Approximation algorithms for regret minimization in vehicle routing problems

THÈSE N^{O} 6163 (2014)

PRÉSENTÉE LE 28 MAI 2014

À LA FACULTÉ DES SCIENCES DE BASE
CHAIRE D'OPTIMISATION DISCRÈTE
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Adrian Aloysius BOCK

acceptée sur proposition du jury:

Prof. M. Troyanov, président du jury
Prof. F. Eisenbrand, directeur de thèse
Prof. L. Sanità, rapporteuse
Prof. M. G. Speranza, rapporteuse
Prof. O.N.A. Svensson, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2014

Acknowledgements

First and foremost, I would like to thank Fritz Eisenbrand for guiding me through my PhD studies and for being an advisor also in personal matters. The working environment in his group at EPFL is truly inspiring. The freedom in the selection of research topics, the flexible working hours, and the possibility to work from other places than EPFL were of inestimable value to me.

Next, I am very much indebted to Laura Sanità. This work would not have been possible without her advice, collaboration, and encouragement. Further thanks go to my co-authors Karthik Chandrasekaran, Elyot Grant, Jochen Könemann, and Britta Peis. I thank my thesis committee for their helpful comments on content and presentation. Special thanks go to Alfonso Cevallos, Martin Niemeier, and Ulrike Schorr for their careful proofreading.

Un grand merci à Jocelyne, who truly is the heart of our group. Furthermore, I am especially grateful to all current and former members of DISOPT and DCG who made EPFL such an enjoyable place for work and the starting point of crazy nights and many other activities like skiing or hiking. I wish to thank my colleagues that I had the pleasure to share an office with at some point. In many fruitful discussions I learned a great deal about math, intercultural differences, food, god(s), life, and the universe. Thank you for sharing your knowledge and opinions, and for the appreciated distraction!

Finally, this thesis has begun how every mathematical essay should begin:

Let $\varepsilon > 0$.

Lausanne, May 2014

Adrian Bock

Abstract

In this thesis, we present new approximation algorithms as well as hardness of approximation results for NP-hard vehicle routing problems related to public transportation. We consider two different problem classes that also occur frequently in areas such as logistics, robotics, or distribution systems. For the first problem class, the goal is to visit as many locations in a network as possible subject to timing or cost constraints. For the second problem class, a given set of locations is to be visited using a minimum-cost set of routes under some constraints. Due to the relevance of both problem classes for public transportation, a secondary objective must be taken into account beyond a low operation cost: namely, it is crucial to design routes that optimize *customer satisfaction* in order to encourage customers to use the service. Our measure of choice is the *regret* of a customer, that is the time comparison of the chosen route with the shortest path to a destination.

From the first problem class, we investigate variants and extensions of the *Orienteering problem* that asks to find a short walk maximizing the profit obtained from visiting distinct locations. We give approximation algorithms for variants in which the walk has to respect constraints on the regret of the visited vertices. Additionally, we describe a framework to extend approximation algorithms for Orienteering problems to consider also a second budget constraint, namely node demands, that have to be satisfied in order to collect the profit. We obtain polynomial time approximation schemes for the Capacitated Orienteering problem on trees and Euclidean metrics.

Furthermore, we study variants of the *School Bus problem* (SBP). In SBP, a given set of locations is to be connected to a destination node with both low operation cost and a low maximum regret. We note that the Orienteering problem can be seen as the pricing problem for SBP and it often appears as subroutine in algorithms for SBP.

For tree-shaped networks, we describe algorithms with a small constant approximation factor and complement them by showing hardness of approximation results. We give an overview of the known results in arbitrary networks and we prove that a general variant cannot be approximated unless $P = NP$. Finally, we describe an integer programming approach to solve School Bus problems in practice and present an improved bus schedule for a private school in the lake Geneva region.

Key words:
Combinatorial Optimization, Approximation algorithm, Vehicle routing problem, Orienteering, Set-cover integer linear programming formulation

Zusammenfassung

In dieser Arbeit präsentieren wir Approximationsalgorithmen sowie Nichtapproximierbarkeitsergebnisse für NP-schwere Tourenplanungsprobleme, wie sie im öffentlichen Nahverkehr auftreten. Wir betrachten zwei verschiedene Klassen von Problemen, die auch in Bereichen wie Logistik, Robotik oder Verteilersystemen häufig vorkommen. Die erste Problemklasse hat zum Ziel, auf einer Route unter Einhaltung gewisser Zeit- oder Kostenschranken möglichst viele Orte zu besuchen. In der zweiten Problemklasse muss eine vorgegebene Menge von Orten unter gewissen Bedingungen durch Routen mit möglichst geringen Betriebskosten abgedeckt werden. Wegen der Relevanz solcher Aufgaben im Bereich des öffentlichen Nahverkehrs muss außer den Kosten meist noch eine weitere Zielfunktion berücksichtigt werden: Es ist nämlich entscheidend, auch die *Kundenzufriedenheit* zu optimieren, um die Kunden zur Nutzung des Service-Angebots zu motivieren. Unser Maß hierfür ist der *Zeitverzug* eines Kunden, der durch den Zeitunterschied zwischen der Busroute und der kürzesten Verbindung zu einem Zielpunkt bestimmt ist.

Aus der ersten Klasse untersuchen wir Varianten und Erweiterungen des *Orienteering-Problems*. Darin wird dazu aufgefordert, einen kurzen Weg mit möglichst großem Profit zu bestimmen, den man durch den Besuch verschiedener Orte bekommt. Wir entwerfen Approximationsalgorithmen für jene Varianten, in denen der Pfad Bedingungen an den Zeitverzug der besuchten Punkte einhalten muss. Darüberhinaus beschreiben wir eine allgemeine Konstruktion, wie Approximationsalgorithmen für Orienteering-Probleme um eine Budgetbedingung erweitert werden können. Diese Budgetbedingung führt eine Nachfrage an jedem Ort ein, die erfüllt werden muss, um den Profit zu erhalten. Wir erhalten Polynomialzeit-Approximationsschemata für das kapazitätsbeschränkte Orienteering-Problem, wenn das zugrundeliegende Netzwerk eine Baumstruktur ist oder die Abstandsfunktion euklidisch ist.

Außerdem befassen wir uns mit Varianten des *Schulbus-Problems* (SBP). Für dieses Problem ist eine gegebene Menge von Orten mit sowohl niedrigen Betriebskosten als auch minimalem Maximal-Zeitverzug an einen Zielpunkt anzubinden. Wir stellen Algorithmen für Netzwerke mit Baumstruktur vor, deren Approximationsfaktor kleine Konstanten sind, und komplementieren diese mit entsprechenden Nichtapproximierbarkeitsergebnissen. Wir geben einen Überblick über bekannte Ergebnisse für beliebige Netzwerke und beweisen, dass eine allgemeine Variante unter der Annahme $P \neq NP$ nicht approximiert werden kann. Schließlich beschreiben wir einen auf ganzzahliger Programmierung basierenden Ansatz zum Lösen von Schulbus-Problemen in der Praxis und präsentieren einen verbesserten Busfahrplan für eine Schule am Genfer See.

Schlagwörter:
Kombinatorische Optimierung, Approximationsalgorithmus, Tourenplanung, Orienteering, Ganzzahliges lineares Programm zur Mengenüberdeckung

Contents

List of Figures

List of Tables

1 Introduction

In this thesis, we present approximation algorithms for several NP-hard vehicle routing problems on graphs. Such problems fall into the category of combinatorial optimization problems that ask to find an optimal solution from a finite set of feasible solutions to the problem. For instance, we could wish to find a solution with minimum cost or maximum profit. Yet the set of feasible solutions may be very large, and it would be very inefficient and thus impractical to examine all solutions to find the best one. In order to solve such difficult problems, there is a need for efficient algorithms that compute a solution whose value may be suboptimal, but is still provably close to the optimum. For some of these problems, we provide hardness of approximation results showing that it is impossible to find such an approximation algorithm with a certain factor or better unless $P = NP$.

Vehicle routing is an important and active topic in computer science and operations research. In a vehicle routing problem, the objective typically is to find a minimum-cost set of routes in a network such that a given set of locations is visited and some constraints are obeyed. The constraints and cost are often related to the distance travelled, number of routes or vehicles used, coverage of the network by the routes, etc. Alternatively, the goal can be to visit a large number of locations subject to constraints, e.g. prescribed start and terminal locations. Problems of these two kinds are frequent and crucial in areas such as logistics, robotics, distribution systems, and public transportation (see, e.g., the survey by [TV01]).

A fundamental example of a vehicle routing problem is the *Traveling Salesman problem* (TSP). In the TSP, we are given a set of locations to be visited with (metric) distances between them and the task is to find the shortest tour through all locations. People are working on TSP since the 1930s [TSP]. There has been a lot of work on efficient approximation algorithms for TSP [Chr76, Aro98, Mit99, Kle05, SV12] as well as on heuristics to solve TSP close to optimality (see [TSP] for a bibliography).
TSP has always been a driving force for the development of new algorithmic techniques that were then successfully applied also to related vehicle routing problems. As a famous example, we mention the seminal work of Arora and Mitchell [Aro98, Mit99] on TSP in the plane. Recently, there was improvement made also for further special cases of TSP, namely for

planar metrics [Kle05] and for the so called graphical TSP [SV12].

We focus in this thesis on vehicle routing problems relevant to public transportation (e.g. bus systems) where all customers, situated at different locations in the network, have to be taken to a common target destination (e.g. a school). For such tasks, a secondary objective often must be taken into account beyond minimizing operation cost: namely, it is crucial to design routes that also optimize *customer satisfaction* in order to motivate customers to use the service.

Customer satisfaction is often measured in the literature by objectives that incorporate the customer's waiting time, the *latency*. For instance, the aforementioned TSP aims at minimizing the maximum latency of a customer on the tour. However, the latency is independent of the customer's home location relative to the target destination. This can lead to an unfairness perceived by the customers, since a client close to the target destination may be served much later than a client that is located further away. Therefore, we introduce the notion of regret which compares the waiting time of a customer to his shortest-path distance from the target destination. Our concept of regret addresses the described issue and allows for a client-centric point of view. We distinguish two possibilities of how the comparison of the waiting time and the shortest-path distance can be made: Namely, we refer to an *additive regret* if the difference is considered, and we call *multiplicative regret* or delay factor the ratio of the two quantities. We observe that a constraint on the additive regret is more restrictive if the customers are further away from the target destination, whereas a bound on the delay factor is more limiting if the customers are situated rather close to the depot.

We consider in the following two general categories of problems: In the first category, a solution is required to connect all nodes in the network to a destination or depot. The goal is then to minimize the operation cost while, secondary, maximizing the customer satisfaction. The second category contains problems that ask to find a single route that connects a subset of the locations to the destination. Their task is to maximize the customer satisfaction with minimal operation cost. Both frameworks share the aspect that a trade-off between two objective functions has to be found: the cost of the solution and its value for the customer satisfaction. Since the problems in both categories are typically NP-hard, we are interested in approximation algorithms for them and thus have to make a compromise for one of the two distinct objectives. Either we approximately maximize the customer satisfaction subject to an upper bound on the operation cost (*budget problem*) or we approximately minimize the operation cost subject to a lower bound on the customer satisfaction (*quota problem*).

From the first category, we will study variations of the *School Bus problem* including both budget and quota problems. The second class of problems we explore, namely *Orienteering problems*, falls into the second category. We now present these two problem classes in more detail.

Orienteering

In the Orienteering problem, a single salesman has a travel time budget and might thus not be able to visit all locations in the network. The natural goal is then to maximize the total value obtained from visiting a subset of the locations. We note that the travel time budget is equivalent to a common additive regret threshold for all nodes of the network. The name of the problem stems from an outdoor sport where the participants are given a topological map of the terrain and they wish to maximize their individual score obtained from visiting some of the control points in a limited amount of time.

We describe approximation algorithms and results for the hardness of approximation for several extensions and variations of the Orienteering Problem. In the Orienteering Problem with delay factor (OP-R), we enforce that the multiplicative regret of every visited location does not exceed a given delay factor. Arbitrary deadlines for the locations replace the time budget for the Orienteering Problem with Deadlines (OPD). We note that these deadlines can be seen as individual additive regret thresholds.

For any variation of the Orienteering Problem, we can also consider a generalization that incorporates a second budget constraint: We additionally have to satisfy demands at every location to obtain the profit. However, the total demand shall not exceed a given *capacity* bound. In particular, we study the Capacitated Orienteering Problem (COP): Maximize the total profit obtained for serving the demands of some locations without exceeding neither the travel budget nor the capacity bound.
Additionally, we discuss the quota version of the Orienteering problem, the Minimum Excess Problem (MEP), where we aim at minimizing the maximum regret that incurs by collecting a certain value from a subset of the locations.

Orienteering problems are frequently used as subroutines in algorithms for problems of the first category; we will present examples for applications of Orienteering in the School Bus problem and a related vehicle routing problem.

Following the notation commonly used in the literature, we refer to the maximum additive regret of a vertex on a path also as *excess* of the path and we call the target destination also *root*. In some cases, it is helpful to specify both endpoints for the Orienteering problem. If applicable, we state the problems and results in this more general form.

Our contribution

We briefly summarize our main theoretical results for variations of the Orienteering problem.

- A framework that allows to extend an approximation algorithm for any variant of the Orienteering problem to work also for the capacitated case at a small loss in the approximation factor.
  As a consequence, a $(3+\varepsilon)$-approximation algorithm for COP is obtained.

- Polynomial time approximation scheme (PTAS)[1] for COP on Euclidean metrics.
- PTAS for COP and MEP on tree metrics.
- $O\left(\log \frac{X_{\max}}{X_{\min}}\right)$-approximation algorithm for OPD, where $X_{\max}$ and $X_{\min}$ denote the maximum and minimum additive regret threshold, respectively.
- $O\left(\log \frac{R}{R-1}\right)$-approximation algorithm for OP-R, where $R > 1$ is a given delay factor.

School Bus problem

The School Bus Problem is an NP-hard vehicle routing problem in which the goal is to route buses that transport children to a school such that for each child, the distance travelled on the bus does not exceed the shortest distance from the child's home to the school by more than a given additive *regret* threshold. Subject to this constraint and a bus capacity limit, the goal can be to minimize the number of buses required (SBP) or to minimize the total length of the bus routes (SBP-Σ). We call SBP-R the variant where we have a fixed number of buses to use, and the goal is to minimize the maximum regret of a child.

We present in this thesis an overview of approximation and hardness results for variations of the School Bus problem and we prove relations between some of the considered variants. In addition, we study the variations considering a multiplicative regret or delay factor for approximation algorithms and hardness results. Finally, we describe the natural set-covering integer programming formulation for SBP and give bounds on the integrality gap of its linear relaxation. It is then used to obtain an approximation algorithm for a further extension of the SBP.

School Bus systems are common in countries as Switzerland, Germany or France, since the school assignment is determined by the home location of the children. The transportation demand is known in advance and thus there is big need for planning tools that can solve this complex optimization problem efficiently. We present in Chapter 4 an integer programming based approach that we used to compute an improved bus schedule for a private school in the lake Geneva area.

Note that we focus in our description on the setting of a school bus system. This is however not immanent to the considered problems; they also form the basis of many similar transportation tasks and the methods we develop can be applied to them just as well.

[1] See Basics at the of this chapter for a definition.

Our contribution

We briefly summarize our main theoretical results for variations of the School Bus problem.

- SBP:
  - $O(\log C)$-approximation algorithm, where C denotes the given bus capacity; hard to approximate with a factor better than 2 unless $P = NP$;
  - 4-approximation algorithm on tree metrics; hard to approximate with a factor less than 1.5 unless $P = NP$;
  - Integrality gap bounds for the natural covering LP formulation: upper bound of $O(\log C)$ on general metrics; upper bound of 4 and lower bound of 2 on tree metrics.
- SBP-R:
  - SBP-R cannot be approximated on general metric graphs unless $P = NP$;
  - 13.5-approximation algorithm on tree metrics in case of unlimited capacity.
- SBP-Σ:
  - 4-approximation algorithm on tree metrics; hard to approximate with a factor less than 1.5 unless $P = NP$.

We point out that these School Bus problems have not been studied for approximation algorithms before. Following our work, new results have been obtained in a recent article accepted for publication in a top-tier conference [FS14].

Organization of this thesis

This thesis has two main parts. First, we present in Chapter 2 several approximation results for variants of the Orienteering problem. This chapter is based on joint work with Laura Sanità and discussions with Alfonso Cevallos and Franka Tholen. Our work is reflected in the publications [BS13, BS14]. Second, variations of the School Bus problem are studied. In Chapter 3, we present theoretical insights for variants of the School Bus problem and we give an overview of approximation and inapproximability results. This chapter is joint work with Laura Sanità, Elyot Grant, and Jochen Könemann and it is based on the publications [BGKS11, BGKS13, BS13]. A practical application of the School Bus problem in a real-life setting is presented in Chapter 4.
Both main parts of the thesis are organized to be self-contained and do not build on each other. The introductory sections of each chapter as well as the following paragraph are advised to be consulted for notation and fundamental concepts. At several points of this thesis, we also review work by other authors. These paragraphs are clearly marked as such and we provide reference to the authors.

Basics

All optimization problems considered in this thesis share the property that every feasible solution has a non-negative objective value. We distinguish minimization and maximization problems depending on whether the objective is to minimize or to maximize a certain quantity. An algorithm for an optimization problem is said to be efficient (runs in polynomial time) if its running time can be bounded by a polynomial of constant degree in the size of the problem instance. Most of the problems encountered in this thesis are NP-hard, i.e., there exists no efficient algorithm for those problems unless the complexity classes P and NP are the same. We refer to the book of Vazirani [Vaz01] for rigorous definitions of P and NP and a more detailed introduction to complexity theory. The O-notation [Vaz01] is used for the sake of exposition and readability.

An *α-approximation algorithm* for a minimization (resp. maximization) problem is an efficient algorithm $\mathscr{A}$ that for every instance $\mathscr{I}$ of the problem returns a feasible solution with objective value at most α (resp. at least $\frac{1}{\alpha}$) times the optimal value of $\mathscr{I}$. We refer to the parameter α also as approximation factor or approximation guarantee. Note that α may depend on the input size $|\mathscr{I}|$ and that $\alpha \geq 1$ holds for any algorithm. An exact algorithm has approximation factor 1.

A *polynomial time approximation scheme* (PTAS) for a minimization (resp. maximization) problem is an algorithm $\mathscr{A}$ that, given an instance $\mathscr{I}$ of the problem and a value $\varepsilon > 0$, returns a feasible solution of value at most $(1+\varepsilon)$ $\left(\text{ resp. at least } \frac{1}{(1+\varepsilon)}\right)$ times the optimal value of $\mathscr{I}$. The running time of $\mathscr{A}$ is polynomial in the input size $|\mathscr{I}|$ for any fixed constant ε. A *fully polynomial time approximation scheme* (FPTAS) requires the running time of the algorithm to be polynomial in both the input size *and* $\frac{1}{\varepsilon}$.

We say that a problem is NP-hard to approximate within a factor of β if the existence of a β-approximation algorithm for this problem implies $P = NP$. Furthermore, a problem is inapproximable if it is NP-hard to approximate within any factor β.

Again, we refer to [Vaz01] for more details about the concepts of approximation algorithms and the hardness of approximation.

We will make use of basic concepts of graph theory [KV12] and linear programming [BT97].

2 Orienteering and related variants

2.1 Introduction

In the *Orienteering Problem* (OP), we are given an undirected complete graph $G = (V, E)$ with metric distances $d : E \to \mathbb{N}$ on the edges, start and end nodes $s, t \in V$, node profits $\pi : V \to \mathbb{N}$ and a length bound D. The goal is to find an s-t path $P = (V_P, E_P)$, subject to the length constraint $\sum_{e \in E_P} d(e) \leq D$, that maximizes the profit $\sum_{v \in V_P} \pi(v)$ collected from the visited nodes.

OP is a fundamental network design problem with high theoretical and practical relevance. The problem is widely studied in the literature for approximation, exact, and heuristic algorithms ([CKP12], [CH08], [FDG05], [VSO11]). There is a long list of vehicle routing problems for which efficient algorithms rely on applying algorithms for OP as a subroutine (e.g. Deadline-TSP [BBCM04], Distance-constrained Vehicle Routing Problem [NR12], or the School Bus Problem, cf. Chapter 3).

Many approximation algorithms for OP rely on results for the closely related Q-path problem: Given an undirected complete graph $G = (V, E)$ with metric distances $d : E \to \mathbb{N}$, start and end nodes $s, t \in V$, node profits $\pi : V \to \mathbb{N}$ and a quota Q, the goal is to find the shortest s-t path $P = (V_P, E_P)$ satisfying the quota constraint $\sum_{v \in V_P} \pi(v) \geq Q$. This problem can be strengthened to the *Minimum Excess Problem* (MEP). MEP asks in the same setting as Q-path for an s-t-path P of minimum excess, i.e., minimizing the length of the path minus the shortest path distance between s and t (the regret of t). We observe that MEP and the Q-path problem are equivalent in terms of the optimal solution, however, an α-approximation algorithm for MEP implies an α-approximation algorithm for the Q-path problem, but not vice versa.

In section 2.2, we focus on a natural generalization of OP in which the selected path has to satisfy a second budget constraint besides the length bound. Given node demands $r : V \to \mathbb{N}$ and a capacity bound $C \in \mathbb{N}$, we look for a path P that maximizes $\sum_{v \in V_P} \pi(v)$ and that satisfies both constraints $\sum_{e \in E_P} d(e) \leq D$ and $\sum_{v \in V_P} r(v) \leq C$. We refer to this problem as the *Capacitated*

Orienteering Problem (COP).

Furthermore, we consider the *Orienteering Problem with deadlines* (OPD). For OPD, we are given undirected complete graph $G = (V, E)$ with metric distances $d : E \to \mathbb{N}$ on the edges, a start node $s \in V$ and node profits $\pi : V \to \mathbb{N}$ as for OP, but also node deadlines $D : V \to \mathbb{N}$. Instead of a length bounded path, OPD asks for a rooted path that collects maximum profit from vertices visited before their deadlines. Observe that the special case where all deadlines are the same corresponds to OP where only the start node is specified.

OPD itself is a special case of the *Orienteering Problem with time windows* (OP-TW), where in addition integer release times are given for every vertex. The goal of OP-TW is to find a rooted path that collects maximum profit from vertices visited in the specified time window.

2.1.1 Related work

In the following, we will report results for Orienteering problems that were mostly given for unit profit values. However, they can be generalized to arbitrary profits by standard scaling and rounding techniques [BCK+03] (cf. also Section 2.3) involving an additional $(1 + o(1))$-factor in the approximation guarantee. We denote by OPT the number of vertices of the optimal solution.

OP was introduced by Golden et al. [GLV87] and is NP-hard already on a star (cf. [ABST14] or Section 2.5). Approximation algorithms for OP were designed by Arkin et al. [AMN98] primarily for given points in the plane. The first constant factor approximation algorithm for general metrics was obtained by Blum et al. [BCK+03] and subsequently improved to a $(2+\varepsilon)$-approximation algorithm [BBCM04, CKP12]. OP is NP-hard to approximate within $\frac{1069}{1068}$ combining results in [BCK+03] and [KS12]. In the special case of Euclidean metrics, OP admits a PTAS [CH08], but is NP-hard even for unit profits $\pi \equiv 1$ [Pap77]. OP on tree metrics admits an FPTAS based on dynamic programming [ABST14].

