf/1806.00825

[9] Pelikánová, Petra, Spirkl, Sophie, Šťastná, Aneta. *Notes about Rainbow Cycles* [online]. DIMACS CoSP REU International 2019, Research Experience for Undergraduates. 2020, p. 24–29 [accessed 2020-05-25]. URL: https://iti.mff.cuni.cz/series/2020/680.pdf

[10] Perrier, Nathalie, Langevin, André, Campbell, James F. *A survey of models and algorithms for winter road maintenance. Part I: system design for spreading and plowing*. Computers & Operations Research [online]. 2006, vol. 33, no. 1, p. 209 – 238 [accessed 2019-08-06]. DOI: 10.1016/j.cor.2004.07.006

[11] Perrier, Nathalie, Langevin, André and Campbell, James F. *A survey of models and algorithms for winter road maintenance. Part III: Vehicle routing and depot location for spreading*. Computers & Operations Research

[online].2007, vol. 34, no. 1, p. 211 - 257 [accessed. 6 . August 2019]. DOI: 10.1016/j.cor.2005.05.007

[12] KWAN, Mei-Ko. *Programming method using odd or even pints.* Acta Mathematica Sinica. 1960, vol. 10, p. 263–266.

[13] MERTENS, Stephan. *The easiest hard problem: Number partitioning.* Computational Complexity and Statistical Physics [online]. 2006, vol. 125, no. 2, p. 125–139 [accessed 2020-05-25]. URL: http://arxiv.org/abs/cond-mat/0302536

[14] NOGA, Alon. *Splitting necklaces.* Advances in Mathematics [online]. 1987, vol. 63, no. 3, p. 247–253 [accessed 2020-05-25]. DOI: 10.1016/0001-8708(87)90055-7

[15] FILOS-RATSIKAS, Aris, GOLDBERG, Paul W. *The complexity of splitting necklaces and bisecting ham sandwiches.* Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing [online]. 2018 [accessed 2020-05-25]. URL: http://arxiv.org/abs/1805.12559v2

[16] BONSMA, P.; EPPING, Th; HOCHSTÄTTLER, Winfried. *Complexity results on restricted instances of a paint shop problem for words.* Discrete Applied Mathematics, 2006, vol. 154, no. 9, p. 1335–1343. DOI: 10.1016/j.dam.2005.05.033

[17] BONSMA, P., EPPING, Th, HOCHSTÄTTLER, Winfried. *Complexity results on restricted instances of a paint shop problem for words.* Discrete Applied Mathematics [online]. 2006, vol. 154, no. 9, p. 1335–1343 [accessed 2020-05-25]. DOI: 10.1016/j.dam.2005.05.033

[18] MEUNIER, Frédéric. *Discrete Splittings of the Necklace.* Mathematics of Operations Research. 2008, vol. 33, no. 3, p. 678–688. DOI: 10.1287/moor.1080.0311

[19] BOHMAN, Tom. *A sum packing problem of Erdös and the Conway-Guy sequence.* Proceedings of the American Mathematical Society [online]. 1996, vol. 124, no. 12, p. 3627–3636 [accessed 2020-05-25]. URL: https://www.ams.org/proc/1996-124-12/S0002-9939-96-03653-2/S0002-9939-96-03653-2.pdf

[20] ERDÖS, P. *Problems and results from additive number theory.* Colloque sur la Theorie des Nombres, Bruxelles. 1956.

[21] ERDÖS, P., TURÁN, P. *On a Problem of Sidon in Additive Number Theory, and on some Related Problems.* Journal of the London Mathematical Society. 1941, vol. 16, n. 4, p. 212–215. DOI: 10.1112/jlms/s1-16.4.212

Index