Gupta, Krishnaswamy, Nagarajan and Ravi [GKNR12] give approximation algorithms for a stochastic version of OP. Among their results, they obtain a constant factor approximation algorithm for COP (which they call Knapsack Orienteering). In particular, they design a 2α-approximation algorithm for COP, relying on an α-approximation algorithm for OP as a subroutine. Using the currently best known approximation algorithm for OP given by Chekuri et al. [CKP12] with factor $(2+\varepsilon)$, their result translates into a $(4+\varepsilon)$-approximation algorithm for COP.

The Q-path problem and MEP are usually considered in the unit profit case [CGRT03, BCK+03] and the Q-path problem is then referred to as k-path or k-stroll problem [Mit99, CGRT03]. Both problems are NP-hard even on star metrics (cf. Section 2.5) and the best known approximation factor in general graphs is $(2+\varepsilon)$ for both problems [BCK+03, CGRT03]. As for OP, both are NP-hard for Euclidean metrics (also with unit profits) and there is a PTAS known [Aro98, Mit99, CH08]. Angelelli et al. present an FPTAS for Q-path on tree metrics [ABST14].

Bansal et al. [BBCM04] introduce OPD and give a $O(\log OPT)$-approximation algorithm that relies on an approximation algorithm for OP as a subroutine. They also obtain a $O(\log^2 OPT)$-approximation algorithm for OP-TW and a $O(\log D_{\max})$-approximation using bicriteria optimization, where $D_{\max} := \max_{v \in V} D(v)$ denotes the largest deadline in the instance. These results can be improved if the number of different time windows is constant [CK04] or if the length of all time windows is the same and if additionally the endpoints are not specified, but arbitrary [FW07]. In both cases, a constant factor approximation algorithm is obtained. Furthermore, there is a $O\left(\alpha \cdot \max\left\{\log OPT, \log \frac{L_{\max}}{L_{\min}}\right\}\right)$-approximation relying on an α-approximation for OP [CKP12], where $L_{\max}$ and $L_{\min}$ denote the length of the largest and smallest time window, respectively. We note that OP-TW is NP-hard already on a line [Tsi92] and a $O(\log OPT)$-approximation algorithm for OP-TW that runs in quasi-polynomial time can be obtained [CP05].

The time-dependent OP was studied by Fomin and Lingas [FL01]. Broden et al. [BHN04] give a 4/3-approximation algorithm for the special case of unit profits and distances 1 or 2.

A survey on routing problems with profits is due to Feillet, Dejax, Gendreau [FDG05]. Recently, Vansteenwegen, Souffriau and Van Oudheusden [VSO11] gave a survey on heuristics for Orienteering problems. Butt and Cavalier [BC94] introduced the Multiple Tour Maximum Collection Problem which is the natural extension to a fleet of vehicles. This problem was first referred to as Team Orienteering Problem by Chao, Golden and Wasil [CGW96]. Recent heuristics for the Team Orienteering Problem were presented by Tang, Miller-Hooks [TMH05] and by Archetti, Hertz and Speranza [AHS07], whereas an exact algorithm was presented by Boussier, Feillet, Gendreau [BFG07].
The Capacitated Team Orienteering Problem (CTOP) was first studied by Archetti et al. [AFHS09] where heuristics and a branch-and-price algorithm were proposed. The algorithms were extended to work also for CTOP if a node demand can be served by several vehicles (split deliveries) by Archetti et al. [ABSH14]. They also showed that if split deliveries are allowed, the profit of the optimal solution can be at most twice as large as the optimal profit for CTOP.

Also OP-TW has been studied extensively for heuristics [Sav85, DLSS88, KR92, DDS92, Tha95, TLZO01].

2.1.2 Our contribution

We present a framework that allows to additionally consider node demands and a capacity bound in any variation of the Orienteering problem. If an α-approximation algorithm is known for the variant with unlimited capacity, we obtain a $(\alpha+1+\varepsilon)$-approximation algorithm respecting the capacity bound. Most interestingly, this implies, using the result in [CKP12], an approximation guarantee of $(3+\varepsilon)$ for COP, improving upon [GKNR12].

Furthermore, we show that there is a Polynomial Time Approximation Scheme (PTAS) for COP on tree metrics and on Euclidean metrics. Both results rely on dynamic programming

combined with a feasibilization method introduced by Grandoni and Zenklusen [GRSZ13]. The PTAS for COP on Euclidean metrics builds upon the corresponding known result for OP by Chen and Har-Peled [CH08].

We also give a PTAS for Q-path on tree metrics and extend this result to MEP.

OP on planar graphs is a topic of ongoing research; we describe the first steps taken and give an outline on how a PTAS could potentially be obtained.

For the OPD, we obtain a $O\left(\log \frac{X_{\max}}{X_{\min}}\right)$-approximation algorithm, where $X_{\max}$ and $X_{\min}$ denote the maximum and minimum individual regret threshold, i.e., the difference of the deadline and the shortest distance to the root. We also consider the special case where all deadlines are R times the shortest distance to the root, where R is a delay factor given as part of the input. In this case, we obtain a $O\left(\log \frac{R}{R-1}\right)$-approximation algorithm. We note that our result for OPD also extends to the case in which both endpoints of the path are specified.

Additionally, we observe that for some capacitated vehicle routing problems, good approximation algorithms can be built upon the existence of good approximation algorithms for COP.

A first extension considers a fleet of M capacitated vehicles that can be used to collect profits. This is the so-called Capacitated Team Orienteering Problem (CTOP), that is motivated by several applications in logistics [AFHS09]. As an example, consider a carrier that has to select the most profitable customers whose demands can be served given his capacitated fleet of vehicles.

Important savings can be achieved if a customer can be served by more than one vehicle (split delivery). This variant of the problem is called the Capacitated Team Orienteering with split deliveries (SDCTOP) [ABSH14]. Both CTOP and SDCTOP have been studied in terms of heuristics and exact algorithms, but there is no rigorous analysis of the complexity of these problems in terms of approximation algorithms. We give a constant factor approximation algorithm for both problems, using our approximation results for COP.

A second application concerns the Distance-constrained Vehicle Routing Problem (DVRP) that we will explore in Section 3.4.2. This problem asks for the minimum number of paths respecting both a length bound D and a capacity bound C that is necessary to serve all customer demands. Nagarajan and Ravi [NR12] described a $\min\{O(\log n), O(\log D)\}$-approximation algorithm, where n denotes the number of customers. Recently, an $O\left(\frac{\log D}{\log\log D}\right)$-approximation algorithm was presented by Friggstad and Swamy [FS14]. Combining our results with the greedy algorithm for the set cover problem, we easily obtain an $O(\log C)$ approximation.

2.1.3 Preliminaries

In the following, we use the notation $r(W) := \sum_{v \in W} r(v)$ and analogously $\pi(W) := \sum_{v \in W} \pi(v)$ for any subset of nodes W. For a given path P, we denote by V_P the nodes of P and by E_P its edges. With a slight abuse of notation, we shorten $r(V_P)$ and $\pi(V_P)$ to $r(P)$ and $\pi(P)$ respectively. Furthermore, we will use $d_G(u,v)$ to refer to the shortest path distance between two vertices u and v in G. If G is clear from the context, the subscript is omitted. For a path P, we define $d^P(u,v)$ to be the distance along the path between two vertices $u, v \in V_P$. The segment of the path P (or subpath) between u and v is denoted by $P(u,v)$. Furthermore, we denote $\ell_v := d(s,v)$ for the distance of a vertex $v \in V$ from the root s.

Given a complete graph $G = (V,E)$ with metric edge distances d, we say that d is a *tree metric* if there exists a spanning tree $T = (V,F)$ of G with the following property: $d(e) = d_T(u,v)$ for all $e = \{u,v\} \in E$ where $d_T(u,v)$ is the length of the unique u-v path in T. Similarly, we say d is induced by a planar graph, if there exists a planar subgraph $\tilde{G} = (V,\tilde{E})$ such that $d(e) = d_{\tilde{G}}(u,v)$ for all $e = \{u,v\} \in E$. If the tree metric is induced by a star, then we also refer to it as a star metric.

When dealing with metrics that are induced by a tree or a planar graph, it is useful to think about COP in a slightly different but equivalent way.

Given a tree $T = (V,F)$ with node profits and demands, edge weights $d : F \to \mathbb{N}$ and two nodes s and t, COP asks for a *walk* P between s and t in T that collects maximum profit from (a subset of) the nodes contained in P and satisfies both a length bound D and the capacity bound C. It is straightforward to see that this is equivalent to asking for a *path* between s and t in the metric closure of (T,d) (cf. [KV12]), i.e., the complete graph $(G,\tilde{d})$ that has an edge $e = \{u,v\}$ with $\tilde{d}(e) = d(u,v)$ for every $u, v \in V$.

In the same way, we can consider a metric induced by a planar graph.

The following proposition is an easy observation that is crucial to most of the approximation algorithms for Orienteering problems. Recall that the regret of a vertex u on a path P from v to w is the difference $d^P(v,u) - d(v,u)$. Further, we say that a path P from v to w has excess $d^P(v,w) - d(v,w)$. Note that the excess of a path is the maximum regret among the vertices on the path or equivalently, the regret of the last vertex. This is because the regret is non-decreasing when traversing a path from one endpoint (cf. 3.1.3 for a formal argument). The following simple proposition shows that a rooted path can be split into k rooted paths such that the profit is evenly distributed and one of them has low excess. A similar statement was implicitly used in [BBCM04].

Proposition 2.1.1. *Given an integer $k \geq 1$ and a rooted path P with profit Π and excess X, there is a rooted path of profit $\geq \frac{\Pi}{k}$ and excess at most $\frac{X}{k}$.*

Proof. We define $k+1$ breakpoints v_i $(i = 0,\ldots,k)$ that split the path P into k segments of

profit $\geq \frac{\Pi}{k}$. This can be achieved by following P starting from $v_0 = s$ until a share of profit $\geq \frac{\Pi}{k}$ is visited. Note that the breakpoints are not necessarily distinct. The k new rooted paths P_i are obtained by connecting each of the segments to the root. Every path has obviously profit $\geq \frac{\Pi}{k}$ and it remains to prove that there exists a path with small excess. We show that the sum of the excess of the paths P_i is exactly X.
Let $X_i = d(s, v_i) + d^P(v_i, v_{i+1}) - d(s, v_{i+1})$ denote the excess of path P_i for $i = 0, \ldots, k-1$. This yields

$$\sum_{i=0}^{k-1} X_i = \sum_{i=0}^{k-1} d(s, v_i) + d^P(v_i, v_{i+1}) - d(s, v_{i+1}) = d^P(s, v_k) - d(s, v_k) = X$$

and thus the statement follows. □

In Section 2.4 and 2.5, we present polynomial time approximation schemes for COP and MEP in special metrics. For the sake of readability, we only show that the described algorithms obtain a value at least $(1-\varepsilon)$ times the optimum for any $\varepsilon > 0$ and omit the additional notation proving that this also implies an $(1+\tilde{\varepsilon})$-approximation algorithm for every $\tilde{\varepsilon} := \frac{\varepsilon}{1-\varepsilon} > 0$.

2.2 Capacitated Orienteering on general graphs

In this section, we describe a framework to extend approximation algorithm for variants of Orienteering to respect also an additional budget constraint for given node demands. For simplicity of exposition, we focus on the relation of OP and COP.

2.2.1 A $(3+\varepsilon)$-approximation algorithm for COP

We present a $(3+\varepsilon)$-approximation algorithm for COP. This result follows from Theorem 2.2.1 and the $(2+\varepsilon)$-approximation algorithm for OP [CKP12]. We observe that Theorem 2.2.1 holds in the same way also for any other variation of the Orienteering problem, since the α-approximation algorithm for OP is used as a subroutine in a black box fashion.

Theorem 2.2.1. *Given an α-approximation algorithm for OP and $1 > \varepsilon > 0$, there is a $(1+\alpha+\varepsilon)$-approximation algorithm for COP.*

Proof. Given an instance (G, d, π, r, C, D) of COP, we let P^* be an optimal solution. We first set $\delta := \frac{\varepsilon}{3}$, and $\hat{\delta} := \frac{\delta}{1+\delta}$. Clearly, both δ and $\hat{\delta}$ are fixed small constants. Consequently, we observe that if $|V_{P^*}| \leq \frac{1}{\hat{\delta}}$, then an optimal solution can be found by enumeration. Therefore, we can assume $|V_{P^*}| > \frac{1}{\hat{\delta}}$.

As a first step, we guess the $\frac{1}{\hat{\delta}}$-cardinality subset S of nodes of highest demand values visited by P^* (this can be done in polynomial time by trying out all possibilities). We have $S \subset V_{P^*}$ with $|S| = \frac{1}{\hat{\delta}}$ and the following property: for any $u \in S$ and any $v \in V_{P^*} \setminus S$, $r(v) \leq r(u)$.

Based on this guessing step, we are going to slightly modify our instance. First, we eliminate all the nodes $v \notin S$ with $r(v) > \hat{\delta}C$ from our instance, as such nodes are surely not contained in V_{P^*}: if so, then $r(u) > \hat{\delta}C$ for all $u \in S$ would hold by our choice of S, leading to a violation of the capacity bound C. Second, we order all the nodes according to their profit and demand ratio, i.e. $\frac{\pi(1)}{r(1)} \geq \cdots \geq \frac{\pi(n)}{r(n)}$, where n is the total number of nodes left in our instance. By scaling, we can also assume $\frac{\pi(n)}{r(n)} \geq 1$. Finally, we let $k := \left\lfloor \log_{(1+\delta)} \frac{\pi(1)}{r(1)} \right\rfloor$. Consider Algorithm 1.

Algorithm 1 $(1+\alpha+\varepsilon)$-approximation algorithm for COP on modified instances

1: $P_{\text{uncap}} \leftarrow \emptyset$, $P_{\text{cap}} \leftarrow \emptyset$ and $i \leftarrow \left\lfloor \log_{(1+\delta)} \frac{\pi(1)}{r(1)} \right\rfloor + 1$.
2: **while** $P_{\text{cap}} = \emptyset$ and $i > 0$ **do**
3: $\quad i \leftarrow i - 1$.
4: $\quad$ For all $v \in V$, define new profit values $\pi^{(i)}(v) := \begin{cases} \pi(v) & \text{if } \frac{\pi(v)}{r(v)} \geq (1+\delta)^i \\ 0 & \text{otherwise} \end{cases}$
5: $\quad$ Apply an α-approximation algorithm to the OP instance $(G, d, \pi^{(i)}, r, D)$, that returns a path P.
6: $\quad$ **if** $r(P) \leq C$ **then**
7: $\quad\quad P_{\text{uncap}} \leftarrow P$
8: $\quad$ **else**
9: $\quad\quad P_{\text{cap}} \leftarrow P$
10: $\quad$ **end if**
11: **end while**
12: **if** $P_{\text{cap}} \neq \emptyset$ **then**
13: $\quad$ Remove greedily the node v in P_{cap} of smallest demand $r(v)$ (and shortcut the resulting path accordingly), until $r(P_{\text{cap}}) \leq C$.
14: **end if**
15: **Return** $\operatorname{argmax}\left\{\pi(P_{\text{uncap}}), \pi(P_{\text{cap}})\right\}$

The algorithm does a number of (at most $k+1$) iterations. In iteration i, it considers only the subset of nodes of the graph whose profit per demand ratio is at least $(1+\delta)^i$, and computes a path using an approximation algorithm for OP, therefore ignoring the capacity bound (step 5). If the resulting path P respects the capacity bound C, then this path will be stored as the last feasible path (P_{uncap}) found using the approximation algorithm for OP.

In each iteration, the algorithm enlarges the set of nodes to be considered until eventually $r(P) > C$ holds for the path returned by the approximation algorithm for OP. At this point, the algorithm stores this path (P_{cap}) and exits the `WHILE` loop.

By the end of the `WHILE` loop execution, the algorithm has stored two paths: P_{uncap} which is a feasible path for our COP instance, and P_{cap} which is in contrast not feasible because the capacity bound is not respected. For this reason, the algorithm modifies P_{cap} in step 13

by greedily discarding the nodes of smallest demand until it becomes a feasible path. The algorithm finally outputs the more profitable path among P_{uncap} and P_{cap}.

We claim that Algorithm 1 gives a $(\alpha + 1 + \varepsilon)$-approximation for COP.

Before going into the details of our analysis, note that we do not require P_{uncap} and P_{cap} to contain the set S. The reason for guessing S is to ensure that $r(P_{\text{cap}}) \geq (1-\hat{\delta})C$ holds *after* step 13: this is because in step 13 of the algorithm we will never discard nodes in S from P_{cap}, since those will have a high demand value but $r(S) \leq C$ holds. Consequently, we only discard nodes v with demand $r(v) \leq \hat{\delta}C$ and therefore $r(P_{\text{cap}}) \geq (1-\hat{\delta})C$: this inequality will be used later in our analysis.

Let i^* be the value of the variable i at the end of the algorithm. If $i^* = 0$, then the path P_{uncap} is an α-approximation for COP, since it respects the capacity bound and since the profits $\pi^{(0)} \equiv \pi$ are considered. Thus we can focus on the case $i^* > 0$, i.e., the `WHILE` loop was exited because the variable P_{cap} was set.

We define

$$V^+(i^*) := \left\{ v \in V_{P^*} : \frac{\pi(v)}{r(v)} \geq (1+\delta)^{i^*+1} \right\}$$
$$V^-(i^*) := \left\{ v \in V_{P^*} : \frac{\pi(v)}{r(v)} < (1+\delta)^{i^*+1} \right\}$$

Trivially, $\pi(P^*) = \pi(V^+(i^*)) + \pi(V^-(i^*))$. We are now going to give a bound on both $\pi(V^+(i^*))$ and $\pi(V^-(i^*))$.

Our first claim is that

$$\pi(V^+(i^*)) \leq \alpha\pi(P_{\text{uncap}}).$$

This follows by the approximation guarantee of the algorithm used for the OP instance at step 5. More precisely, note P_{uncap} has been found in iteration i^*+1, and that the optimal solution P^* is feasible for the OP instance $(G, d, \pi^{(i^*+1)}, r, D)$ considered in that iteration. Therefore,

$$\pi(P_{\text{uncap}}) \geq \frac{1}{\alpha}\pi^{(i^*+1)}(P^*) = \frac{1}{\alpha}\pi(V^+(i^*)).$$

Our second claim is that

$$\pi(V^-(i^*)) \leq (1+\delta)^2\pi(P_{\text{cap}}).$$

To see this, first observe that $\pi(V^-(i^*)) \leq \sum_{v \in V^-(i^*)} \frac{\pi(v)}{r(v)} \cdot r(v) \leq C \cdot (1+\delta)^{i^*+1}$ by definition and feasibility of P^*. Then, observe that $\pi(P_{\text{cap}}) = \sum_{v \in V_{P_{\text{cap}}}} r(v) \cdot \frac{\pi(v)}{r(v)} \geq (1-\hat{\delta})C(1+\delta)^{i^*}$, because the

path P_{cap} only contains nodes with a profit per demand ratio of at least $(1+\delta)^{i^*}$ and because its total demand is $r(P_{\text{cap}}) \geq (1-\hat{\delta})C$. Therefore,

$$\pi(V^-(i^*)) \leq C \cdot (1+\delta)^{i^*+1} \leq \frac{1}{1-\hat{\delta}}(1+\delta)\pi(P_{\text{cap}}) = (1+\delta)^2\pi(P_{\text{cap}}).$$

Combining the two claims, we now conclude that

$$\begin{aligned}\pi(P^*) &= \pi(V^+(i^*)) + \pi(V^-(i^*)) \\ &\leq \alpha \cdot \pi(P_{\text{uncap}}) + (1+\delta)^2 \cdot \pi(P_{\text{cap}}) \\ &\leq (1+\varepsilon+\alpha)\max\left\{\pi(P_{\text{uncap}}), \pi(P_{\text{cap}})\right\}\end{aligned}$$

□

We remark that in our setting the profit of a node is only collected if the corresponding demand is fulfilled (no partial serving is possible). However, the results above can be easily extended to also allow partially served nodes.

2.2.2 Applications of an approximation algorithm for COP

In this section, we present an application of our approximation algorithm for COP to a fleet of vehicles. The Capacitated Team Orienteering problem (CTOP) generalizes the COP and allows to use M paths respecting both the capacity and the length bound.

Observe that the Capacitated Team Orienteering Problem can be seen as an instance of the *maximum k-coverage problem* that is defined as follows (see [Hoc82]): For a universe and set system on its elements, choose k sets whose union contains as many elements as possible.

With this observation, the following theorem applies to COP what was proved in [HP98] for maximum k-coverage. This was also noted in [BCK+03] for OP. However, we state its short proof for completeness.

Theorem 2.2.2. *There is a* 3.53*-approximation for CTOP.*

Proof. We show that given an α-approximation algorithm for COP, there is a $\left(1-e^{-1/\alpha}\right)^{-1}$-approximation algorithm for CTOP. This is obtained by applying the greedy algorithm for maximum k-coverage to our setting. The theorem then follows from $\alpha := 3+\varepsilon$.

- `For` $i = 1$ `to` M `do`
- Run the α-approximation algorithm for COP to obtain path P_i.
- Remove all covered nodes from the instance.

- return $P_1, \ldots, P_M$

We follow the same steps as for maximum k-coverage to prove the claimed approximation guarantee [HP98].
Let OPT denote the profit collected by the optimal solution and $\pi^{(i)}_{\text{left}}$ denote the difference between OPT and the amount of profit collected until iteration i.
Note that in every iteration i, $\pi^{(i-1)}_{\text{left}}$ can be collected using M paths, since the optimal solution does it. Thus the profit of the path that the α-approximation algorithm for COP returns in the following iteration is $\pi^{(i)}_{\text{newly covered}} \geq \frac{\pi^{(i-1)}_{\text{left}}}{\alpha M}$.

By induction on i, we have

$$\begin{aligned}\pi^{(i)}_{\text{left}} &\leq \pi^{(i-1)}_{\text{left}} - \pi^{(i)}_{\text{newly covered}} \\ &\leq \pi^{(i-1)}_{\text{left}} \left(1 - \frac{1}{\alpha M}\right) \leq \left(1 - \frac{1}{\alpha M}\right)^i OPT\end{aligned}$$

We obtain

$$\pi\left(\bigcup_{i=1}^{M} P_i\right) \geq OPT - \pi^{(M)}_{\text{left}} \geq \left(1 - \left(1 - \frac{1}{\alpha M}\right)^M\right) OPT \geq \left(1 - e^{-1/\alpha}\right) OPT$$

□

If we now allow node demands to be served by several paths, we can apply the result of Archetti et al. [ABSH14] to obtain an approximation algorithm for SDCTOP. Namely, they proved that an α-approximation algorithm for CTOP can be turned into a 2α-approximation algorithm for SDCTOP. Combining this observation with our results, we have the following corollary.

Corollary 2.2.3. *There is a 7.06-approximation algorithm for SDCTOP.*

2.3 Scaling and Rounding techniques

Our results for tree metrics and Euclidean metrics developed in the next sections rely on dynamic programming. However, the running time will depend on the total profit of the nodes $\pi(V)$ and the capacity bound C for COP. For Q-path or MEP, the running time will depend on the sum of the edge lengths. Since these values are in general not polynomial in the input size, the running time of the dynamic program would not be polynomial. We show in this section how these obstacles can be overcome.

The first lemma shows that by preprocessing our instance using standard scaling and rounding techniques (see Knapsack problem in [Vaz01]), we can focus on the case where the profits are polynomially bounded in the input size at a very small loss. In the following, we denote the size of a given instance $\mathcal{I} = (G, d, \pi, r, C, D)$ of COP by $|\mathcal{I}|$.

Lemma 2.3.1. *Suppose having an algorithm $\mathscr{A}$ that returns for every $\delta > 0$ a feasible solution to a given instance $\mathscr{I} = (G, d, \pi, r, C, D)$ of COP whose value is at least $(1-\delta)$ times the optimal value and that runs in time polynomial in $(|\mathscr{I}| \cdot \pi(V))$ for δ fixed. Then there is a PTAS for COP. Furthermore, if $\mathscr{A}$ runs in time polynomial in $(|\mathscr{I}| \cdot \pi(V))$ and $\frac{1}{\delta}$, then there is an FPTAS for COP.*

Proof. In order to show a PTAS for COP, we have to design an algorithm that finds a feasible path P with $\pi(P) \geq (1-\varepsilon)\pi(P^*)$ for a given instance (G, d, π, r, C, D) of COP with optimal solution P^* and any fixed $\varepsilon > 0$ and that runs in time polynomial in $|\mathscr{I}|$. We perform the following steps.

Let $\pi_{\max} := \max_{v \in V} \pi(v)$ and $K := \dfrac{\varepsilon \pi_{\max}}{2n}$ for $n = |V|$. Observe that every vertex that does not lie on a feasible s-t-path can be ignored. Therefore, we can rely on the fact that $\pi(P^*) \geq \pi_{\max}$ holds.

For each $v \in V$, we define $\tilde{\pi}(v) := \left\lfloor \frac{\pi(v)}{K} \right\rfloor$. Now consider the instance $(G, d, \tilde{\pi}, r, C, D)$ and let $\tilde{P}^*$ denote its optimal solution. Apply algorithm $\mathscr{A}$ to $(G, d, \tilde{\pi}, r, C, D)$ with $\delta := \varepsilon/2$, and let P be the path output by $\mathscr{A}$. Note that we can assume $\pi(P^*) \geq \pi_{\max}$ without loss of generality. Using $\pi(P^*) \geq \pi_{\max}$, we get

$$\begin{aligned}
\pi(P) &\geq K\tilde{\pi}(P) \geq (1-\delta)K\tilde{\pi}(\tilde{P}^*) \geq (1-\delta)K\tilde{\pi}(P^*) = (1-\delta)K \sum_{v \in V_{P^*}} \tilde{\pi}(v) \\
&\geq (1-\delta) \sum_{v \in V_{P^*}} (\pi(v) - K) \geq (1-\delta)(\pi(P^*) - Kn) \\
&\geq (1-\delta)\left(1 - \frac{\varepsilon}{2}\right)\pi(P^*) \geq (1-\varepsilon)\pi(P^*).
\end{aligned}$$

Note that $\tilde{\pi}(V) = O\left(\frac{n^2}{\varepsilon}\right)$. Therefore algorithm $\mathscr{A}$ applied on our modified instance will have a running time polynomial in $|\mathscr{I}|$, yielding a PTAS for COP. Observe that if the running time of algorithm $\mathscr{A}$ depends only polynomially on $\frac{1}{\delta}$, then we obtain an FPTAS for COP.

□

Following the lines of the previous proof, we also show that we can similarly scale and round the distances on the edges of the graph.

Lemma 2.3.2. *Suppose having an algorithm $\mathscr{A}$ that finds the optimal solution to a given instance $\mathscr{I} = (G, d, \pi, Q)$ of Q-path in time polynomial in $(|\mathscr{I}| \cdot L)$ where $L = \sum_{e \in E} d(e)$. Then there is a FPTAS for Q-path.*

Proof. In order to show a FPTAS for Q-path, we have to design an algorithm that finds a feasible path P with $\pi(P) \geq Q$ and $d(P) \leq (1+\varepsilon)d(P^*)$ for a given instance (G, d, π, Q) of Q-path with optimal solution P^* and any fixed $\varepsilon > 0$ and that runs in time polynomial in $|\mathscr{I}|$.

We will modify the distances d as follows. We first guess the longest edge in the optimal solution P^*. In other words, we enumerate all $|E|$ possibilities for such a choice. Let $d_{\max}$ denote the length of this edge. By assumption, we can remove all edges of length $> d_{\max}$ from the graph (and thus all components of the graph that become disconnected from s), since the optimal solution does not consider them either.
Set $K := \frac{\varepsilon d_{\max}}{n}$. For each $e \in E$, we then define $d'(e) := \left\lfloor \frac{d(e)}{K} \right\rfloor$. Apply algorithm $\mathscr{A}$ to (G, d', π, Q) and let P' be the solution output by $\mathscr{A}$. We obtain

$$d(P') \leq Kd'(P') + K|P'| \leq Kd'(P') + Kn \leq Kd'(P^*) + \varepsilon d_{\max} \leq (1+\varepsilon)d(P^*).$$

Observe that $L' = \sum_{e \in E} d'(e) = O\left(\frac{n^2}{\varepsilon}\right)$ and thus algorithm $\mathscr{A}$ runs in time polynomial in $|\mathscr{I}|$ and $\frac{1}{\varepsilon}$. □

We now show how to proceed if the capacity bound C is not polynomially bounded in the input size. In contrast to the profit values or distances, we cannot apply standard scaling and rounding techniques directly, since even a small loss of capacity may have a big impact on the profit value of the solution. To overcome this difficulty, we rely on a *feasibilization* method introduced in [GRSZ13]. This approach consists in guessing a constant number of vertices of highest profit in the optimal solution as a preprocessing step. Like that, we can ensure that the vertices outside the guessed set of vertices have a small profit in comparison to the optimal value. Then a scaling and rounding of the demands can be shown to yield a PTAS.

Lemma 2.3.3. *Suppose having an algorithm $\mathscr{A}$ that returns for every $\delta > 0$ a feasible solution to a given instance $\mathscr{I} = (G, d, \pi, r, C, D)$ of COP whose value is at least $(1-\delta)$ times the optimal value and that runs in time polynomial in $(|\mathscr{I}| \cdot C)$ for δ fixed. Then there is a PTAS for COP.*

Proof. We want to design an algorithm that finds a feasible path P with $\pi(P) \geq (1-\varepsilon)\pi(P^*)$ for a given instance (G, d, π, r, C, D) of COP with optimal solution P^* and a fixed $\varepsilon > 0$ and that runs in time polynomial in $|\mathscr{I}|$. We perform the following steps.

First, we set $\delta := \varepsilon/2$, and we guess the $\frac{1}{\delta}$ nodes visited by the optimal solution with the largest profit. Note that we can simply enumerate all possibilities in time $O\left(n^{\frac{2}{\varepsilon}}\right)$. Let $S \subset V_{P^*}$ be the guessed set of nodes. Knowing S, we will define a modified COP instance (G, d, π', r', C', D). The capacity bound and the demands are scaled and rounded as follows. Let $\tilde{C} := C - \sum_{v \in S} r(v)$ and $c_{\max} := \max_{v \notin S : r(v) \leq \tilde{C}} r(v)$. For $K := \frac{\delta}{n} c_{\max}$, we set $C' := \left\lfloor (1-\delta) \frac{\min\{\tilde{C}, nc_{\max}\}}{K} \right\rfloor$. Moreover, we define new demand values for the nodes by setting $r'(v) := 0$ for $v \in S$ and $r'(v) := \left\lfloor \frac{r(v)}{K} \right\rfloor$ otherwise. Note that this scaling and rounding operation yields a modified capacity bound $C' = O(n^2/\delta)$.

In addition, we define new profit values π' as follows:

$$\pi'(v) := \begin{cases} 0 & \text{for all } v \notin S \text{ such that } \pi(v) > \min_{u \in S} \pi(u) \\ 0 & \text{for all } v \notin S \text{ such that } r(v) > c_{\max} \\ \pi(v) & \text{otherwise} \end{cases}$$

In words, we set to zero the profits of the nodes that are surely not contained in the optimal solution P^*. Observe that $\pi'(P^*) = \pi(P^*)$ by construction.

Consider the instance (G, d, π', r', C', D). To this new instance (G, d, π', r', C', D), we apply algorithm $\mathscr{A}$ with parameter δ to obtain a path P. Without loss of generality, we assume that P does not contain any node v with $\pi'(v) = 0$ (if so, we could remove such a node from P and shortcut consistently, without affecting value $\pi'(P)$).

We are now going to show that P is feasible for the original instance (G, d, π, r, C, D). The bound on the length is clearly satisfied for P. It remains to show the feasibility with respect to the capacity constraint.

We have:

$$\begin{aligned} r(P) \leq r(S) + r(P \setminus S) &\leq r(S) + Kr'(P \setminus S) + Kn \\ &\leq r(S) + KC' + \delta c_{max} \\ &\leq r(S) + (1-\delta)(C - r(S)) + \delta c_{max} \leq C. \end{aligned}$$

Finally, we show that P collects profit $\pi(P) \geq (1-\varepsilon)\pi(P^*)$. Our analysis follows the steps in [GRSZ13].

First, we describe how the optimal path P^* of the original instance can be modified to a path P' that is feasible for the modified instance (G, d, π', r', C', D) by dropping some selected nodes of P^* such that $\pi(P') \geq (1-\delta)\pi(P^*)$ holds. This follows the ideas of the greedy algorithm for the Knapsack problem [KV12].
Consider the following process: We remove from P^* the nodes in $P^* \setminus S$ with smallest ratio $\pi'(v)/r'(v)$ until we get a path P' that satisfies $r'(P') \leq C'$. Let $W = P^* \setminus P'$ denote the set of removed nodes and let $w^* \in W$ be the last removed vertex.
Observe that $\dfrac{\pi'(P^* \setminus S)}{r'(P^* \setminus S)} \geq \dfrac{\pi'(W \setminus w^*)}{r'(W \setminus w^*)}$ since we remove vertices with the smallest ratio first.
It follows that $\pi'(W \setminus w^*) \leq \dfrac{r'(W \setminus w^*)}{r'(P^* \setminus S)} \cdot \pi'(P^* \setminus S) \leq \delta \pi'(P^* \setminus S)$ and thus

$$\pi'(W) \leq \delta \pi'(P^* \setminus S) + \max_{v \in P^* \setminus S} \pi'(v).$$

Note that P' is a feasible solution for the instance (G, π', r', C', D) which contains the nodes

$P^* \setminus W$. It holds that

$$\begin{aligned}\pi'(P' \setminus S) &= \pi'(P^* \setminus (S \cup W)) \\ &= \pi'(P^* \setminus S) - \pi'(W) \\ &\geq (1-\delta)(\pi'(P^*) - \pi'(S)) - \max_{v \in P^* \setminus S} \pi'(v) \\ &= (1-\delta)(\pi(P^*) - \pi(S)) - \max_{v \in P^* \setminus S} \pi'(v).\end{aligned}$$

We eventually obtain:

$$\begin{aligned}\pi(P') = \pi(S) + \pi(P' \setminus S) &= \pi(S) + \pi'(P' \setminus S) \\ &\geq \pi(S) + (1-\delta)(\pi(P^*) - \pi(S)) - \max_{v \in P^* \setminus S} \pi'(v).\end{aligned}$$

Note that $\max_{v \in P^* \setminus S} \pi'(v) \leq \delta \pi(S)$ by construction of S. Therefore

$$\pi(P') \geq \pi(S) + (1-\delta)\pi(P^*) - (1-\delta)\pi(S) - \delta\pi(S) \geq (1-\delta)\pi(P^*).$$

Observe that the optimal value of the modified instance (G, π', r', C', D) is thus at least $\pi'(P')$. By the guarantee of $\mathscr{A}$, we have $\pi(P) = \pi'(P) \geq (1-\delta)\pi'(P') = (1-\delta)\pi(P')$ and hence

$$\pi(P) \geq (1-\delta)\pi(P') \geq (1-\delta)^2 \pi(P^*) \geq (1-\varepsilon)\pi(P^*).$$

As for the running time, note that $\mathscr{A}$ runs in time polynomial in $(|\mathscr{I}| \cdot C')$ and therefore in time polynomial in $|\mathscr{I}|$, and we apply such an algorithm at most $O(n^{\frac{2}{\varepsilon}})$ times, i.e., once for every possible modified instance obtained by a different node subset S of cardinality $2/\varepsilon$. The result follows. □

We conclude this section by observing that the proofs of the two lemmas above can be combined and yield the following corollary, that will be used in the following two sections to show the existence of a PTAS for COP both on tree metrics and Euclidean metrics.

Corollary 2.3.4. *Suppose having an algorithm $\mathscr{A}$ that returns for every $\delta > 0$ a feasible solution to a given instance $\mathscr{I} = (G, d, \pi, r, C, D)$ of COP whose value is at least $(1-\delta)$ times the optimal value and that runs in time polynomial in $(|\mathscr{I}| \cdot \pi(V) \cdot C)$ for δ fixed. Then there is a PTAS for COP.*

2.4 Capacitated Orienteering on Euclidean metrics

The aim of this section is to give a PTAS for COP in the plane, extending the result by Chen and Har-Peled [CH08] who considered the case $\pi \equiv 1$, $r \equiv 1$, and $C = |V|$. They observe that even this special case is NP-hard in the plane by a reduction from the Traveling Salesman problem. An instance of OP with unit profits will be denoted by (G, D). In this section, $d(v, w)$ refers to the Euclidean distance (in dimension 2) between the points $v, w \in V$ whereas $d^P(v, w)$ refers

to the distance between v and w along a path P. We denote $n := |V|$ and we characterize each point by an x- and a y-coordinate.

2.4.1 The PTAS for OP in the plane

In this section we describe the PTAS for OP in the plane [CH08]. This result is based on a dynamic programming approach for k-TSP due to Mitchell [Mit99]. In particular, their algorithm relies on a dynamic program for the *k-path* problem, the unit profit case of Q-path problem: Given a graph G with Euclidean distances, nodes $s, t \in V$ and an integer $k \leq n$, find an s-t path of minimum length visiting at least k nodes.

We sketch the dynamic program in more detail, because it will be crucial for our result. Assume that all points are contained in an axis-parallel square $\mathscr{B}$. Following Chen and Har-Peled, we use the notion of a *window* for a closed axis-parallel rectangle $w \subseteq \mathscr{B}$. With this notion and given a fixed $\varepsilon > 0$, table entries can be defined for the dynamic program, each characterized by:

(i) a (minimal) window w that contains at least one point of V with its boundaries determined by (up to) four points of V,

(ii) integer $h \in \{0, 1, \ldots, |V|\}$ indicating the number of nodes that should be visited within w,

(iii) boundary information specifying at most $m = O(\frac{1}{\varepsilon})$ crossing segments (each determined by a pair of points of V, one inside or on the boundary of w, another outside w) for each of the four sides of the boundary of w,

(iv) connectivity constraints indicating which pairs of crossing segments are required to be connected within w.

Each table entry stores a set of short paths inside w meeting both the boundary and the connectivity constraints visiting at least h nodes. The stored information can be combined together in a recursive manner by subdividing each window into smaller windows by cutting w along a horizontal or vertical line. In the base case, if the window w has at most m points in its interior, then the problem can be solved by enumerating all possible solutions because $m = O(\frac{1}{\varepsilon})$ is constant. Otherwise, the algorithm tries all possible cuts for the current window recursively, enumerating over all possible choices of valid boundary information along this cut and computing the cheapest option (we refer to [CH08, Mit99] for details).

As shown in [Mit99], this dynamic program yields a PTAS for the k-path problem. The key insight in [CH08] is that a stronger result can be obtained with a different analysis. To this end, some more notation is needed.
A path P can be described by listing the nodes it contains in order as $P = \langle v_1, \ldots, v_k \rangle$. Let $\mu = \lceil 2/\varepsilon \rceil$ and let $1 = i_1 < \cdots < i_\mu = k$ be any sequence of $\mu \leq k$ integers. The *μ-skeleton* of P is the path $\langle v_{i_1}, \ldots, v_{i_\mu} \rangle$. Clearly, different choices of μ integers yield different μ-skeleton paths.

The μ-skeleton of P of maximum length is called *optimal* μ-skeleton of P. The difference between the length of P and its optimal μ-skeleton is the μ-*excess* of P, denoted by $X_{P,\mu}$.
The crucial point of the analysis in [CH08] is that the dynamic program for the k-path problem exceeds the optimal length only in terms of the μ-excess of the optimal path.

Theorem 2.4.1. *[CH08] Let P be any s-t-path that visits k nodes. For any fixed ε, the dynamic program computes a path from s to t that visits k nodes and has length at most $d^P(s,t) + \frac{X_{P,\mu+1}}{\mu+1}$, where $\mu = \lceil 2/\varepsilon \rceil$, in time polynomial in n.*

We now show how the k-path problem is related to Orienteering.
A feasible solution for an Orienteering instance (G, D) can be found as follows: starting from $k = 0$ up to $k = n$, apply the dynamic program above for the k-path problem on G and starting and ending nodes s and t. Let $\tilde{P}$ be the best path (maximum number of visited points) with length $\leq D$ found with this procedure. $\tilde{P}$ is then a feasible path for the given Orienteering instance. If k_{opt} is the number of points visited by an optimal solution P^* of the Orienteering instance, the authors in [CH08] show that

Lemma 2.4.2. *[CH08] The path $\tilde{P}$ visits at least $(1-\varepsilon)k_{opt}$ nodes.*

To prove this lemma, the authors show that for $k = (1-\varepsilon)k_{opt}$, the dynamic program has to find a path of length $\leq D$.

The crucial idea is to define a particular $(\mu+1)$-skeleton. Specifically, for a path P with k nodes, let $\mathscr{S}_P$ denote the $(\mu+1)$-skeleton of P obtained by subdividing P into segments with a similar number of nodes. Formally, we can find a sequence $1 = i_1 \leq \cdots \leq i_{\mu+1} = k$ of $\mu+1$ integers such that $\left|V_{P(i_j,i_{j+1})}\right| \geq \dfrac{k}{\mu}$ holds for all $j = 1,\ldots,\mu$, where $P\left(i_j, i_{j+1}\right) = \langle v_{i_j}, \ldots, v_{i_{j+1}} \rangle$ denotes the subpath of P from v_{i_j} to $v_{i_{j+1}}$. Let $X(P,\mathscr{S}_P) := d^P(s,t) - d^{\mathscr{S}_P}(s,t)$ denote the excess of P with respect to the $(\mu+1)$-skeleton $\mathscr{S}_P$.

Proposition 2.4.3. *[CH08] Let $\mu > 0$. Any s-t path P visiting k nodes can be shortcut (by eliminating some of its nodes) to a path P_{cut} from s to t that visits $\geq (1-\frac{1}{\mu})k$ nodes and that has length at most $d^P(s,t) - \frac{X(P,\mathscr{S}_P)}{\mu}$ such that $\mathscr{S}_P$ is a $(\mu+1)$-skeleton of P_{cut}.*

Let P^*_{cut} denote the path that we obtain if we apply Proposition 2.4.3 to P^* with $\mu = \lceil 2/\varepsilon \rceil$. Then P^*_{cut} visits $(1-\frac{1}{\mu})k_{opt} \geq (1-\varepsilon)k_{opt}$ nodes. Now apply Theorem 2.4.1 on P^*_{cut} and value $k := (1-\varepsilon)k_{opt}$. It follows that the dynamic program finds in polynomial time a path of length

$$\leq d^{P^*_{cut}}(s,t) + \frac{X_{P^*_{cut},\mu+1}}{\mu+1} \leq d^{P^*}(s,t) - \frac{X(P^*,\mathscr{S}_{P^*})}{\mu} + \frac{X_{P^*_{cut},\mu+1}}{\mu+1} \leq d^{P^*}(s,t) \leq D,$$

since $X(P^*,\mathscr{S}_{P^*}) \geq d^{P^*_{cut}}(s,t) - d^{\mathscr{S}_{P^*}}(s,t) \geq X_{P^*_{cut},\mu+1}$ holds because $\mathscr{S}_{P^*}$ is a $(\mu+1)$-skeleton for both P^* and P^*_{cut}.

In other words, the path $\tilde{P}$ found by the dynamic program visits at least $(1-\varepsilon)k_{opt}$ nodes. This

completes our review of the PTAS for Orienteering in the plane.

2.4.2 Our modifications

Similarly to the previous algorithm for OP, our result relies on a dynamic program, but for a more general *Capacitated Quota-path* Problem (CQPP), a capacitated generalization of the Q-path problem that is defined as follows: We are given a graph G with Euclidean distances, nodes $s, t \in V$, node profits $\pi : V \to \mathbb{N}$, demands $r : V \to \mathbb{N}$, a capacity bound C and an integer $Q \leq \sum_{v \in V} \pi(v)$. We want to find an s-t path of minimum length that collects at least profit Q by serving a total demand $\leq C$. Clearly, the *Capacitated Quota-path* problem is a generalization of the k-path problem that can be obtained by setting $\pi \equiv 1$, $r \equiv 1$ and $C = |V|$. We denote by (G, d, π, r, C, Q) an instance of the CQPP.

For a given $\varepsilon > 0$, we extend the dynamic program of the previous subsection as follows. First, instead of remembering an integer indicating the number of nodes that should be visited within a window w, we rather consider in (ii) a profit threshold to be collected, that is an integer h in the range $\{0, 1, \ldots, Q\}$.

Second, we need to handle the demands and the capacity bound. To this end, we additionally specify in a table entry of the dynamic program

(v) an integer j in the range $\{0, \ldots, C\}$ indicating that the total demand of the nodes covered inside the window w should not exceed j.

It is straightforward to show that the stored information can be combined in a recursive manner, following exactly the same steps in [CH08, Mit99] as previously described, to compute some $s - t$ path that collects an amount of profit $\geq Q$ from a subset of nodes of total demand $\leq C$. In particular, Theorem 2.4.1 extends to the theorem below.

Theorem 2.4.4. *Let P be an s-t-path that collects a profit $\geq Q$ from a total demand of $\leq C$. For any fixed $\varepsilon > 0$, the dynamic program computes a path from s to t that collects a profit $\geq Q$ from a total demand of $\leq C$ and has length at most $d^P(s,t) + \frac{X_{P,\mu+1}}{\mu+1}$, where $\mu = \lceil 2/\varepsilon \rceil$, in time polynomial in $(n \cdot C \cdot Q)$.*

Proof. First, let us argue about the running time of our modified dynamic program. As in [CH08], there are $O(n^4)$ choices of a (minimal) window w and $O(n^{2m})$ choices of crossing segments on each of the four sides of w (recall $m = O(\frac{1}{\varepsilon})$). Therefore, the total number of possible entries of the dynamic program is $O(n^4 \cdot (n^{2m})^4 \cdot C \cdot Q)$, because the number of possible connectivity constraints is constant (since we assume $\varepsilon > 0$ to be a fixed constant, we omit dependencies on ε in the discussion of the running time). Moreover, for computing each entry in a recursive manner, we have to consider $O(n)$ choices of a horizontal/vertical cut, $O(n^{2m})$ choices for new boundary information on the cut, $O(C \cdot Q)$ choices of demand

bounds and profit quotas on both sides of the cut and $O(1)$ choices of connectivity constraints. By multiplying the number of entries with the number of choices for the partitioning into subproblems, we obtain an overall time complexity for the algorithm that is polynomial in $(n \cdot C \cdot Q)$.

Now, let us argue about the length. Consider the instance for the k-path problem defined on the graph induced by the nodes in V_P, and obtained by setting $k = |V_P|$. The dynamic program used in [CH08] for this k-path problem outputs a path $\tilde{P}$ that visits all nodes in V_P and whose length is at most $d^P(s,t) + \frac{X_{P,\mu+1}}{\mu+1}$. Now observe that $\tilde{P}$ is also a solution to our CQPP instance since it has a profit $\geq Q$ and total demand $\leq C$. Moreover, $\tilde{P}$ will be implicitly considered by our extended dynamic program when applied to our CQPP instance for appropriate choices of h and j in each table entry. This is because our table entries are defined exactly as in [CH08] except that the number of nodes is replaced by profits and the extra information about the capacity is added.

Therefore, our dynamic program will output a path whose length is at most the length of $\tilde{P}$. The result follows. □

Let us explain how the CQPP is related to COP.
We can find a feasible solution for a COP instance (G, d, π, r, C, D) as follows. For all values of Q from 0 to $\pi(V)$, we apply the dynamic program above for CQPP on the instance (G, d, π, r, C, Q).

Let $\tilde{P}$ be the best path (in terms of profit) with length $\leq D$ found with this procedure. $\tilde{P}$ is then a feasible path for the given COP instance. If P^* is the optimal path of our COP instance, then we obtain

Lemma 2.4.5. $\pi(\tilde{P}) \geq (1-\varepsilon)\pi(P^*)$.

Proof. We show that for $Q = (1-\varepsilon)\pi(P^*)$, our extended dynamic program has to find a path of length $\leq D$.
Our first claim is that P^* can be shortcut (by eliminating some of its nodes) to a path P^*_{cut} from s to t that collects profit $\geq (1-\varepsilon)\pi(P^*)$ and has small length. This can be seen as follows: replace every node $v \in V_{P^*}$ with $\pi(v)$ copies (i.e. $\pi(v)$ nodes with the same coordinates) and apply Proposition 2.4.3 to P^* with $k := \pi(P^*)$ and $\mu = \lceil 2/\varepsilon \rceil$.

We thereby obtain a path P^*_{cut} that collects profit $\geq (1-\varepsilon)\pi(P^*)$ and has length $\leq d^{P^*}(s,t) - \frac{X(P^*,\mathscr{S}_{P^*})}{\mu}$. Now apply Theorem 2.4.4 for P^*_{cut} and value $Q := (1-\varepsilon)\pi(P^*)$. It follows that the dynamic program finds a path of length

$$\leq d^{P^*_{cut}}(s,t) + \frac{X_{P^*_{cut},\mu+1}}{\mu+1} \leq d^{P^*}(s,t) - \frac{X(P^*,\mathscr{S}_{P^*})}{\mu} + \frac{X_{P^*_{cut},\mu+1}}{\mu+1} \leq d^{P^*}(s,t) \leq D,$$

since $X(P^*,\mathscr{S}_{P^*}) \geq X_{P^*_{cut},\mu+1}$ as we have already argued.

In other words, the path $\tilde{P}$ found by our dynamic program collects profit at least $(1-\varepsilon)\pi(P^*)$.

□

Putting all together, and using Corollary 1 we get

Theorem 2.4.6. *There is a PTAS for COP on Euclidean metrics.*

2.5 Results on tree metrics

The main result of this section is a PTAS for COP on tree metrics. We extend our results for COP also to Q-path and MEP.

As pointed out in the preliminaries, we will consider a tree $T = (V, E)$ with distances $d : E \to \mathbb{N}$ instead of the complete graph given by the metric closure of T. We denote in the following $n = |V|$ and $\pi(V) = \sum_{v \in V} \pi(v)$. Moreover, for any two nodes $u, v \in V$ we denote the unique $u - v$ path in T by P_{uv}.

Lemma 2.5.1. *OP, Q-path, and MEP are NP-hard on star metrics.*

Proof. We give a reduction from the *Knapsack* problem: Given non-negative integers n, $c_1, \ldots, c_n$, $w_1, \ldots, w_n$, W and K, it is NP-hard to decide whether there is a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in S} c_j \geq K$ and $\sum_{j \in S} w_j \leq W$ [GJ79]. We construct a star T with $n+1$ vertices s, $v_1, \ldots, v_n$ and set $d(e_i) := w_i$ for edge $e_i = \{s, v_i\}$ and $i = 1, \ldots, n$. Further, we define profits $\pi(v_i) = c_i$ for $i = 1, \ldots, n$ and $\pi(s) = 0$.

It follows that it is NP-hard to decide whether there is a closed walk rooted at s of length $\leq 2W$ and profit $\geq K$. This implies that OP, MEP and Q-path are NP-hard on star metrics. □

We observe that by a straightforward reduction from the Knapsack, COP is already NP-hard on graphs that are a line.

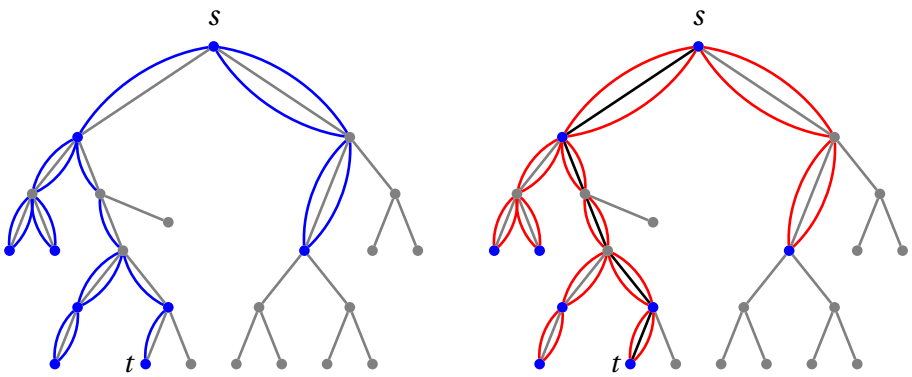
2.5.1 COP on tree metrics

In the following, we present the PTAS for COP on tree metrics. First, we will describe a dynamic program of pseudopolynomial size that we can turn then into a polynomial time algorithm at small loss by the results of section 2.3

Theorem 2.5.2. *COP on tree metrics can be solved in time $O(n\pi(V)^2 C^2)$.*

Proof. Before describing the algorithm, we need to state some observations.

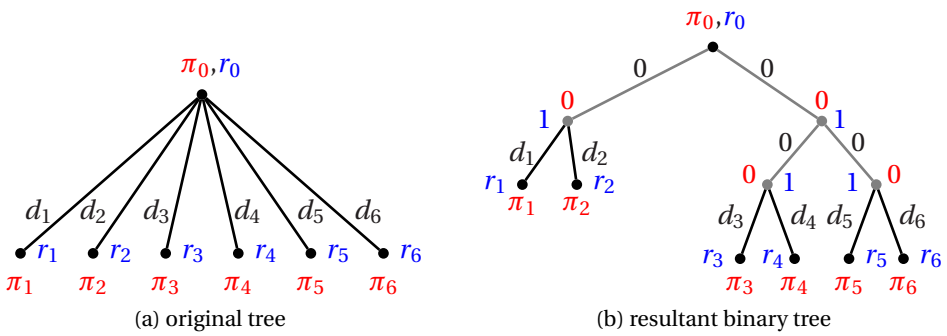
Suppose having a tree graph T and a walk P that is a feasible solution for our instance. It is easy to see that P visits the edges of the unique $s - t$ path P_{st} in T exactly once and every other edge in P exactly twice. Therefore, we could set the length of the edges of path P_{st} to zero and



(a) original instance (b) after modification

Figure 2.1 – Modification along the unique s-t-path. The optimal s-t-walk in the original instance is indicated in blue whereas the equivalent, optimal closed walk around s in the modified instance is shown in red. Note that in the modified instance, the black edges on the unique s-t-path have weight 0. The selected vertices whose demand is fulfilled and whose profit is thus collected are indicated in blue, respectively.

hence reduce the problem to finding a *closed walk* starting and ending at node s of length $\leq R := D - d^P(s,t)$, that collects maximum profit from nodes whose total demand does not exceed capacity C (see Fig. 2.1). In the following, such a closed walk will be referred to as a *tour* from s. By adding edges of zero length and nodes with zero profit and unit demand (see Fig. 2.2), we can also assume that tree T is binary.



(a) original tree (b) resultant binary tree

Figure 2.2 – Modifications to obtain a binary tree. Profits are indicated in red, demands in blue and edge weights in black.

Consider the following dynamic program. We have table entries $A_v[k,c]$ storing the length of the shortest tour from v in the subtree below v that collects profit at least k using at most c units of capacity. We initialize the table for each $v \in V$ as follows:

$$A_v[k,c] = \begin{cases} 0 & \text{if } k = 0; \\ 0 & \text{if } c \in \{r(v), \dots, C\} \text{ and } k \leq \pi(v); \\ +\infty & \text{otherwise} \end{cases}$$

Note that we are interested in values of c up to C, and of k up to $\pi(V) = \sum_v \pi(v)$. We consider

the tree rooted at s and compute table entries for the node w as soon as the table entries for its two children v_1 and v_2 have already been computed. We can either collect the profit of w and decrease the capacity according to its demand or ignore it. In both cases, we have to consider all possible ways to split profit k and capacity c among its two children. Let χ_k be 1 if $k > 0$ and 0 otherwise. We set $A_w[k,c] = \min\{A_w[k,c], (\mathrm{A}), (\mathrm{B})\}$ where

$$(\mathrm{A}) := \min_{\substack{k'=0,\dots,k,\\ c'=0,\dots,c}} \left\{A_{v_1}[k',c'] + 2d(v_1,w)\cdot\chi_{k'} + A_{v_2}[k-k',c-c'] + 2d(v_2,w)\cdot\chi_{k-k'}\right\}$$

$$(\mathrm{B}) := \min_{\substack{k'=0,\dots,k-\pi(w),\\ c'=0,\dots,c-r(w)}} \left\{\begin{array}{c} A_{v_1}\left[k',c'\right] + A_{v_2}\left[(k-\pi(w))-k',(c-r(w))-c'\right] \\ +2d(v_1,w)\cdot\chi_{k'} + 2d(v_2,w)\cdot\chi_{(k-\pi(w))-k'} \end{array}\right\}$$

Value (A) corresponds to combining tours rooted at v_1 and v_2 to a cheapest tour of capacity c and profit k without considering w, while (B) incorporates a profit of $\pi(w)$ for visiting w.

Once the root of the tree is processed, the maximum value of k is output such that $A_s[k,C] \leq R$.

Every node in the tree is processed exactly once by the dynamic program. The running time is $O(\pi(V)\cdot C)$ to compute the minimum for each table entry and thus we obtain the claimed overall running time of $O(n\pi(V)^2C^2)$.

□

Theorem 2.5.2 and Lemma 2.3.1 imply the following corollary.

Corollary 2.5.3. *There is an FPTAS for OP on tree metrics.*

Similarly, we can combine Theorem 2.5.2 with Corollary 1 to obtain the main result of this section.

Theorem 2.5.4. *There is a PTAS for COP on tree metrics.*

2.5.2 Q-path and MEP on tree metrics

In this section, we show that the roles of profit and length can be exchanged in the dynamic program used in the proof of Theorem 2.5.2. Thus we obtain a PTAS for Q-path and for MEP on tree metrics.

We denote in the following $L = \sum_{e\in E} d(e)$.

Theorem 2.5.5. *Q-path on tree metrics can be solved in time $O(nL^2)$.*

Proof. We can assume by the modifications presented in the proof of Theorem 2.5.2 that the given tree graph is binary and the length of the path between s and t is zero.

Consider the following dynamic program. We have table entries $A_v[\ell]$ storing the maximum profit of a tour from v in the subtree below v that has length at most ℓ. We initialize the table for each leaf $v \in V$ as follows:

$$A_v[\ell] = \begin{cases} \pi(v) & \text{if } \ell \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Note that we are interested in values of ℓ up to L. We consider the tree rooted at s and compute table entries for the node w as soon as the table entries for its two children v_1 and v_2 have been computed. We have to consider all possible ways to split the length budget ℓ among the children v_1 and v_2. We set

$$A_w[\ell] := \max_{\ell'=0,\dots,\ell} \left\{ \pi(w) + A_{v_1}\left[\ell' - 2d(v_1,w)\right] + A_{v_2}\left[(\ell - 2d(v_2,w)) - \ell'\right] \right\}$$

This value corresponds to combining tours rooted at v_1 and v_2 to the most profitable tour of length ℓ considering a profit of $\pi(w)$ for visiting w.

Once the root of the tree is processed, we output the minimum value of ℓ such that $A_s[\ell] \geq Q$.

Every node in the tree is processed exactly once by the dynamic program. The running time is $O(L)$ to compute the minimum for each table entry and thus we obtain the claimed overall running time of $O(nL^2)$.

□

In order to turn the algorithm from Theorem 2.5.5 into an FPTAS for Q-path, we apply a standard scaling and rounding technique.

Corollary 2.5.6. *There is an FPTAS for Q-path on tree metrics.*

We can also extend this result to MEP.

Theorem 2.5.7. *There is an FPTAS for MEP on tree metrics.*

Proof. We prove that MEP and Q-path are equivalent on trees, i.e there is an α-approximation algorithm for MEP on trees if and only if there is an α-approximation algorithm for Q-path on trees. Because the if-part is clear by definition, we only show how MEP can be reduced to Q-path on trees.
For a given an instance (G, d, π, Q) of MEP where G is a tree graph, let P^* be an optimal solution for MEP. It is easy to see that P^* visits the edges of the unique $s-t$ path P_{st} in T exactly once and every other edge in P^* exactly twice. Therefore, we can contract the edges of path P_{st} and hence reduce the problem to finding a *closed walk* starting and ending at node s with profit $\geq Q' := Q - \pi(P_{st})$ of smallest total length. Let G' denote the tree graph obtained from T by

contracting the edge of path P_{st}. We observe that P^* induces a feasible solution P' for Q-path to the instance (G', d, π, Q') with start and end vertex s of length $d^{P^*}(s,t) - d(P_{s,t})$. Hence if we apply an α-approximation algorithm for Q-path in G', we obtain a walk P of length at most

$$d^P(s,t) \leq \alpha d^{P'}(s,t) = \alpha(d^{P^*}(s,t) - d(P_{s,t}))$$

which is precisely the excess of the walk $\tilde{P}$ obtain from concatenating P and P_{st} in G. This concludes the reduction and with Corollary 2.5.6, we obtained the claimed FPTAS for MEP. □

2.6 Orienteering on planar graphs

In this section, we consider OP on a metric induced by a planar graph. Since OP can be seen as a prize collecting network design problem, the recent results for prize collecting Steiner Tree, prize collecting Steiner Forest and prize collecting Stroll [BCE+11] on planar graphs give rise to the question whether there exists a PTAS for OP as well. As pointed out in [BCE+11], the techniques developed for those prize collecting problems do not easily generalize for OP. We outline here the main ideas of their techniques and show a roadmap of the challenges to overcome for Orienteering.

The concept of the *treewidth* of a graph is crucial for many algorithms on planar graphs.

Definition 2.6.1. *Given a graph G, a tree decomposition of G is a pair (T, X) of a tree T and a function $X\colon V(T) \to 2^{V(G)}$ (each node i of the T is associated to a subset $X(i) \subset V(G)$ of the vertices in G) with the following properties:*

- *Every vertex $v \in V(G)$ is contained in at least one set $X(i)$ for a node i of T.*
- *Every edge $\{u, v\} \in E(G)$ is a subset of at least one set $X(i)$ for some node i of T.*
- *If $v \in X(i) \cap X(j)$, then $v \in X(h)$ for every node h of T on the unique path from i to j in T.*

The width of a tree decomposition (T, X) of G is the maximum cardinality of a vertex subset minus one, i.e. $\max_{i \in V(T)} |X(i)| - 1$. The treewidth of G is the minimum width of a tree decomposition of G. We say the treewidth of a family of graphs is bounded if it is $O(1)$.

The problem of determining the treewidth of a graph is NP-hard [ACP87]. However, there is a polynomial time algorithm that decides for a fixed constant γ whether a graph has treewidth at most γ [Bod96]. The seminal work by Baker [Bak94] introduces the concept of reducing a problem on a planar graph to the corresponding problem on a subgraph of constant treewidth. Using this technique, she obtained PTAS's for Minimum Vertex Cover and Maximum Independent Set. A similar framework was recently used to reduce several prize-collecting problems on planar graphs to their corresponding versions on graphs with bounded treewidth [BCE+11]. More precisely, an α-approximation algorithm for graphs of bounded treewidth implies an

$(\alpha+\varepsilon)$-approximation algorithm on planar graphs for each of those prize-collecting problems. The basic step to apply this framework also to OP hence is to consider graphs with bounded treewidth. We observe that our results from section 2.5 for OP and MEP extend to this graph class. In particular, the problems remain NP-hard, but we can use dynamic programming to obtain a PTAS for OP and MEP.

Theorem 2.6.2. *There is a PTAS for OP and MEP on a metric induced by a graph with bounded treewidth.*

We focus on the general strategy and refer to the technical report [BC14] for the proof and further details.

Solving OP on graphs with bounded treewidth is only a preliminary step towards a potential PTAS for OP on planar graphs. The challenging difficulty lies in finding a suitable reduction of the planar instance to one or several graphs whose treewidth is bounded. In [BCE+11], the crucial step in the reduction to graphs with bounded treewidth presented is to restrict the instance at a small loss in the objective function to a subgraph with a spanning tree of very small total edge distance. This is achieved by a clever application of a primal-dual approximation algorithm for the respective prize-collecting problem in general graphs. How a similar result can be obtained for OP or MEP is the subject of ongoing research. It remains a challenging open question whether a PTAS for OP or MEP on planar graphs can be found.

2.7 Orienteering with deadlines

Recall that the Orienteering problem with deadlines (OPD) asks to find a rooted path that collects as much profit as possible by visiting vertices before their deadline. We observe that the result for OPD obtained in this section can be extended to hold also in the more general case in which both endpoints s and t of the path are specified. This variant is referred to as *point-to-point* (P2P). First, we consider an interesting special case where the deadlines are proportional to the root distance of the vertices and then we present the approximation result for OPD.

2.7.1 Orienteering with delay factor

We consider the special case in which we are given a graph $G=(V,E)$ with metric distances $d: E \to \mathbb{N}$, node profits $\pi: V \to \mathbb{N}$, a root node s and a delay factor R. Every vertex is to be visited by time $D(v) = R \cdot \ell_v$ to obtain its profit (recall that ℓ_v denotes the shortest distance from v to the root s). We refer to this problem as Orienteering with delay factor (OP-R).

The case $R=1$ can be solved by using dynamic programming techniques: First, we consider only edges that lie on some shortest path to the root. We orient those edges towards the endpoint that is farther from the root and obtain a directed acyclic graph (DAG). A topological order on this DAG (cf. [KV12]) can be used to devise a dynamic program that computes the

optimal profit in polynomial time. We therefore assume in the remainder of this section that $R > 1$ holds.

Tree metrics

In her master thesis, Tholen [Tho14] observed that OP-R is NP-hard already on tree metrics. We refer to this thesis for the details of the reduction from Knapsack to OP-R. We note that P2P-OP-R is NP-hard already on star metrics, i.e., if both endpoints of the path are specified.

Proposition 2.7.1. *OP-R is NP-hard on tree metrics.*

Furthermore, there is an FPTAS for OP-R on star metrics based on dynamic programming [Tho14] that can easily be extended to hold also for P2P-OP-R.

General graphs

The idea of our algorithm for general graphs is based on partitioning the vertices according to their shortest distance from the root. This approach has been successfully applied to several vehicle routing problems before [NR12, CKP12, FS14]. The main result of this section is as follows.

Theorem 2.7.2. *If $R > 1$, there is a $O\left(\log \frac{R}{R-1}\right)$-approximation algorithm for OP-R.*

Proof. Intuitively, our algorithm first partitions the vertex set into $O(\log \ell_{\max})$ subsets according to their distance from the root and then we apply on each set of the partition an α-approximation algorithm for OP with a suitable length bound. Note that this strategy immediately yields a $O(\alpha \log \ell_{\max})$-approximation algorithm. However, by carefully choosing the length bound, we can turn the paths into rooted tours and concatenate several of them together. The details are presented in Algorithm 2. Let us argue that the path P returned by the algorithm visits every vertex before its deadline. We prove this by induction on k (cf. step 10), the case $k = 0$ being obvious by the choice of the length bound in step 5 of the algorithm. For every $k > 0$, $j = 1, \dots, M$ and $w \in \tilde{P}_{j+kM}$ we get

$$\begin{aligned} d^P(r,w) &\leq \ell_w + (R-1)2^{j+kM-2} + \sum_{k'<k} (R+1)2^{j+k'M} \\ &\leq \ell_w \left(1 + \frac{R-1}{2}\right) + (R+1)\frac{2^{j+kM}}{2^M - 1} \\ &\leq \ell_w \left(1 + \frac{R-1}{2} + \frac{R-1}{2}\right) = R\ell_w. \end{aligned}$$

The first inequality follows from the length bound in step 5 and the induction hypothesis, since every path $\tilde{P}_{j+k'M}$ can be turned into a tour of length at most $R2^{j+k'M} + 2^{j+k'M}$. The bound for finite geometric series implies the second inequality, whereas the last inequality is direct from our choice of M.

Algorithm 2 $O\left(\alpha \log \frac{R}{R-1}\right)$-approximation algorithm for OP-R

1: $V_i := \{s\} \cup \left\{v \in V : 2^{i-1} \leq \ell_v < 2^i\right\}$ for all $i \geq 1$.
2: $M \leftarrow 4 + \left\lceil \log_2 \left(\frac{R}{R-1}\right)\right\rceil$
3: **for** $i = 1$ **to** $\left\lceil \log_2 \ell_{\max}\right\rceil + 1$ **do**
4: **for** every vertex $w \in V_i \setminus \{s\}$ **do**
5: Compute a path $\tilde{P}_i(w)$ from s to w of length $\leq \ell_w + (R-1)2^{i-2}$ using the α-approximation algorithm for OP on $G[V_i]$.
6: Set $\tilde{P}_i$ to be the maximum profit path among $\tilde{P}_i(w)$ for $w \in V_i$.
7: **end for**
8: **end for**
9: **for** $j = 1, \ldots, M$ **do**
10: Obtain a path P_j by concatenating the paths $\tilde{P}_{j+kM}$ for $k \geq 0$ using shortest paths between their endpoints.
11: **end for**
12: $P \leftarrow P_{j^*}$ for $j^* := \operatorname{argmax}_{j=1,\ldots,M} \pi(P_j)$.
13: **Return** P

The running time is obviously given by $O(n \log \ell_{\max})$ many calls of the OP subroutine and M path concatenations. Since $M = O\left(\log \frac{R}{R-1}\right)$ and $\log_2 \ell_{\max}$ are polynomial in the input size, our algorithm for OP-R runs in polynomial time.

It remains to show the quality of the path P with respect to the optimal solution P^*.
Claim: $\pi(P) \geq \frac{1}{4M\alpha}\pi(P^*)$

We prove the claim in several steps. First, we show that we obtain a $\frac{1}{4\alpha}$ fraction of the profit of the optimal solution restricted to V_i for all $i \geq 1$. Second, we show that some segments of the optimal solution can be concatenated as in step 10 and thus third, we output at least a $\frac{1}{M}$-fraction of the total profit of the concatenated paths.

Let $V_i^* := V_i \cap V(P^*)$ denote the intersection of every V_i with the optimal solution for $i \geq 1$. Further, we shortcut the optimal path P^* to contain only the vertices V_i^* and denote this path by $\hat{P}_i^*$. As the sets V_i form a partition of the non-root vertices, we have $\pi(P^*) = \sum_{i \geq 1} \pi(\hat{P}_i^*)$ (w.l.o.g. $\pi(s) = 0$). Every path $\hat{P}_i^*$ has excess at most $(R-1)2^i$ by the feasibility of the optimal solution. Using Proposition 2.1.1, the path $\hat{P}_i^*$ can thus be shortcut to a path $\tilde{P}_i^*$ that has profit at least $\frac{1}{4}\pi(\hat{P}_i^*)$ and excess at most $(R-1)2^{i-2}$. This implies that the path $\tilde{P}_i$ found in step 6 of Algorithm 2 has profit at least $\frac{1}{\alpha}\pi(\tilde{P}_i^*) = \frac{1}{4\alpha}\pi(\hat{P}_i^*)$ by the appoximation guarantee of the OP subroutine.

These paths $\tilde{P}_i^*$ can be concatenated as in step 10 to M paths P_j^* for $j = 1, \ldots, M$. Each such path P_j^* visits all vertices before their deadline as shown for every P_j in the beginning of the proof.

We conclude the proof of the claim by observing that the M paths P_j^* have a total profit of $\frac{1}{4\alpha}\pi(P^*)$.

It follows that Algorithm 2 is a $O\left(\log \frac{R}{R-1}\right)$-approximation algorithm for OP-R using the $(2+\varepsilon)$-approximation algorithm for OP [CKP12]. □

We remark that this result implies a constant factor approximation algorithm if $R = 1 + \Omega(1)$. We complement Theorem 2.7.2 with the following observation that rules out a PTAS for OP-R.

Theorem 2.7.3. *OP-R is NP-hard to approximate to within a factor of* $\frac{2138}{2137}$.

Proof. We consider the following special case of OP that we call OP-cycle: Given a complete graph G with metric distances on the edges, a root node s and a length bound D, we want to find the cycle rooted at s with a maximum number of vertices that has length at most D. OP-cycle is clearly the special case of OP where both endpoints are the same and the profits are $\pi \equiv 1$. From the work of Blum et al. [BCK+03] and a stronger inapproximability result for the TSP with edge lengths 1 or 2 [KS12], it follows that OP-cycle is NP-hard to approximate with a factor better than $\frac{1069}{1068}$.

We reduce OP-cycle to OP-R. Let k^* be the number of vertices in the optimal solution of OP-cycle (we simply try all $n = |V(G)|$ possibilities). Suppose there is a $(1+\beta)$-approximation algorithm for OP-R with $\beta \leq \frac{1}{2137}$. We construct a new graph G' by adding a supersource s' that is connected only to s by an edge of length $2D$ [1]. Furthermore, we add an extra vertex t at distance $10D$ from s' that is the designated endpoint of the solution to OP-R. The profit of t is set to k^*, $\pi(s) := 0$, and the profits of all other vertices are 1. We set $R = 1.5$ and apply the $(1+\beta)$-approximation algorithm to G' with start vertex s'.

We observe that every tour around s in G of length at most D that visits k' vertices corresponds to a walk from s' that visits s', t and the k' vertices on the tour before their deadlines, and has profit $k^* + k'$. Conversely, let P' be a solution to OP-R on G'. By construction, P' restricted to G is a tour C' around s. In particular, if P' ends in t and thus has length at most $15D$, then C' has length at most D.

Let P denote the output of the $(1+\beta)$-approximation algorithm applied to G' with start vertex s'. Since $\beta < 1$, the path P has to visit t. As we previously observed, this means that P restricted to G is a tour C around s with length at most D. Let k be the number of vertices on C. We have that

$$k = \pi(P) - k^* \geq \frac{1}{1+\beta} 2k^* - k^* = \frac{1-\beta}{1+\beta} k^* \geq \frac{1068}{1069} k^*,$$

so we get a $\frac{1069}{1068}$-approximation algorithm for OP-cycle, a contradiction. □

[1] In order to obtain a complete metric graph, we can consider the metric closure of G', cf. Section 2.1.3.

We remark that the result can be extended to OP-R with unit profits $\pi \equiv 1$.

2.7.2 Arbitrary deadlines

We define the individual excess or regret $X(v) := \lfloor D(v) - \ell_v \rfloor$ of a vertex $v \in V$. Further, let $X_{\max} = \max_{v \in V} X(v)$ and $X_{\min} = \min_{v \in V, X(v) > 0} X(v)$ denote the maximum and minimum individual regret, respectively. We can assume the total profit of all vertices to be polynomially bounded (otherwise we can apply the scaling and rounding techniques presented in the proof of Lemma 2.3.1 and we lose at most a small constant factor in the approximation guarantee).

Theorem 2.7.4. *There is a $O\left(\log \frac{X_{\max}}{X_{\min}}\right)$-approximation algorithm for OPD.*

Proof. We partition the vertices as follows

$$V_0 := \{s\} \cup \{v \in V \colon X(v) = 0\}$$
$$V_i := \{s\} \cup \left\{v \in V \colon 2^{i-1} \le X(v) < 2^i\right\} \qquad \text{for all } i \ge 1.$$

We observe that a rooted path that visits as many vertices in V_0 as possible can be found in polynomial time by dynamic programming. Our idea consists in applying an α-approximation algorithm for OP to every vertex set V_i ($i \ge 1$) with length bound $D = \ell_w + 2^{i-1}$ between s and any vertex $w \in V_i$. We output the best path that we can find with this procedure over all $i \ge 0$. Let M denote the number of non-empty sets V_0, V_i for $i \ge 1$. We show that the previous strategy (summarized in Algorithm 3) yields an $\alpha 2M$-approximation algorithm.

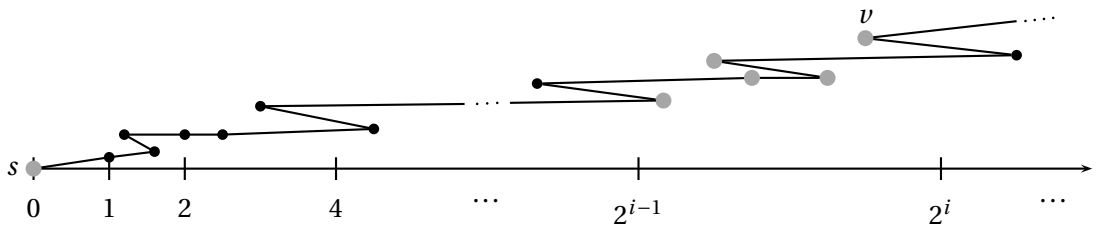


Figure 2.3 – Optimal solution P^* with the vertices arranged by distance from s; its restriction V_i^* to the set V_i is indicated in gray. Note that the vertical positioning of P^* is only for the sake of exposition.

Let P^* denote the optimal solution and V_i^* the vertices in V_i visited by P^*. We fix an index i with $V_i^* \neq \emptyset$. Let v denote the vertex in V_i^* visited last on P^* (starting from the root). Figure 2.3 reflects the situation and notation. By definition of V_i, the excess of the path from s to v along P^* has excess at most 2^i. This means that with Proposition 2.1.1 we can split this path into two parts in order to obtain two rooted paths each having profit at least $\frac{1}{2}\pi(V_i^*)$ from vertices in V_i. One of those paths has at most excess 2^{i-1}, so the α-approximation algorithm for OP has to find eventually a path with profit $\ge \frac{1}{\alpha 2}\pi(V_i^*)$ (for some choice of the end node $w \in V_i$).

We observe that every path P_w for $w \in V_i$ has excess at most 2^{i-1} (i.e. w has regret $\le 2^{i-1}$). By the monotonicity of the excess/regret, every vertex $v \in V_i$ on P_w has regret at most 2^{i-1} and thus all deadlines of the vertices on P_w are met. Furthermore, we can assume that $X_{\max} >$

Algorithm 3 $O\left(\alpha \log \frac{X_{\max}}{X_{\min}}\right)$-approximation algorithm for OPD

1: $V_0 := \{s\} \cup \{v \in V : X(v) = 0\}$
2: $V_i := \{s\} \cup \left\{v \in V : 2^{i-1} \leq X(v) < 2^i\right\}$ for all $i \geq 1$
3: Compute optimal solution P_0 for OPD restricted to V_0 by dynamic programming.
4: **for** $i = 1$ **to** $\lceil \log_2 X_{\max} \rceil$ **do**
5: **for** every $w \in V_i \setminus \{s\}$ **do**
6: Find maximum profit path $\tilde{P}_w$ from s to w in $G[V_i]$ of length at most $\ell_w + 2^{i-1}$ using the α-approximation algorithm for OP.
7: **end for**
8: $P_i \leftarrow \operatorname{argmax}_{w \in V_i \setminus \{s\}} \pi\left(\tilde{P}_w\right)$.
9: **end for**
10: $\tilde{P} \leftarrow \operatorname{argmax}_{i \geq 0} \pi(P_i)$.
11: $P \leftarrow \tilde{P}$
12: **Return** P

$2X_{\min}$ holds as otherwise only two sets V_i and V_{i+1} are non-empty and thus we immediately obtain a constant factor approximation algorithm.

It remains to show that there are at most $M = O\left(\log \frac{X_{\max}}{X_{\min}}\right)$ non-empty sets V_i. Let $i_{\min}$ denote the minimum index ≥ 1 with $2^{i_{\min}-1} \leq X_{\min} < 2^{i_{\min}}$ and analogously $i_{\max}$ such that $2^{i_{\max}} \leq X_{\max} < 2^{i_{\max}+1}$. Then there are at most

$$i_{\max} - i_{\min} + 1 = \log_2\left(\frac{2^{i_{\max}}}{2^{i_{\min}}}\right) + 1 \leq \log_2 \frac{X_{\max}}{X_{\min}} + 1 = O\left(\log \frac{X_{\max}}{X_{\min}}\right)$$

non-empty sets V_i.

We conclude that Algorithm 3 is a $O\left(\log \frac{X_{\max}}{X_{\min}}\right)$-approximation algorithm for OPD using a $(2+\varepsilon)$-approximation algorithm for OP [CKP12]. □

As we observed previously, this result can be easily extended to the point-to-point case as well.

Corollary 2.7.5. *There is a $O\left(\log \frac{X_{\max}}{X_{\min}}\right)$-approximation algorithm for P2P-OPD.*

Proof. The only small modifications that are necessary in Algorithm 3 consist of adapting the length bounds in steps 3 and 6 where the Orienteering subroutine is applied. In the dynamic program used for step 3, we ensure that the computed path can be extended to the endpoint t such that the deadline of t is met. In step 6, we set the length bound to the minimum of $\ell_w + 2^{i-1}$ and $D(t) - d(w,t)$ for a vertex $w \in V_i$. Note that this quantity may be negative; in this case we continue the FOR-loop with the next choice of a vertex in V_i. The rest of the algorithm remains unchanged and the analysis works out as before. To see this, note that the optimal

solution ends in t and restricted to each interval, it can be split into two s-t-paths with half of the total profit each by connecting simply the rooted paths obtained from Proposition 2.1.1 to t. Clearly, the deadline of t is met for both such paths. We conclude that Algorithm 3 outputs a solution with profit at least half of the maximum profit of the optimal solution restricted to some interval. □

The challenging question for OPD is whether a constant factor approximation can be achieved. We observe that the analysis of the $O(\log OPT)$-approximation algorithm for OPD presented in [BBCM04] is tight, i.e., there is an instance on which their algorithm obtains only a logarithmic fraction of the optimal profit [Tho14].

It is conjectured that OP-TW has a $O(\log OPT)$-approximation algorithm [CKP12] and they observe that an α-approximation algorithm for OP-TW with $L_{\max} \leq 2L_{\min}$ implies an α-approximation algorithm for OPD. Furthermore, an α-approximation algorithm for OPD implies a $O(\alpha \log OPT)$-approximation algorithm for OP-TW in general. In other words, an $O(1)$-approximation algorithm for OPD would prove the conjecture for OP-TW.

3 The School Bus problem and related variants

3.1 Introduction

In the *School Bus Problem* (SBP), we have to route buses that pick up children and take them from their homes to a school. However, parents do not want their children to spend too much time on the bus relative to the time required to transport them to school by car along a shortest path. In fact, if the additional distance travelled by the bus exceeds a certain *regret* threshold, the parents would rather drive their children to school by themselves, which is unacceptable. Subject to this, the goal is to cover all of the children using a minimum number of buses. Recall that the setting of a school bus system is not intrinsic to the considered problems or to the methods we develop for them. The observations and statements presented in this section translate directly to other transportation or vehicle routing tasks.

Formally, we are given an undirected network $G = (V, E)$ with metric distances on the edges $d : E \to \mathbb{N}$, a depot or root node $s \in V$ representing the school, and a set $W \subseteq V$ representing the houses of children. Additionally, we are given a bus capacity $C \in \mathbb{N}$, and a regret bound $R \in \mathbb{N}$. The aim is to construct a minimum cardinality set $\mathscr{P}$ of walks ending at s (bus routes) and assign each child $w \in W$ to be the responsibility of some bus $p(w) \in \mathscr{P}$ such that (i) for each walk $P \in \mathscr{P}$, the total number of children w with $p(w) = P$ is at most the capacity C; (ii) for every child the regret bound is respected, that is: $d^P(w,s) \leq d(w,s) + R$, where $d^P(w,s)$ is the distance from the child w to the school s on the walk $p(w)$, and $d(w,s)$ is the shortest distance from w to s in the graph G.

In an additional variation of the problem, the goal is to minimize the total length of the walks instead of their number. We refer to this variant of the School Bus Problem as SBP-Σ. The School Bus Problem with Regret Minimization (SBP-R) is a further variation where we have a fixed number N of buses we can use, and the goal is to minimize the maximum regret R.

For these variants of the School Bus problem, the regret bound is added to the shortest distance from the root to obtain a deadline for each vertex. However, it is also very interesting to consider a regret deadline that is a multiple of the shortest distance from the root. In that

case, we are given a delay factor $R \geq 1$ instead, and again we ask to minimize the number of routes (SBP-mult) or the total length (SBP-Σ-mult). SBP-R-mult refers to the variation in which the number of bus routes is given and we seek to minimize the maximum delay factor R.

Like many vehicle routing problems, all mentioned variants of the School Bus problem are strongly NP-hard, even on the simplest of graphs. To see this, consider a star centered at the school s with children located at all other vertices. If k of the edges are very long, say longer than $\frac{R}{2}$, then in any feasible solution we need at least k buses: one bus starting at each of these k endpoints. Then, determining if such k buses are sufficient with a regret bound R is precisely equivalent to solving the bin-packing decision problem with k bins of capacity $\frac{R}{2}$ and objects sized according to the distances from the remaining children to the school. It follows that bin-packing can be reduced to SBP and SBP-R on stars. We show in Section 3.4.1 that also SBP-mult and SBP-R-mult are NP-hard on stars. Observe that SBP-Σ and SBP-Σ-mult are trivial on stars, since the optimal solution just sends a new bus for every leave. However, we show in Section 3.4.1 and 3.2.2 that both problems are NP-hard already on trees of height 2.

Many variants of vehicle routing have been studied in the context of exact, approximate, and heuristic algorithms. For variations of the School Bus Problem, heuristic methods for practical applications have been examined (see the survey in [PK10]), but there was no literature concerning formal approximability and inapproximability results prior to our work. Our goal is to advance the state of the art in this perspective.

3.1.1 Related work

There is an enormous number of results concerning vehicle routing problems; we refer to the survey in [TV01] and briefly discuss previous work on the School Bus problem and related questions.

The Capacitated Vehicle Routing Problem (CVRP) enforces a limit C on the number of visited locations in each route, and the goal is the minimization of the total length of all the routes. The paper [HRK85] establishes a strong link to the underlying Travelling Salesman Problem (TSP) by giving an approximation algorithm that relies on approximation algorithms for TSP. Their result is a $(1+\alpha)$-approximation, where α is the factor of an approximation algorithm for TSP. Interestingly, the formulation of CVRP as a covering integer linear program has integrality gap at most 2.5 [TV01]. In the special case of Euclidean metrics, Das and Mathieu [DM10] found a quasi-polynomial approximation scheme for CVRP. Depending on the capacity bound C, it is possible to obtain a PTAS in the euclidean plane for some special cases (if the capacity is either small [ACL09], or very large [AKTT97]). For tree metrics, there is a 2-approximation [LLM91].

The Capacitated Vehicle Routing Problem with delay factor (CVRP-R) additionally considers a delay factor $R \in \mathbb{R}_+$ and requires every vertex v to be visited before $R \cdot d(s,v)$. There is a $(1+\alpha+\frac{3}{R-1})$-approximation algorithm for CVRP-R by Gørtz et al. [GNR11] that can be

	approximation	hardness
DVRP	2	1.5
DVRP-Σ	4	1.5
DVRP-D	2	NP-hard

Table 3.1 – Results for variants of DVRP on a tree metric (with unlimited capacity)

strengthened to a $(1+\frac{2}{R-1})\alpha$-approximation in case of unlimited capacity, where again α denotes the factor of an approximation algorithm for TSP. Typically, the CVRP-R asks to find tours as bus routes. We observe that CVRP-R is the variant of SBP-Σ-mult considering tours instead of walks. However, the results in [GNR11] also hold in case of walks and thus imply the same approximation results for SBP-Σ-mult.

The SBP is closely related (but not equal) to the Distance Constrained Vehicle Routing Problem (DVRP). DVRP enforces a bound D on the length of a vehicle tour, and the goal is the minimization of the number of tours used to visit all locations. It was raised and studied for applications in [LDN84] and [LSLD92]. Routing problems like the DVRP can directly be encoded as instances of Minimum Set Cover, and thus often admit logarithmic approximations. Nagarajan and Ravi [NR12] give a careful analysis of the set cover integer programming formulation of the DVRP and bound its integrality gap by $O(\log D)$ on general graphs and by $O(1)$ on a tree. They also obtain a constant approximation for the DVRP on a tree and a $O(\log D)$ approximation in general. We remark here that, despite the similarity of the two problems, a straightforward adaptation of their methods to the SBP does not work. Friggstad and Swamy [FS14] recently improved the approximation factor for DVRP to $O\left(\frac{\log D}{\log\log D}\right)$.

As for SBP, the roles of the objective and the length bound can be exchanged. DVRP-D denotes the variant that asks to cover all vertices by N tours whose maximum length is minimized. DVRP-D admits a $(2.5-\frac{1}{N})$-approximation algorithm [FHK76] and a $O(1)$-approximation algorithm in case a capacity bound is considered [GMNR11]. Since DVRP-D clearly generalizes TSP, it is NP-hard to approximate within a factor of $\frac{123}{122}$ [KLS13]. We will prove in Theorem 3.1.6 an approximation-preserving reduction from DVRP-D to DVRP. Together with the 2-approximation algorithm by Nagarajan and Ravi [NR12] for DVRP, this implies a 2-approximation algorithm for DVRP-D on tree metrics.
DVRP is also studied with respect to minimizing the total length of the tours instead of their number. We denote this variant by DVRP-Σ. Li et al. [LSLD92] show that DVRP and DVRP-Σ are closely related (cf. Lemma 3.1.5). Combining this result with the 2-approximation algorithm for DVRP [NR12], we obtain a 4-approximation algorithm for DVRP-Σ on trees.

In terms of hardness results for tree metrics, is has been observed in [NR12] that DVRP cannot be approximated within a factor < 1.5 unless P=NP by a reduction from the partition problem [GJ79]. A very similar construction yields that also DVRP-Σ is NP-hard to approximate to a

factor better than 1.5 (cf. SBP-Σ). DVRP-D is clearly NP-hard. We summarize the results for DVRP, DVRP-D and DVRP-Σ on tree metrics in table 3.1.

In the Minimum Latency problem (MLP, also known as Traveling Repairman problem), we are given an undirected graph with metric distances on the edges and a root vertex. The goal is to find a rooted walk that visits all vertices and minimizes the sum of arrival times. There is a 3.59-approximation algorithm [CGRT03] known. However, MLP admits a PTAS for special cases like Euclidean metrics or metrics induced by a tree or a planar graph [Sit14]. We note that MLP is NP-hard even on trees [Sit02]. The k-Traveling Repairman problem considers k walks that cover all vertices while minimizing the sum of all arrival times. We observe that this problem is equivalent to a variation of the School Bus problem in which we minimize the total sum of the regret at all vertices for a given number of buses. A combination of the results in [FHR07] and [CGRT03] yields a 8.49-approximation algorithm.

Our publications on the School Bus problem [BGKS11, BGKS13] inspired Friggstad and Swamy [FS14] to study this problem as well. They obtain a constant factor approximation algorithm for SBP in general graphs and they also bound the integrality of the natural set cover formulation to a small constant. In addition, they present a $O(N^2)$-approximation algorithm for SBP-R in case of unlimited capacity, where N denotes the given number of buses. For multiplicative regret, they give a $O(\log \frac{R}{R-1})$-approximation algorithm for SBP-mult and a $O(1)$-approximation algorithm for SBP-R-mult with unlimited capacity. Furthermore, they obtain approximation results for bus routing problems with asymmetric distance metrics.

Many practical problems involving school buses have been studied, but primarily within the context of *heuristic methods* for real-life instances. We refer to [PK10] for a thorough survey of possible formulations and heuristic solution methods. Our notion of regret was first introduced as a vehicle routing objective in [SBL05]. They considered a more general problem involving time windows for customers and applied metaheuristics to produce solutions to real-life instances.

In a further variant that generalizes both SBP and SBP-mult, we consider deadlines $D : V \to \mathbb{N}$ for all vertices. The task is then to find the minimum number of rooted walks that cover all vertices before their deadlines. There is a $O\left(\log \frac{R_{\max}}{R_{\min}}\right)$-approximation algorithm [FS14], where $R_{\max} = \max_v D(v) - d(v,s)$ and $R_{\min} = \min_v D(v) - d(v,s)$.

This variant can be seen as a special case of the Vehicle Routing with time windows (VRP-TW), where every node is associated with a time interval in which it must be visited [DDS92]. There is a $O(\log^3 |V|)$ approximation algorithm for VRP-TW based on a $O(\log^2 |V|)$-approximation for Orienteering with time windows [BBCM04].

3.1.2 Our contributions

The main focus of our work lies on tree metrics. If we restrict the input to this case, we can achieve constant factor approximation algorithms for SBP, SBP-Σ and SBP-R. Table 3.2 gives

an overview of our results on tree metrics with unlimited capacity. Specifically, we will give a

	approximation	hardness
SBP	3	1.5
SBP-Σ	3	1.5
SBP-R	13.5	NP-hard

Table 3.2 – Results for variants of SBP on a tree metric with unlimited capacity

simple combinatorial 4-approximation algorithm for SBP on trees. The approximation factor reduces to 3 in the case of unlimited capacity. Our algorithm for SBP also implies an integrality gap bound matching the approximation factor. Namely, we will show that the integrality gap of the natural set-cover formulation of the SBP on trees is at most 4. In case of unlimited capacity, the gap is at most 3. Furthermore, the same algorithm as for SBP yields a 3-approximation algorithm for SBP-Σ in case of unlimited capacity.

On the negative side, we prove in section 3.2.2 an inapproximability factor of 1.5 for the SBP on trees. Using a very similar reduction as for DVRP-Σ, we can show that SBP-Σ is also NP-hard to approximate within a factor of 1.5.

Furthermore, we give a combinatorial 13.5-approximation for the SBP-R on trees in the case of unlimited capacity.

	approximation	hardness
SBP	32	2
SBP-R	inapproximable	

Table 3.3 – Results for variants of SBP

As we will show, the SBP can be formulated as a set covering problem. With this observation, we will easily derive a $O(\log C)$-approximation to SBP in general graphs, as well as a $O(\log C)$ upper bound on the integrality gap of the natural set-cover formulation applied to the SBP. We note that in [FS14], a 32-approximation algorithm for SBP as well as an upper bound of 16 on the integrality gap of this formulation is shown. Further, we prove that unless $P = NP$, SBP-R is inapproximable for general metric graphs, since we show that it is NP-hard to check whether an instance of SBP-R has regret 0 or strictly positive. An overview of the known results for general graphs is presented in Table 3.3.

Finally, we give an overview of new and known results for the School Bus problem with multiplicative regret and we present an application of our approximation algorithm for Capacitated Orienteering to a generalized version of DVRP with node demands.

3.1.3 Preliminaries

Let us recall some definitions and notation. We denote by $d(s,v)$ the shortest path distance of a vertex $v \in V$ from the root s. For a given path P, we denote by V_P the nodes of P and by E_P its edges. With a slight abuse of notation, we shorten $d(E_P)$ to $d(P)$. Furthermore, we define $d^P(u,v)$ to be the distance along the path between two vertices $u, v \in V_P$. The segment of the path P (or subpath) between u and v is denoted by $P(u,v)$.

Given a complete graph $G = (V,E)$ with metric edge distances d, we say that d is a *tree metric* if there exists a spanning tree $T = (V,F)$ of G with the following property: $d(e) = d_T(u,v)$ for all $e = \{u,v\} \in E$ where $d_T(u,v)$ is the length of the unique u-v path in T. If the tree metric is induced by a star, then we also refer to it as a star metric. As pointed out in Section 2, we will simply consider rooted walks in the underlying tree graph instead of, equivalently, rooted paths in the complete graph obtained by taking the metric closure [KV12].

We first observe that the capacity bound can be neglected for a slight loss in the approximation factor for both the SBP and SBP-Σ. This fact was also observed in [HRK85, GNR11] for the similar problems CVRP and CVRP-R.

Lemma 3.1.1. *Given an α-approximation algorithm to the SBP-Σ with unlimited capacity for each bus, there is an $(\alpha+1)$-approximation to the SBP-Σ with capacity bound C.*

Proof. Given an instance of the SBP-Σ with capacity bound C, we first ignore the capacity bound and run the α-approximation algorithm for the resulting SBP-Σ instance with unlimited capacity. The output will be a set of walks $P_1, \dots, P_k$ covering vertices W. Note that by short cutting, we can assume that two walks P_j and P_i have only the root node s in common for all $i \neq j$ and that every P_j contains only vertices from W. The idea is to cut each walk into parts of capacity C and connect them directly to the root node s.
For $j = 1, \dots, k$ and each walk P_j, we index the vertices in the order in which they appear on P_j. Let $n_j := |V(P_j)|$. By adding dummy vertices on P_j at distance 0 from s, we can assume that n_j is a multiple of C. We partition the vertices into groups of C consecutive vertices and start a new walk from the root for each group. Starting with the first vertex, a group is formed with the following $C-1$ vertices. We now shift the indexing of the vertices by shifting the initial vertex along the walk P_j. Note that we consider the sequence of vertices on P_j to be cyclic, i.e we wrap around and start again from s whenever the other endpoint of P_j is reached. Hence we obtain n_j possibilities for grouping the vertices of P_j into portions of C vertices. For each partitioning into groups, we connect the first vertex of each group to the root s and then follow P_j. In case a group of vertices is formed by some vertices from the end and some from the beginning of P_j, we construct two walks to cover them: One connects s to the first vertex of the group and then follows P_j, and the second follows P_j from s until the last vertex of the group. The set of walks covering P_j with smallest total length among the n_j possibilities is output. We employ a simple averaging argument described in [HRK85] to bound the total length of the resulting walks.The total length of all walks in the n_j possibilities of grouping the

vertices is at most $\frac{n_j}{C} \sum_{v \in V(P_j) \cap W} d(s,v) + n_j \cdot d(P_j)$. The first term reflects that every vertex is first in a group for $\frac{n_j}{C}$ possibilities and the second term says that every edge of the walk P_j is used at most once in each possibility. Since we output the shortest among the n_j sets of walks covering the vertices on P_j, its total length is at most $\sum_{v \in V(P_j) \cap W} \frac{d(s,v)}{C} + d(P_j)$.

Summing up over j, we obtain a solution for SBP-Σ respecting the capacity bound C of total length at most

$$\sum_{j=1}^{k} d(P_j) + \sum_{j=1}^{k} \sum_{v \in V(P_j) \cap W} \frac{d(s,v)}{C} = \sum_{j=1}^{k} d(P_j) + \frac{1}{C} \sum_{v \in W} d(s,v)$$

Since the term $\frac{1}{C} \sum_{v \in W} d(s,v)$ is a well known lower bound for CVRP (called Radius lower bound [HRK85]), it is also a lower bound on the optimal value of SBP-Σ. Let APX_C denote the total length of the walks that we output in case of capacity C and let OPT_C, OPT_∞ denote the optimal values of the capacitated and uncapacitated case of SBP-Σ, respectively. Clearly, $OPT_\infty \leq OPT_C$ holds and thus we obtain

$$APX_C \leq \sum_{j=1}^{k} d(P_j) + \frac{1}{C} \sum_{v \in W} d(s,v) = \alpha OPT_\infty + OPT_C \leq (\alpha + 1) OPT_C.$$

□

The proof of the result for SBP is essentially identical to that of a similar statement proven in [NR12] for the DVRP, but we recall the proof for the sake of completeness.

Lemma 3.1.2. *Given an α-approximation algorithm to the SBP with unlimited capacity for each bus, there is an $(\alpha+1)$-approximation to the SBP with capacity bound C.*

Proof. Given an instance of the SBP with capacity bound C, we first ignore the capacity bound and run the α-approximation algorithm for the resulting SBP instance with unlimited capacity. The output will be a set of walks $P_1, \ldots, P_k$ covering all vertices W. Note that by short cutting, we can assume that two walks P_j and P_i have only the root node s in common for all $i \neq j$. The idea is to cut each walk into parts of capacity C and connect them directly to the root node s. Let APX_C denote the number of buses output in case of capacity C and OPT_C, OPT_∞ the optimal solutions of the capacitated and uncapacitated case, respectively. We obtain

$$APX_C \leq \sum_{i=1}^{k} \left(\frac{|\{w \in W : w \text{ is covered by } P_i\}|}{C} + 1 \right) \leq \frac{|W|}{C} + \alpha OPT_\infty \leq (1+\alpha) OPT_C.$$

□

If P is a walk ending at s and covering a subset S of nodes, then we say that P has regret R if a regret bound of R is respected for all children in S. The following useful fact holds for all variants of the School Bus problem with additive regret:

Proposition 3.1.3. *Let P be a walk starting at some node v and ending at s. For all nodes covered by P the regret bound R is respected if and only if it is respected for v.*

Proof. Assume for contradiction that there is a node $w \in S$ with $d^P(w,s) > d(w,s) + R$ while $d^P(v,s) \leq d(v,s) + R$. The triangle inequality then implies

$$d^P(v,s) \geq d^P(w,s) + d(v,w) > R + d(w,s) + d(v,w) \geq R + d(v,s),$$

a contradiction. □

We note that such a statement is not true in case of multiplicative regret.

Furthermore, a simple observation shows that a walk with too high regret can be split into a few walks of small regret.

Proposition 3.1.4. *Let P be a walk rooted at s with regret βR for some $\beta > 1$. Then the vertices $V(P)$ on P can be covered using $\lceil \beta \rceil$ walks rooted at s with regret at most R.*

Proof. Let t be the other endpoint of P. Intuitively, we traverse P starting from s and stop whenever the regret would exceed R and start a new walk. Let $u \to v$ denote the shortest connection, i.e. an edge, between u and v, whereas $u \rightsquigarrow v$ denotes the path between u and v along P. Further, let v_1 be the first vertex on P such that $d^P(s,v_1) > d(s,v_1) + R$ and by u_1 the immediate predecessor of v_1 on P. By construction, $d^P(s,u_1) - d(s,u_1) \leq R$ holds. Analogously, we find vertices $s = v_0, v_1, \ldots, v_k$ on the walk P such that the walk $s \to v_{i-1} \rightsquigarrow v_i$ has regret $> R$ for all $i = 1, \ldots, k$. For the immediate predecessor u_i of v_i on P, we obtain a rooted walk $s \to v_{i-1} \rightsquigarrow u_i$ with regret at most R for all $i = 1, \ldots, k$. It remains a last piece between v_k and t that is used to form the walk $s \to v_k \rightsquigarrow t$ with regret $\leq R$. It follows from the additivity of the regret that k is less than β:

$$\begin{aligned} \beta R = d^P(s,t) - d(s,t) &= d^P(v_k,t) - d(s,t) + \sum_{i=1}^{k} d^P(v_{i-1},v_i) \\ &\geq \sum_{i=1}^{k} d(s,v_{i-1}) + d^P(v_{i-1},v_i) - d(s,v_i) \\ &> kR \end{aligned}$$

The second inequality follows by construction and the first from $d(s,t) \leq d(s,v_k) + d^P(v_k,t)$. We conclude that we obtain $k+1 \leq \lceil \beta \rceil$ walks that cover all vertices of P. □

This proposition was generalized in [FS14] as follows: The set of vertices on any k walks ending at s with total regret $\beta \cdot kR$ can be covered using at most $(\beta+1)k$ walks ending at s with regret at most R.

3.1.4 Relations of different objectives

We first state a well known result for DVRP.

Lemma 3.1.5. *[LSLD92] Let $N_\sharp, L_\sharp$ denote the number of tours and the total sum of the tour lengths, respectively, in case of DVRP. Let N_Σ, L_Σ denote the number of tours and the sum of the tour lengths for DVRP-Σ. We have*

$$\frac{1}{2} N_* D \leq L_* \leq N_* D$$

for $* = \Sigma, \sharp$. It follows $N_\Sigma \leq 2N_\sharp$ and $L_\sharp \leq 2L_\Sigma$.

We note that a similar relation between SBP and SBP-Σ is not true. Simple examples show that an optimal solution to one problem can be arbitrarily bad with respect to the objective of the other problem.
Next, we observe a simple relation between DVRP and DVRP-D.

Theorem 3.1.6. *An α-approximation algorithm for the DVRP implies an $\lceil\alpha\rceil$-approximation algorithm to DVRP-D.*

Proof. Let N denote the number of tours prescribed in the DVRP-D instance and let D^* denote its optimal value. Using binary search, we find in polynomial time the minimum length bound $D \leq D^*$ such that the α-approximation algorithm for DVRP outputs at most αN tours. From those tours, we concatenate every $\lceil\alpha\rceil$ tours at the root to a single tour. In that way, we obtain N tours of maximum length $\lceil\alpha\rceil D \leq \lceil\alpha\rceil D^*$. □

Finally, we give a approximation preserving reduction from SBP-R to SBP.

Theorem 3.1.7. *An α-approximation algorithm for the SBP-R implies an $\lceil\alpha\rceil$-approximation algorithm for SBP.*

Proof. Let R denote the regret bound of a given SBP instance with n vertices and let N^* be its optimal value.
Suppose we are given an α-approximation algorithm for SBP-R. We compute the minimum number N' of bus routes such that the maximum regret is bounded by $\alpha \cdot R$ by trying all n possible choices of the parameter N for SBP-R. We observe that $N' \leq N^*$ holds because the approximation algorithm for SBP-R finds N^* routes with maximum regret at most αR.

Using Proposition 3.1.4, we obtain at most $\lceil\alpha\rceil N' \leq \lceil\alpha\rceil N^*$ routes of regret R and thus a $\lceil\alpha\rceil$-approximation algorithm for SBP. □

3.2 The School Bus problem

3.2.1 General graphs

We present a covering integer linear programming formulation of the SBP. Let $\mathscr{S}$ be the family of all sets of C or fewer children that can be covered by a single walk ending at s having regret at most R. We introduce a variable x_S for each $S \in \mathscr{S}$ and give the following formulation:

$$\begin{aligned} \min \quad & \sum_{S \in \mathscr{S}} x_S && \text{(IP)} \\ \text{s.t.} \quad & \sum_{S : w \in S} x_S \geq 1 \quad \forall w \in W \\ & x_S \in \{0,1\} \quad \forall S \in \mathscr{S}. \end{aligned}$$

The greedy strategy for Set Cover implies a $O(\log C)$-approximation to SBP using a constant factor approximation algorithm for capacitated Orienteering (cf. Section 2.2) as a subroutine. The technique of Dual-Fitting for Set Cover (cf. [Vaz01]) lets us bound the integrality gap of the LP relaxation of (IP) by $O(\log C)$. However, we omit the details of these results, since the analysis is pretty standard and a better bound of 16 was described in [FS14] for the integrality gap as well as a 32-approximation algorithms based on this formulation. There is a lower bound of 2 on the integrality gap already on star graphs.

On the negative side, we observe the following inapproximability result that was independently noted in [FS14].

Theorem 3.2.1. *There is no approximation algorithm for SBP with a factor less than 2 unless P=NP.*

Proof. We provide a simple reduction from the following NP-hard decision problem of a variant of TSP [GJ79]: Given a graph G and an integer L, is there a Hamilton path including all vertices of length at most L?

We add a new root vertex s at distance L from every vertex in G and we set the regret bound $R = L$. We observe that the optimal solution of SBP in the new graph has one bus route if and only if there is a Hamilton path in G of length at most L. Note that any α-approximation algorithm with $\alpha < 2$ would thus allow us to decide the Hamilton path problem. □

3.2.2 Tree metrics

For the SBP on trees, we first present a 4-approximation algorithm and from that, we derive a 4-approximation for SBP-Σ on trees. Afterwards, we show that the integrality gap of the natural set cover formulation for SBP on trees can be bounded by 4 as well. Finally, we prove that both SBP and SBP-Σ on trees are hard to approximate within a factor 1.5.

In this section, we will mainly focus on the infinite capacity version of the SBP and SBP-R on a

tree T with root s. We denote by $P(u,v)$ and $d(u,v)$ the unique path from u to v in T, and its corresponding length. For a subset of edges $F \subseteq E_T$, we let $d(F) := \sum_{e \in F} d(e)$. For a connected set of edges, we call a walk that visits each edge exactly twice an *Euler tour*. In the following, a subtree of T denotes a tree consisting of a node v of T and all vertices w such that the path between w and s contains v. We also say that such a subtree is rooted at v.
We note that subtrees of T that contain no vertices in W will never be visited by a bus in any optimal solution, and thus we can assume without loss of generality that all leaves of T represent the location of children. In such an instance, a feasible solution will simply cover all of T with bus routes. Moreover, when assuming infinite capacity, any solution is still feasible if every node of T represents a child location and we can therefore assume, without loss of generality, that $W = V$.

A 4-approximation for SBP

Our main result for SBP on tree metrics is the following.

Theorem 3.2.2. *There exists a polynomial time 4-approximation for the SBP on trees. The approximation factor reduces to 3 in the case of unlimited capacity.*

We prove Theorem 3.2.2 by first giving a combinatorial 3-approximation for the SBP with unlimited capacity on graphs that are trees, and subsequently applying Lemma 3.1.2. Our algorithm is based on the following intuitive observations:

- When the input tree is very short (say, of height at most $\frac{R}{2}$ on an instance with regret R), then it is relatively easy to obtain a 2-approximation for the SBP by simply cutting an Euler tour of the tree into short pieces and assigning each piece to a bus.
- General trees can be partitioned into smaller pieces (subtrees) such that at least one bus is required for each piece, but each piece can be solved almost optimally via a similar Euler tour method.

We begin with some definitions. We call a set of vertices $\{a_1, \ldots, a_m\} \subseteq V$ *R-independent* if for all $a_i \neq a_j$, we have $d(a_i, \mathrm{lca}(a_i, a_j)) > \frac{R}{2}$, where $\mathrm{lca}(a_i, a_j)$ is the lowest common ancestor of the vertices a_i and a_j in T. By iteratively marking the leaf in T furthest from the root such that R-independence is maintained among marked leaves, we can obtain, in polynomial time, an inclusion-wise maximal R-independent set of leaves A such that all vertices in T are within a distance of $\frac{R}{2}$ from a path $P(s,a)$ for some $a \in A$. We shall call A a set of *anchors*. By construction, no two distinct anchors a_i and a_j can both be covered by a walk of regret at most R, immediately yielding the following lower bound:

Proposition 3.2.3. *The size $|A|$ of the set of anchors is a lower bound on the number of buses that is needed in any feasible solution.*

We now give a second useful lower bound. Let $Q := \bigcup_{a \in A} P(s,a)$. We call Q the *skeleton* of T, noting that Q is a subtree of T whose leaves are the anchors. Observe that all edges in the skeleton Q will automatically be covered if a bus visits each anchor. Since each anchor must be visited at least once, it suffices to only consider covering the anchors and the *non-skeletal edges* of T, i.e. the edges in $T \setminus Q$. The edges in $T \setminus Q$ form a collection of disjoint subtrees, each of which has height at most $\frac{R}{2}$. We call these *short subtrees*. See figure 3.1 for an example tree visualizing these definitions.

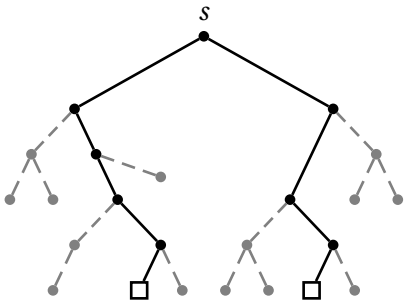


Figure 3.1 – Example of a tree with unit distance edges and $R = 6$. A maximal set of anchors is drawn as square nodes, the corresponding skeleton is shown in solid and the grey, dashed parts of the tree are the short subtrees.

Suppose that a feasible walk starts at a vertex v in a short subtree $\mathcal{T}$. It will cover all the edges in $P(s,v)$, and may possibly cover some additional detour edges having total length at most $\frac{R}{2}$. Since $\mathcal{T}$ is a short subtree, the non-skeletal edges in $P(s,v)$ have total length at most $\frac{R}{2}$. It follows that:

Observation 1. *The set of non-skeletal edges covered by any feasible walk P must have total length at most R: at most $\frac{R}{2}$ in length along the path from its starting vertex to the root, and at most $\frac{R}{2}$ length in edges covered by detours.*

From this, we can observe the following lower bound on the number of buses:

Proposition 3.2.4. *The number $\frac{1}{R}\sum_{e \in T \setminus Q} d(e)$ is a lower bound on the number of buses that are needed in any feasible solution.*

We build our 3-approximation from these two lower bounds by partitioning the edges of T into a family of subtrees each containing a single anchor, and approximating the optimal solution well on each of these subtrees. For anchors $A = \{a_1, \ldots, a_m\}$, we define associated paths of edges $\{P_1, \ldots, P_m\}$ as follows: (i) $P_1 = P(s, a_1)$, and (ii) $P_i = P(s, a_i) \setminus \left(\bigcup_{j=1}^{i-1} \{P_j\} \right)$ for $2 \leq i \leq m$. The edges in $\{P_1, \ldots, P_m\}$ form a partition of the skeleton Q into paths (not necessarily ending at s), each of which starts at a different anchor.

We then let T_i be the set of all edges in both the path P_i and the set of all short subtrees attached to P_i. If a short subtree is attached to a junction point where two paths P_i and P_j

meet, we arbitrarily assign it to either P_i or P_j so that the sets $\{T_1, \ldots, T_m\}$ form a partition of all of the edges of T into a collection of subtrees, each containing a single anchor.

For each $1 \leq i \leq m$ we define a directed walk W_i that starts at the anchor a_i, proceeds along P_i in the direction toward the root s, and collects every edge in T_i by tracing out an Euler tour around each of the short subtrees in T_i that are attached to P_i. One may easily verify that it is always possible to quickly find such a walk such that the following properties are satisfied:

- W_i contains each edge in P_i exactly once and always proceeds in the direction toward s when collecting each edge in P_i.
- W_i contains each edge in $T_i \setminus P_i$ exactly twice: once proceeding in the direction away from s, and once in the direction toward s.

We now greedily assign the edges in the short subtrees in T_i to buses by simply adding edges to buses in the order in which they are visited by W_i. We first initialize a bus β_1 at the anchor a_i and have it travel along W_i until the total length of all of the edges it has traversed in the *downward* direction (away from the root s) is exactly $\frac{R}{2}$. At this point, we assume it lies on some vertex v_1 (if not, we may imagine adding v_1 to the middle of an existing edge in T_i, although this will not be relevant to our solution as there are then no children at v_1). We send bus β_1 from v_1 immediately back to the root s and create a new bus β_2 that starts at v_1 and continues to follow W_i until it too has traversed exactly $\frac{R}{2}$ length in edges of T_i in the downward direction. We assume it then lies at a vertex v_2, create a new bus β_3 that starts at v_2 and continues to follow W_i, and so on. Eventually, some bus β_k will pick up the last remaining children and proceed to the root s, possibly with leftover detour to spare. We observe that the number of buses used is exactly $\left\lceil \frac{2\sum_{e \in T_i \setminus P_i} d(e)}{R} \right\rceil$, since each bus other than the last one consumes exactly $\frac{R}{2}$ of the downward directed edges in W_i, and W_i proceeds downward along each edge in $T_i \setminus P_i$ exactly once. We also note that this is a feasible solution since a bus travelling a total downward direction of $\frac{R}{2}$ must make a detour no greater than R.

Doing this for each edge set T_i yields a feasible solution to the original instance using exactly $\sum_{i=1}^{m} \left\lceil \frac{2\sum_{e \in T_i \setminus P_i} d(e)}{R} \right\rceil$ buses. This is at most

$$m + \frac{2}{R} \sum_{i=1}^{m} \sum_{e \in T_i \setminus P_i} d(e) = m + 2\frac{\sum_{e \in T \setminus Q} d(e)}{R} \leq 3OPT$$

by Proposition 3.2.3 and Proposition 3.2.4, where OPT is the optimal number of buses required in any feasible solution. Together with Lemma 3.1.2, this proves Theorem 3.2.2.

Interestingly, the same algorithm also gives a 3-approximation algorithm for SBP-Σ in case of unlimited capacity.

Theorem 3.2.5. *There is a 3-approximation algorithm for SBP-Σ with unlimited capacity on tree metrics.*

Proof. Assuming unlimited capacity for the buses, we consider again the previous algorithm that gives a 3-approximation algorithm for SBP. We first observe that the total sum of edge distances in the tree clearly is a lower bound on the optimal value of SBP-Σ. In order to argue that the total length of the computed solution is at most three times the optimal value, we show that every edge is used at most three times as often as in an optimal solution. Recall that the algorithms finds first a maximal set of anchors and then covers the short subtrees gradually.

Let $\{u, v\}$ denote an edge in the tree and let w.l.o.g. v be the vertex closer to the root. We distinguish two cases.

- If $\{u, v\}$ is an edge of the skeleton, then it follows as in the proof of Theorem 3.2.2 that the algorithm covers the subtree rooted at u using at most three times the minimum number of rooted paths needed to do so. We observe that each of those paths traverses the edge $\{u, v\}$ exactly once.
- If $\{u, v\}$ is an edge in a short subtree, then there is no anchor in the subtree rooted at u. Let k denote the minimum number of rooted paths needed to cover the subtree rooted at u with paths of regret $\leq R$. It follows from observation 1 that the algorithm covers the subtree rooted at u using at most $2k$ such paths. However, exactly one of those paths traverses the edge $\{u, v\}$ twice, all other paths traverse $\{u, v\}$ once. This means that $\{u, v\}$ is used at most $2k + 1 \leq 3k$ times.

Finally, we note that any solution of SBP-Σ has to use at least the minimum number of rooted paths to cover the subtree rooted at u. This concludes the proof of the theorem. □

Together with Lemma 3.1.1, this implies a 4-approximation algorithm for the capacitated case.

Corollary 3.2.6. *There is a 4-approximation algorithm for SBP-Σ on tree metrics.*

Integrality gap of set-cover formulation

We will next show that the bounds given in Propositions 3.2.3 and 3.2.4 are necessarily also respected by fractional solutions to the LP relaxation (LP) of (IP). Together with the argument above, this immediately implies that (LP) has an integrality gap of at most 4 (and 3 in the case of infinite capacities).

Theorem 3.2.7. *The integrality gap of the natural set-cover formulation of the SBP on trees is at most 4. In case of unlimited capacity, the gap is at most 3.*

Proof. The dual of (LP) is a packing problem with a variable y_v for each $v \in V$; we think of y_v as the *profit* of child v. The dual then has an exponential number of constraints bounding the

total profit of each feasible set of children that can be picked up in a single walk.

$$\begin{aligned} \max \quad & \sum_{v \in V \setminus \{s\}} y_v \\ \text{s.t.} \quad & \sum_{v \in S} y_v \leq 1 \quad \forall \text{ feasible sets of children } S \\ & y_v \geq 0 \quad \forall v \in V \setminus \{s\}. \end{aligned} \tag{D}$$

To prove our bound, we need to state the following lemma proven in [NR12]:

Lemma 3.2.8. [Distribution Lemma] *For any tree H with root s and distance function d on the edges, it is possible to distribute a total profit of K among the leaves of H such that the profit contained in any connected subgraph F of T containing s is at most $\frac{d(F)}{d(H)} \cdot K$.*

There are three things to prove for a feasible fractional solution to the LP relaxation of (IP):

i) The number $\frac{|V|}{C}$ is a lower bound on the value of any fractional solution.
This lower bound is trivial. We assign a profit of $\frac{1}{C}$ to every node $v \in V \setminus \{s\}$. Any walk that collects profit > 1 has to visit more than C nodes.

ii) The size $|A|$ of the set of anchors is a lower bound on the value of any fractional solution.
The profit function that assigns a profit 1 to every anchor and 0 to all other vertices is a feasible solution to (D), since a feasible walk can never visit more than one anchor and thus collects profit at most 1.

iii) The lower bound of Proposition 3.2.4 holds for all fractional solutions.
In order to show this, we apply the distribution lemma to every short subtree H from the collection $\mathcal{H}$ of short subtrees that form $T \setminus Q$. On each subtree H, an amount of $d(H)/R$ is distributed among its leaves. Every other vertex gets profit 0. Therefore we distribute in total a profit of $\sum_{e \in T \setminus Q} d(e)/R$ over the vertices of T.
It remains to prove the feasibility of this dual solution. Consider a feasible walk P in T and assume that it collects a profit > 1. P visits the following length among edges of short subtrees:

$$\sum_{e \in (T \setminus Q) \cap P} d(e) = \sum_{H \in \mathcal{H}} \frac{\sum_{e \in H \cap P} d(e)}{d(H)} d(H) = \sum_{H \in \mathcal{H}} \frac{\sum_{e \in H \cap P} d(e)}{d(H)} \text{profit}(H) \cdot R$$

By the distribution lemma, we know that

$$\sum_{H \in \mathcal{H}} \frac{\sum_{e \in H \cap P} d(e)}{d(H)} \text{profit}(H) \cdot R \geq \text{profit}(P) \cdot R > R$$

This is a contradiction to Observation 1, since we would have length more than R of non-skeletal edges in P.

Now we bring the three lower bounds together to obtain the claimed result. Let APX denote the feasible integer solution that we obtain from the 3-approximation algorithm in the proof of Theorem 3.2.2 combined with Lemma 3.1.2. Combining their proofs, we have:

$$APX \leq \frac{|V|}{C} + |A| + 2\frac{\sum_{e \in T \setminus Q} d(e)}{R} \leq 4 \cdot OPT_f.$$

Note that in case of unlimited capacity, the term $\frac{|V|}{C}$ is omitted and we obtain an integrality gap of at most 3. □

The worst example that we are aware of has integrality gap 2. It consists of a star with $n+1$ nodes and edges of unit distance from the center. Set $R := 2(n-2)$ and the capacity $C = \infty$. The best integer solution uses exactly two routes while the fractional solution considers n routes (each possible route that skips exactly one leaf) with $\frac{1}{n-1}$ fraction. This yields a fractional solution close to 1 for n big enough and thus the claimed result.

Hardness results

Theorem 3.2.9. *The SBP is NP-hard to approximate within a factor better than* $\frac{3}{2}$.

Proof. We reduce partition to SBP. In an instance $\mathscr{I}$ of the partition problem, we are given a collection $\{a_1, \ldots, a_m\}$ of m non-negative integers with $\sum_{i=1}^{m} a_i$ even and $B := \frac{1}{2}\sum_{i=1}^{m} a_i$. The goal is to determine whether there exists a subset $S \subseteq [m]$ such that $\sum_{i \in S} a_i = B$. This problem is known to be NP-complete [GJ79].

To complete our inapproximability proof, we construct an instance of SBP on trees as follows. The root s has 2 children c_1, c_2 and m additional children with labels 1 to m. The edges (s, c_1) and (s, c_2) have length $2B$, while for each $j = 1, \ldots, m$, edge (s, j) has length a_j. The regret bound is $R := 2B$. Note that the size of the SBP instance is polynomial in the size of $\mathscr{I}$, and the construction runs in polynomial time.

Suppose $\mathscr{I}$ is a yes-instance; that is, there is some subset $S \subseteq [m]$ with $\sum_{j \in S} a_j = B$. Then we can construct one walk starting at c_1 and ending at s visiting vertices $\{j : j \in S\}$, and one walk starting at c_2 and ending at s visiting vertices $\{j : j \in [m] \setminus S\}$). Note that the regret of each walk is exactly $2B$.

Conversely, suppose $\mathscr{I}$ is a no-instance. Then we claim that the optimal value of the SBP instance is at least 3. Due to the length of the edges (s, c_1) and (s, c_2), the nodes c_1 and c_2 must be the starting nodes of two different walks, since otherwise we will exceed the regret bound. On the other hand, each of these 2 walks can only cover a subset $S \subseteq [m]$ with $\sum_{j \in S} a_j \leq B$, otherwise the regret bound will be exceeded. Since $\sum_{i=1}^{m} a_i = 2B$, but there is no valid walk covering a subset $S \subseteq [m]$ such that $\sum_{j \in S} a_j = B$, we require at least 3 walks to cover all nodes. The result follows.

□

We note that the NP-hardness of SBP-Σ on trees of height 2 can be shown by a reduction from bin packing. However, we omit the details and present instead a stronger inapproximability result for SBP-Σ similar to the one for SBP.

Theorem 3.2.10. *The SBP-Σ is NP-hard to approximate within a factor better than* $\frac{3}{2}$.

Proof. The reduction is again from the NP-complete partition problem [GJ79]: Given m non-negative integers $\{a_1, \ldots, a_m\}$ with even total sum $\sum_{i=1}^{m} a_i$ and let $B := \frac{1}{2}\sum_{i=1}^{m} a_i$. The goal is to determine whether there exists a subset $S \subseteq [m]$ such that $\sum_{i \in S} a_i = B$. Let $\mathscr{I}$ denote such an instance of the partition problem.

We construct an instance of SBP-Σ as follows. The root s has 1 child c at distance L for a parameter $L \gg B$ whose precise value is determined later. The node c has two children t_1, t_2 at distance $B+1$ and m additional children connected using edges (c, j) of length a_j for $j = 1, \ldots, m$. The regret bound is $R := 2B$. Note that the size of the SBP-Σ instance is polynomial in the size of $\mathscr{I}$, and the construction runs in polynomial time.

Suppose $\mathscr{I}$ is a yes-instance; that is, there is some subset $S \subseteq [m]$ with $\sum_{j \in S} a_j = B$. Then we can construct one walk starting at t_1 and ending at s visiting vertices $\{j : j \in S\}$, and one walk starting at t_2 and ending at s visiting vertices $\{j : j \in [m] \setminus S\}$. Note that the regret of each walk is exactly $2B$. The total length of each walk is $L + B + 1 + 2B$. So the optimal value of SBP-Σ is (at most) $2L + 6B + 2$.

Conversely, suppose $\mathscr{I}$ is a no-instance. We first observe that the optimal solution of the SBP-Σ instance uses at most 3 walks, since the extra cost outbalances the possible saving ($a_i < L$ holds for all $i = 1, \ldots, m$). Next, we claim that the optimal value is at least $3L + 5B + 2$. Due to the length of the edges (c, t_1) and (c, t_2), the nodes t_1 and t_2 must be the starting nodes of two different walks, since otherwise we will exceed the regret bound. On the other hand, each of these 2 walks can only cover a subset $S \subseteq [m]$ with $\sum_{j \in S} a_j \leq B$, otherwise the regret bound will be exceeded. Since $\sum_{i=1}^{m} a_i = 2B$, but there is no valid walk covering a subset $S \subseteq [m]$ such that $\sum_{j \in S} a_j = B$, we require at least 3 walks to cover all nodes. The third walk starts at a node $j \in [m]$ at distance at most B from c. Thus the total length of the three walks is at least $3L + 5B + 2$.

Suppose there is an α-approximation algorithm for SBP-Σ with $\alpha < 1.5$. Then we choose the parameter L big enough, namely $L > \frac{(6\alpha-5)B+(\alpha-1)2}{3-2\alpha}$, and thus we are able to decide the partition instance $\mathscr{I}$ using this algorithm. The result follows. □

3.3 The School Bus problem with regret minimization

In the School Bus problem with regret minimization (SBP-R), we are given a complete undirected graph $G = (V, E)$ with general metric distances $d : E \to \mathbb{N}$, a node s representing the school, and a set $W \subseteq V$ representing children's locations. Additionally, we are given a bus capacity $C \in \mathbb{N}$, and a number $N \in \mathbb{N}$ of available buses. We want to select N paths ending at s (that correspond to bus routes) such that each child is covered by at least one path and the total number of children covered by each path is at most C. The goal is to minimize the *regret* value R, defined as follows.

For a given set of paths covering all children, denote by P^w the path covering a child w and by $d^{P^w}(w, s)$ the distance between w and s along the path P^w. Then

$$R := \max_{w \in W} \left\{ d^{P^w}(w, s) - d(w, s) \right\}.$$

We first describe a 13.5-approximation for SBP-R on trees with unlimited capacity. We show that SBP-R cannot be approximated in general graphs since it is NP-hard to decide whether the optimal regret value is zero or strictly positive.

3.3.1 Tree metrics

First, SBP-R is considered on tree metrics. Recall that SBP-R differs from SBP because of the exchanged roles of maximum regret and number of bus routes. In case of SBP-R, the number of routes is bounded by a given parameter $N \in \mathbb{N}$ while the maximum regret is to be minimized. We present here a polynomial time 13.5-approximation algorithm for SBP-R on tree metrics in case of unlimited capacity. The notation and definitions introduced in Section 3.2.2 are also employed in the following.

Without loss of generality, we may assume the tree T to be binary. Suppose we can fix a value R for the regret. We will develop an algorithm that, given an instance and the value R, either outputs a set of at most N bus routes, with a maximum regret of $13.5R$, or asserts that every solution with at most N buses must have a regret value $> R$. Then, we can do binary search on the regret values and output the best solution found.

Suppose we have guessed a value for R. We begin by finding a set of anchors A with respect to R as described in section 3.2.2. Our algorithm shall only generate solutions where each bus begins its journey at an anchor. As it turns out, this restriction will be very helpful and causes only a small loss in the approximation factor that we obtain. By assigning a bus to start at each of the anchors, we can ensure that the anchors and skeleton will be covered. However, this time, covering the short subtrees (as defined for SBP in section 3.2.2) is much trickier because the number of buses we may use is limited. In fact, it may be the case that all feasible solutions using N buses require some bus to travel from an anchor, up one part of the skeleton, and then down another part of the skeleton some way towards a different anchor, before returning

upward to travel to the root. Consequently, greedy assignments of short subtrees to buses using methods like those used in section 3.2.2 do not work for SBP-R.

We introduce a few additional definitions to help us describe how we get around this difficulty. Every vertex v of the skeleton is called a *junction point* if it is either the root s or if v has degree more than 2 in Q. Let J be the set of junction points. The skeleton Q can be split at its junction points into a set of edge-disjoint paths, which we will call *core segments*. Formally, a path in Q is a core segment if and only if its endpoints are anchors or junction points and it contains no junction points in its interior. See figure 3.2 for an example tree visualizing these definitions.

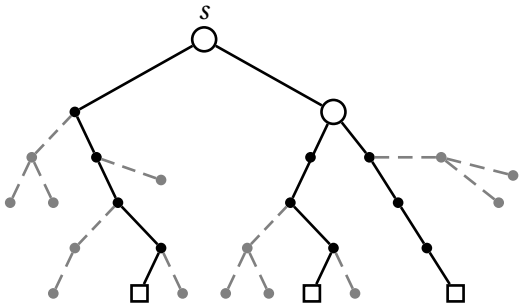


Figure 3.2 – Example of a tree with unit distance edges and $R = 6$. Anchors are drawn as square nodes, junction points as empty circles.

Now, we employ the fact that, in a solution of regret R using N buses, no bus starting at some anchor $a \in A$ may travel too far off from the path from a to the root s; in particular, it may not travel a distance of more than $\frac{R}{2}$ along core segments not contained in the path from a to s. As intuition for how our algorithm exploits this, imagine taking each short subtree in T, detaching it from the skeleton Q, and then reattaching it to Q at some point $\frac{R}{2}$ closer to the root s from its original point of attachment in T. In the resultant tree T', a short subtree $\mathscr{T}'$ will be attached to the skeleton Q along the path from an anchor a to the root s if and only if the root of the corresponding short subtree $\mathscr{T}$ in T could be reached by a bus starting at a having regret at most R. Accordingly, by relocating all of the short subtrees, we effectively get rid of the need to consider buses that travel downward along portions of the skeleton. We will show that this relocation results only in a small loss in the approximation guarantee.

However, we must also ensure that the short subtrees themselves can be collected efficiently by a small number of buses. To do this, we cluster and cut the short subtrees into suitable pieces (called *tickets*), each of which can be collected efficiently by a bus starting at some particular anchor. By sizing the tickets appropriately, we also obtain a lower bound on the number of buses with regret R that are required to cover all of the points.

The next lemma shows how we obtain suitable tickets from a collection of short subtrees.

Lemma 3.3.1. *There exists a polynomial-time algorithm that, given a rooted path P in the skeleton Q, and a collection $\mathscr{C}$ of short subtrees whose roots lie on P, produces a partition of the edges of $\mathscr{C}$ into tickets $E_1^{\mathscr{C}}, \ldots, E_k^{\mathscr{C}}$ with $k \leq \left\lfloor \frac{\sum_{\mathscr{T} \in \mathscr{C}} d(\mathscr{T})}{R} \right\rfloor$ and an overhead ticket $E_0^{\mathscr{C}}$ such that:*

(P1) All of the edges in $E_0^{\mathscr{C}}$ can be collected with an additional regret $\leq 2.5R$ by a single bus whose route contains P.

(P2) For all $1 \leq i \leq k$, all the edges in $E_i^{\mathscr{C}}$ can be collected with an additional regret at most $3R$ by a single bus whose route contains P.

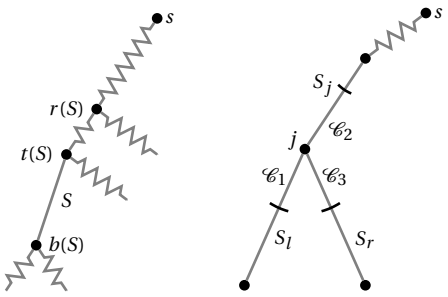
Proof. First, we identify the roots of all short subtrees with the lowest node v of path P. The idea is to cut the Euler tour of all short subtrees in $\mathscr{C}$ into suitable pieces. Starting from v, cut it at the first node such that the current piece has length $> 2R$. Continue like this to obtain $k+1$ walks. Denote by $E_1^{\mathscr{C}}, \ldots, E_k^{\mathscr{C}}$ all but the last piece. The last part of the Euler tour defines the overhead ticket $E_0^{\mathscr{C}}$. Note that $k \leq \left\lfloor \frac{\sum_{\mathscr{T} \in \mathscr{C}} d(\mathscr{T})}{R} \right\rfloor$, and that every cutting point is endpoint of a ticket and starting point of another ticket, thus it is covered by exactly two tickets. Since every node needs to be covered only once in a solution, we can ignore the last edge in each set $E_i^{\mathscr{C}}$ $(1 \leq i \leq k)$. By construction, the resulting length is at most $2R$. Both the starting and the end point of each ticket $E_i^{\mathscr{C}}$ $(1 \leq i \leq k)$ are at distance at most $\frac{R}{2}$ from v (cf. the definition of a short subtree). Since the Euler tour goes back to v after one subtree is finished, we obtain from a ticket a set of tours of total length at most $3R$. Note that we can arrange those tours in bottom-up order of the roots of visited short subtrees on the skeleton. A bus whose root walk contains P can then cover all edges corresponding to a ticket with regret at most $3R$. The last piece $E_0^{\mathscr{C}}$ of the Euler tour remaining from the cutting procedure certainly has length $\leq 2R$. Since the Euler tour goes back to v, only the distance to the starting point of the last piece has to be connected to v in order to obtain a set of tours. As in the previous case, one bus can cover these tours with regret $2.5R$, since the height of each subtree is at most $\frac{R}{2}$.
This strategy fulfills the properties (P1) and (P2) and runs in polynomial time. □

The next simple lemma will be useful later when we assign the overhead tickets to buses.

Lemma 3.3.2. *One can find a mapping $\phi : J \longrightarrow A$ from junction points to anchors in polynomial time with the following properties:*
(i) Every junction point $j \in J$ is mapped to an anchor $\phi(j)$ down in the subtree rooted at j.
(ii) For all $a \in A$, there is at most one junction point $j \in J$ with $\phi(j) = a$.

Proof. We construct ϕ by iteratively considering anchors $A = \{a_1, \ldots, a_m\}$ and building the skeleton Q using paths from anchors to junction points. We begin with the path $P(s, a_1)$ and set $\phi(s) = a_1$. When each new path $P(s, a_i)$ is added, a new junction point j_i is formed, namely the lowest intersection point of $P(s, a_i)$ with the previous paths. We set $\phi(j_i) = a_i$ for the remaining i, and we easily see that the resulting mapping ϕ satisfies properties (i) and (ii). □

Now, for our final algorithm, we don't actually detach, move, and reattach all of the short subtrees in T. Instead, we employ a bit more care and cleverness to combine this step with the ticketing process. For every core segment S, let $t(S)$ and $b(S)$ be the top and the bottom junction points in S. Furthermore, let $r(S)$ be the highest junction point at distance at most



(a) notation (b) situation in step 4

Figure 3.3 – Illustrations for Algorithm 4

$R/2$ from $t(S)$ (see figure 3.3a). The junction point $r(S)$ represents the common ancestor of all anchors from which a bus could reach S without exceeding the regret R. Our algorithm essentially deals with all short subtrees on S by converting them to tickets, and then 'placing' all of these tickets on $r(S)$, where they will be collected by buses travelling through the junction $r(S)$.

Algorithm 4 describes this procedure formally.

Let B be the number of bus routes output by the algorithm. We show:
(i) the regret of each route output by the algorithm is at most $13.5 \cdot R$ and
(ii) B is a lower bound on the number of buses with regret at most R that are needed to cover all nodes.

Lemma 3.3.3. *Algorithm 4 outputs a set of buses with maximum regret* $13.5R$.

Proof. Any bus starting at an anchor a collects at most 4 different regret amounts:

- at most one non-overhead ticket in step 5. Suppose that the ticket corresponds to short subtrees with roots on core segment S. We claim that these roots are at distance at most R from $P(s,a)$. To see this, note that the ticket might have been placed at $r(S)$ in step 3(f) of the algorithm. In this case, $P(s,a)$ is by definition at distance at most $R/2$ from the top end point $t(S)$ of segment S, and from there the short subtrees hanging off S can be reached within an additional distance of at most $R/2$. The claim follows and, together with property $(P2)$ from lemma 3.3.1, we can cover a ticket with a walk of regret $\leq 5R$.
- at most two remaining pieces of the Euler tour in step 3(e). There is at most one junction point j with $\phi(j) = a$ by (ii) of lemma 3.3.2. For a junction point j, each part E_0 and F_0 can be covered with regret $\leq 2.5R$ by (P2). Note that an additional regret R can be necessary to get to the roots of the short subtrees in either E_0 or F_0 that do not lie on the path from a to the root.
- at most one remaining piece of the Euler tour at a assigned in step 4(b). This ticket K_0 can be covered with regret $\leq 2.5R$ by (P2).

Algorithm 4 13.5-approximation algorithm for SBP-R

1: Find a maximal R-independent set of anchors A, i.e. leaves such that a feasible bus can visit at most one of them. This defines a skeleton Q of the instance T.
2: Initialize a default bus at each anchor $a \in A$.
3: **for** each junction point $j \in J$ in bottom-up order **do**
4: Assign an arbitrary left-to-right ordering of the two segments S_l, S_r with $t(S_l) = j = t(S_r)$. If $j = s$, then $S_l = S_r$ is possible. See also figure 3.3b.
5: Let $\mathscr{C}_1$ be the collection of short subtrees whose root node lies in the left core segment S_l at distance $\leq R/2$ from j. Denote by S_j the core segment with $b(S_j) = j$, and let $\mathscr{C}_2$ be the collection of short subtrees whose root node lies on S_j at distance $> R/2$ from $t(S_j)$. Note that $C_2 = \emptyset$ is possible.
6: Apply Lemma 3.3.1 to $\mathscr{C}_1 \cup \mathscr{C}_2$ on $P = S_l \cup S_j$, and obtain tickets $E_1, \ldots, E_y$ and overhead ticket E_0.
7: Let $\mathscr{C}_3$ be the collection of short subtrees whose root node lies in the right core segment S_r at distance $\leq R/2$ from j.
8: Apply Lemma 3.3.1 to $\mathscr{C}_3$ on path $P = S_r$, and obtain tickets $F_1, \ldots, F_z$ and overhead ticket F_0.
9: Assign E_0 and F_0 to the default bus at $\phi(j)$.
10: Place the y tickets $E_1, \ldots, E_y$ and z tickets $F_1, \ldots, F_z$ at $r(S_l) = r(S_r)$.
11: **end for**
12: **for** each anchor $a \in A$ **do**
13: Let $\mathscr{C}_a$ be the collection of short subtrees whose root node lies in the core segment S_a with $b(S_a) = a$ at distance $> R/2$ from $t(S_a)$.
14: Apply Lemma 3.3.1 to $\mathscr{C}_a$ on $P = S_a$, and obtain tickets $K_1, \ldots, K_w$ and overhead ticket K_0.
15: Assign K_0 to the default bus at a.
16: Place the w tickets $K_1, \ldots, K_w$ at a.
17: **end for**
18: Assign every bus greedily the lowest ticket available on its path to the root.
19: Add a new bus for each unmatched ticket.

In total, the bus from anchor a collects regret of at most $5R + 5R + R + 2.5R = 13.5R$. □

In the following, we refer to the non-overhead tickets that are assigned in step 6 of the algorithm as *unmatched tickets*. Let B be the number of bus routes output by the algorithm. We obtain:

Lemma 3.3.4. *B is a lower bound on the number of buses with regret at most R needed to cover all points.*

Proof. We can interpret B as the number of anchors plus the number of unmatched tickets. We traverse the tree now bottom up from the anchors and inductively compute at each anchor or junction point a lower bound on the number of buses needed to cover some subtree below.

Let j be a node that is either an anchor or a junction point. Denote by T_j the subtree of the skeleton rooted at j, together with all the (non-overhead) tickets placed within this subtree. Our claim is that the number of buses with regret R needed to cover T_j is at least the number of anchors in T_j plus the number of unmatched tickets in T_j. We distinguish two cases at j:

Case 1: There is at least one unmatched ticket placed at j. In T_j, the number of non-overhead tickets clearly exceeds the number of anchors by exactly the number of unmatched tickets within T_j. However, the number of non-overhead tickets is a lower bound by observation 1 and our claim holds.

Case 2: There is no unmatched ticket placed at j. If j is an anchor, we clearly need one bus to pick up j. Otherwise, we can focus on the lower bounds at the two successor junction points j_1, j_2 below j. We claim that a bus starting in the subtree T_{j_2} cannot go to the subtree T_{j_1} to pick up edges from a ticket (and vice versa). To prove this, we can assume that a ticket $\mathcal{T}$ is placed at j_1. Consider an edge $e \in T \setminus Q$ that is contained in $\mathcal{T}$. We distinguish the cases how the ticket $\mathcal{T}$ was formed to prove that every bus that covers e cannot start in T_{j_2}. Let T_e denote the short subtree containing e.

- T_e is rooted at a core segment S at distance greater than $R/2$ from $t(S) = j$, i.e. $T_e \in \mathcal{C}_2$ for j_1 ($T_e \in \mathcal{C}_a$ if j_1 is an anchor a). Then a bus starting in T_{j_2} can not pick up an edge of T_e, since the regret of the corresponding walk would be $> R$.

- T_e is rooted on a core segment S with $r(S) = j_1$, i.e. $T_e \in \mathcal{C}_1 \cup \mathcal{C}_3$ for some junction point j' (possibly $j' = j_1$). By the definition of $r(S)$, it follows that T_e is rooted at distance greater than $R/2$ from j. Therefore, a bus starting in T_{j_2} can not pick up an edge of T_e, since the regret of the corresponding walk would be $> R$.

It follows that the lower bound at j is the sum of the buses needed at each of the disjoint subtrees rooted at j_1 and j_2. Note that the number of anchors (unmatched tickets, resp.) within T_j is exactly the sum of the anchors (unmatched tickets, resp.) within T_{j_1} and T_{j_2}. This proves our induction step, and therefore our claim. □

Theorem 3.3.5. *There exists a* 13.5*-approximation algorithm for the SBP-R on trees in the case of unlimited capacity.*

Proof. Observe that lemma 3.3.3 and 3.3.4 yield a 13.5-approximation algorithm for SBP-R on trees: Given a value for R, we find a feasible solution to the SBP-R with value at most $13.5 \cdot R$ if $B \leq N$ holds. If not, then lemma 3.3.4 ensures that $OPT > R$. Thus doing binary search for R yields the result. □

3.3.2 General graphs

In this section, we present an inapproximability result for the SBP-R that is based on a reduction of an NP-hard Satisfiability problem to SBP-R. The main idea of the reduction is to construct an instance of SBP-R where every node in W can be covered via a path that is a shortest path from its location to the school (and therefore we can have a solution with maximum regret 0) if and only if there is a satisfying truth assignment for the Satisfiability problem.

In our construction, the capacity bound C will play a crucial role: we ensure that $N \cdot C = |W|$, and therefore each bus must pick up exactly C children. This implies that the problem reduces to partitioning the children among the buses. In particular, we will introduce a set of children for each clause, and two sets of nodes for each variable, namely a "true" set and a "false" set. A bus route from a node of the "true" set via a vertex from the clause set to the school for example corresponds to setting the variable to true to satisfy the clause.

By carefully constructing the graph, we can ensure that, in order to have regret zero, the set of children corresponding to a clause must be picked by a bus together with one set of variable nodes corresponding to one of the literals in the clause. Furthermore, by introducing a special gadget graph, we can ensure that for every variable, either the "true" set or the "false" set can be picked together with sets corresponding to clauses, but not both. This implies that each child can be picked up by a bus that never deviates from the shortest path to the school if and only if there is a truth assignment satisfying the given Satisfiability formula.

Theorem 3.3.6. *It is NP-hard to distinguish whether an instance of SBP-R has optimal regret value 0 or strictly positive.*

Proof. The reduction is from a variant of the Satisfiability problem, namely $(3,4)$-SAT. Formally, we are given n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$, where each variable appears in at most 4 clauses and each clause contains exactly 3 literals. The task of deciding if a truth assignment exists that satisfies all clauses is NP-complete [Tov84].

We construct an instance of SBP-R for a given instance of $(3,4)$-SAT as follows.
For every variable x_i, introduce 16 nodes, namely the nodes $x_i[j], \overline{x}_i[j]$ and $a_i[j], b_i[j]$ $\forall 1 \leq i \leq n, j = 1,..,4$. Note that the total number of those nodes is $4 \cdot 2n + 2 \cdot 4n$. We add edges between $x_i[j]$ and $a_i[j]$, $\overline{x}_i[j]$ and $a_i[j]$, $a_i[j]$ and $b_i[j]$ as well as $a_i[j]$ and $b_i[(j+1) \mod 4]$. All nodes $b_i[j]$ are connected to a root node s. We will refer to *variable nodes* for the nodes $x_i[j], \overline{x}_i[j]$, $1 \leq i \leq n, j = 1,..,4$. Figure 3.4 shows on the left this part of the construction.

Additionally, we create two nodes c_k and Δ_k for every clause, $1 \leq k \leq m$. Every Δ_k is connected to the root s and to c_k. The node c_k is connected to all literals that it contains, e.g. to $\overline{x}_i[j]$ for exactly one $j \in \{1,2,3,4\}$ if the literal $\overline{x}_i$ is part of the k-th clause. We remark that for every i and j, either $x_i[j]$ or $\overline{x}_i[j]$ can be connected to a clause c_k, but never both and never to more than one clause. In other words, the connection of the clause nodes to pairs of vertices $x_i[j]$

and $\overline{x}_i[j]$ is one-to-one. Finally, we add $4n-m$ nodes g_ℓ, each connected to all $x_i[j], \overline{x}_i[j]$ and Δ_k, $1 \le i \le n, 1 \le j \le 4, 1 \le k \le m$. An additional node $\star$ is connected to the root s and all g_ℓ, $1 \le \ell \le 4n-m$. See figure 3.4 on the right for the second part of the construction.

All edges have unit weight and we set $W = V$ in our construction, i.e. all vertices correspond to locations of children.

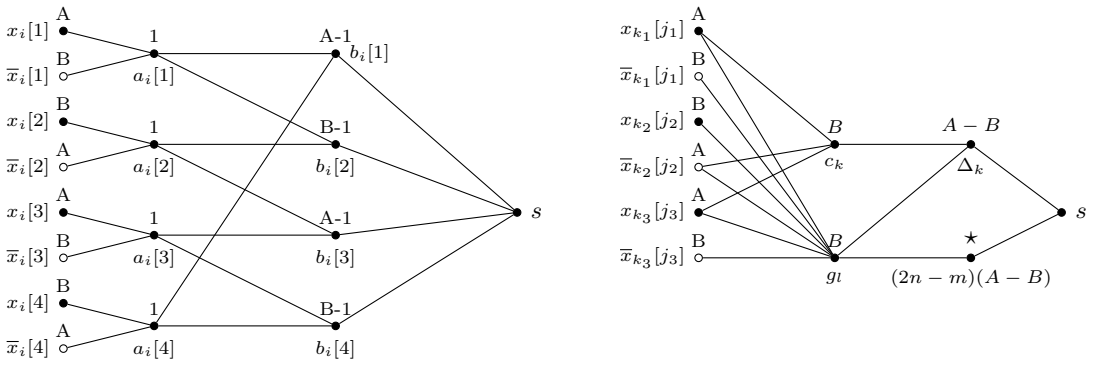


Figure 3.4 – Gadget for each variable x_i and for a clause $C_k = x_{k_1} \vee \overline{x}_{k_2} \vee x_{k_3}$, respectively

To simplify the construction, we define node demands $r : V \to \mathbb{N}$ that represent node copies connected by zero weight edges[1]. Let $A := 3$ and $B := 2$ be constants that we use for the further description.On the variable nodes, we set demands in an alternating way, namely $r(x_i[1]) = r(\overline{x}_i[2]) = r(x_i[3]) = r(\overline{x}_i[4]) := A$ and $r(\overline{x}_i[1]) = r(x_i[2]) = r(\overline{x}_i[3]) = r(x_i[4]) := B$. Furthermore, we set $r(a_i[j]) := 1$ for $j = 1,2,3,4$, $r(b_i[2]) = r(b_i[4]) := B-1$ and $r(b_i[1]) = r(b_i[3]) := A-1$. Finally, we set $r(g_l) = r(c_k) := B$, $r(\Delta_k) := A-B$ and $r(\star) := (2n-m)(A-B)$ for $\ell = 1,\dots,4n-m$ and $k = 1,\dots,m$. Note that $r(\star) > 0$, since $A > B$ and $m \le \frac{4n}{3}$.

The total demand that we distribute is $\sum_{i=1}^{n}\sum_{j=1}^{4} r\left(x_i[j]\right) + r\left(\overline{x}_i[j]\right) = 4nA + 4nB$ on the nodes for the appearances of variables, $\sum_{i=1}^{n}\sum_{j=1}^{4} r\left(a_i[j]\right) + r\left(b_i[j]\right) = 2nA + 2nB$ on the a and b nodes, $\sum_{i=k}^{m} r(c_k) + r(\Delta_k) = mB + m(A-B)$ on the clause nodes, $\sum_{l=1}^{4n-m} r(g_l) = (4n-m)B$ and $r(\star) = (2n-m)(A-B)$. In total, we have a demand of $8n(A+B)$. Recall that this construction involving node demands corresponds to an equivalent construction with $8n(A+B)$ nodes without demands where some nodes are connected by zero weight edges.

We set $N := 8n$ to be the number of bus routes and the capacity bound to be $C := A+B$. Since the total demand equals the total capacity of the buses, every bus clearly has to be full. We now make several observations for a solution to SBP-R with regret value 0.
Clearly, every node must be covered via one of its shortest paths to the root. In other words, every node x must be a starting point of one path of length 3. Observe that a single bus

[1] Observe that instead of zero weight edges, one can think of a path with edges of length γ for $\gamma > 0$ a sufficiently small constant.

starts from every x-node. This is simply by the choice of the number N of buses, since there are N variable nodes. Further, since there is a bus starting at every x-node and there are exactly N nodes at distance 2 from the root s, the solution induces a perfect matching on the x-nodes and the a, c, g-nodes. In particular, each clause node is mapped to an x-node that corresponds to a literal in that clause. Similarly, there is exactly one bus passing through each a-node.
This implies that there is just one bus for every b-node, thus the demand at each b-node is satisfied by a single bus. By construction, a gadget subgraph on a_i and b_i vertices for some variable x_i is covered by either 4 paths containing the edges between $a_i[j]$ and $b_i[j]$ for $1 \leq j \leq 4$, or 4 paths containing only the edges between $a_i[j]$ and $b_i[(j+1) \mod 4]$. This can be seen by the fact that an SBP-R solution of regret 0 induces a maximum matching[2] in the gadget subgraph. Note that in the gadget subgraph, there are exactly two perfect matchings, namely those mentioned previously.

Using these observations, we are able to prove the statement in two steps. First, we show that a satisfying truth assignment for the $(3,4)$-SAT instance can be used to find N paths of regret 0. Second, we show that a solution of N paths with regret 0 to the SBP-R gives rise to a satisfying truth assignment of the $(3,4)$-SAT instance.

Let us first assume to have a satisfying truth assignment. If variable x_i is set to true in this assignment, we choose the paths $\{\overline{x}_i[j], a_i[j], b_i[j], s\}$ for all $1 \leq j \leq 4$. Otherwise, we choose the paths $\{x_i[j], a_i[j], b_i[(j+1) \mod 4], s\}$ for all $1 \leq j \leq 4$. In both cases, note that those $4n$ paths have a total demand of $A+B$ each and have regret 0.
For every clause c_k, we select one node that corresponds to a literal that makes the clause satisfied. We start a path from that node to c_k. If that variable node has demand B, we pick up the demand at the node Δ_k as well before ending the path at s, otherwise we route from c_k directly to s since the bus is full. This procedure defines m paths of regret 0 and demand $A+B$.

For the $(4n-m)$ variable nodes that remain, we choose paths via the $(4n-m)$ nodes g using a perfect matching between them and such that all remaining demands at the Δ-nodes and the $\star$ are satisfied. This is clearly possible since both the Δ-nodes and the $\star$ node are connected to all g-nodes. Hence we obtain $4n-m$ paths of regret 0 and demand $A+B$, since the total remaining demands for those paths was $(4n-m)(A+B)$.

In total, we construct N paths of demand $A+B$ covering all vertices, thus we have a solution for SBP-R of value 0.

Now suppose that we have solution for SBP-R of regret 0. Recall that in such a solution, there is a one-to-one correspondence of the clause vertices c_k and the variable vertices. That means that for every clause c_k, there is a path covering c_k that starts at a node x_i or $\overline{x_i}$. Set the

[2] Here and in the following, this matching is with respect to the graph without the demands, i.e. ignoring the A node copies for a node of demand A.

corresponding literal to true. It remains to show that this choice is never conflicting for a variable.

Suppose that there is a conflict for variable x_i, i.e. we chose $x_i[j]$ and $\overline{x}_i[j']$. Because of the one-to-one correspondence, there is either an edge between $x_i[j]$ and a clause or between $\overline{x}_i[j]$ and a clause, but not for both. Thus $j \neq j'$. Note that due to the tight capacity constraint, we must have a path from $\overline{x}_i[j]$ to $a_i[j]$, $b_i[j]$ and the root. Analogously, we must have a path from $x_i[j']$ to $a_i[j']$, $b_i[(j'+1) \mod 4]$ and the root. However, this is impossible due to the construction of the bipartite graph between the nodes a_i and b_i, since every path in the solution has length 3 and regret 0. □

Note that every approximation algorithm for SBP-R with a finite approximation factor has to find an optimal solution if the optimal regret value is 0.

Corollary 3.3.7. *Unless P=NP, SBP-R is not approximable within any function $f(n)$, where n denotes the size of the input of SBP-R.*

We remark that this result does not hold in case of unlimited capacity.

3.4 Extensions and further variants

3.4.1 Multiplicative regret

In this section, we study variants of the School Bus problem with multiplicative regret. We consider a delay factor $R \geq 1$. A rooted walk P is feasible, if the regret bound $d^P(s,v) \leq R \cdot d(s,v)$ holds for all vertices v on this walk. Recall that for unlimited capacity, SBP-mult admits a $O\left(\log \frac{R}{R-1}\right)$-approximation and a $O(1)$-approximation algorithm for SBP-R-mult [FS14]. Furthermore, a $(1+\frac{2}{R-1})\alpha_{TSP}$-approximation is known for SBP-Σ-mult with unlimited capacity [GNR11], where α_{TSP} denotes the approximation factor for TSP. Together with Lemma 3.1.1, this result implies a constant approximation for the capacitated case (cf. [GNR11]).

We present several hardness and inapproximability results for variants of the School Bus problem with multiplicative regret. An overview of the known results can be found in Table 3.4.

	approximation	hardness
SBP-mult	$O(\log \frac{R}{R-1})$	2
SBP-Σ-mult	$(1.5+\frac{3}{R-1})$	$\frac{123}{122}$
SBP-R-mult	$O(1)$	NP-hard

Table 3.4 – Results for variants of SBP-mult with unlimited capacity

We prove first that SBP-mult and SBP-R-mult are NP-hard on stars even with unlimited capacity.

Lemma 3.4.1. *SBP-mult with unlimited capacity is NP-hard on metrics induced by a star.*

Proof. We give a reduction from the 3-partition problem [GJ79]: Given $3m$ integers $a_1, \ldots, a_{3m}$ and an integer B, decide whether there are m disjoint triples of size B each. This problem is NP-hard even if $\frac{B}{4} < a_i < \frac{B}{2}$ holds for all $i = 1, \ldots, 3m$.

We construct an instance of SBP-mult with unlimited capacity as follows. Let s be the center of a star with $3m$ branches of length $a_1, \ldots, a_{3m}$ and m branches of length $\frac{B}{2}$ each connecting s to m extra vertices $v_1, \ldots, v_m$. Note that this star graph has $4m+1$ vertices. We set $R = 5$.

We first observe that we can without loss of generality restrict our attention to rooted walks whose vertices are traversed in order such that the distance from the root is non-decreasing. Furthermore, any such rooted walk that visits at most 3 vertices is clearly feasible.

The first claim is that every feasible rooted walk visits at most 4 vertices except for the root. To show this, consider a rooted closed walk that visits 4 vertices apart from the root. The minimum length of such a walk is $> 2B$, since every vertex is at distance $> \frac{B}{4}$ from the root[3]. This means that this path cannot be prolonged in a feasible way, because the next vertex would have to be at distance $> \frac{B}{2}$ from the root, but there is no such vertex in the instance.

The second claim is that every feasible rooted walk visiting (at least) two extra vertices v_i, v_j for $1 \leq i < j \leq m$ visits at most 3 vertices except for the root. Without loss of generality, suppose v_j is visited after v_i. Analogously to the previous argument, every rooted closed walk that visits 2 vertices apart from the root has length $> B$. The regret bound of v_j imposes the condition that the length of the closed walk before visiting v_j is at most $2B$. This implies that the length of the closed walk before visiting v_i is at most B and thus the second claim is proved.

Finally, we claim that there is a solution for SBP-mult using m walks if and only if there is a solution to the 3-partition instance.
Suppose we have a solution to the 3-partition instance. Then we create for every triple in the 3-partition a rooted walk that traverses the vertices corresponding to the elements in the triple such that the distance from the root is non-decreasing and ends in an extra vertex. This walk is feasible by construction and visits 4 vertices apart from the root. This means there is a solution for SBP-mult with m walks.

Now suppose that there is a solution for SBP-mult using m walks. To show that these walks define a solution to the 3-partition instance, we employ a cardinality argument. Combining the first claim with the fact that there are $4m$ vertices plus the root s in the instance, all of the m walks in the solution of SBP-mult have to visit exactly 4 vertices except the root. Together with the second claim, this implies that every walk contains exactly one extra vertex. Because

[3] We note that the minimum length is actually $> \frac{9}{4}B$ if we take the feasibility of the last vertex into account.

of the regret bound imposed at the extra vertex, the length of the closed walk before traversing the last edge to the extra vertex is at most $2B$. In other words, the walks in the solution of SBP-mult define a 3-partition of the elements $a_1, \ldots, a_n$.

□

A corollary is the NP-hardness of SBP-R-mult, since the decision problem for SBP-R-mult and SBP-mult is the same.

Corollary 3.4.2. *SBP-R-mult with unlimited capacity is NP-hard even on metrics induced by a star.*

As observed in the introduction, SBP-Σ-mult has a trivial optimal solution on a star. However, it is NP-hard already on trees of height 2.

Lemma 3.4.3. *SBP-Σ-mult with unlimited capacity is NP-hard on tree metrics.*

Proof. We have to alter the construction in the proof of Lemma 3.4.1 only slightly. Given an instance of the 3-partition problem, we construct an instance of SBP-Σ-mult as for SBP, except that we add a new root vertex s' that is connected to the center s of the star by an edge of length $\frac{B}{2}$. We set the delay factor $R := 3$. Note that we can restrict ourselves again to walks that traverse the vertices in order such that the distance from the root is non-decreasing. Furthermore, every rooted walk with at most 3 vertices except for s and s' is feasible, i.e. respects the deadlines imposed by the delay factor. Observe that, as before, every feasible rooted walk has at most 4 vertices and that every feasible rooted walk with 4 vertices contains at most one extra vertex v_i.

Given a 3-partition, we can easily construct m rooted walks of length $3B$ that cover each exactly 4 vertices except for s, s'. Conversely, an analogous cardinality argument as for SBP shows that a solution for SBP-Σ-mult of total length $3mB$ allows to derive a 3-partition. It follows that SBP-Σ-mult with unlimited capacity is NP-hard on trees of height 2. □

Additionally, we obtain the following inapproximability result for SBP-mult similarly to Theorem 3.2.1 that was also independently observed in [FS14].

Lemma 3.4.4. *There is no approximation algorithm for SBP-mult with a factor less than 2 unless P=NP.*

Proof. We provide a simple reduction from the following NP-hard decision problem [GJ79]: Given a graph G and an integer L, is there a Hamilton path including all vertices of length at most L?

We add a new root vertex s at distance L from every vertex in G and we set the regret factor to $R = 2$. We observe that the optimal solution of SBP-mult in the new graph has one bus route if

and only if there is a Hamilton path in G of length at most L. Note that any α-approximation algorithm with $\alpha < 2$ would thus allow us to decide the Hamilton path problem. □

Clearly, SBP-Σ-mult also generalizes TSP (unlimited regret factor) and thus SBP-Σ-mult is NP-hard to approximate within a factor of $\frac{123}{122}$ [KLS13]. For SBP-R-mult, Theorem 3.3.6 implies the following corollary.

Corollary 3.4.5. *There is no approximation algorithm for SBP-R-mult with a factor less than $\frac{5}{3}$ unless P=NP.*

Proof. If the Satisfiability instance in the proof of Theorem 3.3.6 is satisfiable, then there is a solution for SBP-R-mult using only paths of length 3. However, if the formula is not satisfiable, observe that every solution for SBP-R-mult must employ at least one path of length ≥ 5. □

3.4.2 Distance-constrained vehicle routing problem (with split-delivery)

In this section, we present an application of our result for Capacitated Orienteering (COP) to a generalization of the distance-constrained vehicle routing problem (DVRP) that we call DVRP+. We additionally consider node demands $r : V \to \mathbb{N}$ and the task is still to find the smallest number of rooted tours such that each demand is fulfilled and both a length bound $D \in \mathbb{N}$ and a capacity bound $C \in \mathbb{N}$ is respected for each tour. Note that every node can only be served by exactly one vehicle.

We observe that a $O(\log C)$-approximation for this generalization of DVRP follows from the existence of an $O(1)$-approximation algorithm for COP. Together with the previously known results, this implies a $\min\left\{O(\log C),\ O(\log n), O(\frac{\log D}{\log\log D})\right\}$-approximation for DVRP+. It is still open whether a constant factor approximation algorithm can be achieved for DVRP+.

Theorem 3.4.6. *There exists a $O(\log C)$-approximation algorithm for DVRP+.*

Proof. A greedy algorithm, applied to a DVRP+ instance, repeatedly searches for a feasible tour that respects both the capacity bound and the length bound while maximizing the total demand of the visited nodes, until every demand node is satisfied. It turns out that the problem we want to solve in each step of such a greedy algorithm is exactly a COP instance where $s = t$ and $\pi(v) = r(v)$ for all nodes v. Thus we use the constant factor approximation algorithm for COP from Theorem 2.2.1 as subroutine for a greedy algorithm. The result then follows by applying the classical Dual-Fitting analysis for the Set Cover problem [Vaz01].

DVRP+ can easily be formulated as a covering integer program. We present the linear relaxation

$$\begin{aligned} \min \quad & \sum_{T \text{ feasible}} x_T && \text{(LP)} \\ & \sum_{T: v \in T} x_T \geq 1 \quad \forall v \in V \\ & x_T \geq 0 \qquad \forall \text{ feasible tour } T. \end{aligned}$$

and its dual linear program

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v && \text{(D)} \\ \text{s.t.} \quad & \sum_{v \in T} y_v \leq 1 \quad \forall \text{ feasible tour } T \\ & y_v \geq 0 \quad \forall v \in V. \end{aligned}$$

Note that a tour T is feasible if it starts and ends at the root, its length does not exceed D and the satisfied demands are at most C.

The dual fitting approach runs the greedy algorithm for Set Cover and then constructs a feasible dual solution of value at most $O(\log C)$ times the number of tours in the greedy output. Observe that we thereby obtain an upper bound of $O(\log C)$ on the integrality gap of *(LP)* as well as the claimed $O(\log C)$-approximation algorithm.

The greedy algorithm we run works as follows:

1. Set $L = \emptyset$ and profits $\pi = r$.
2. While $L \neq V$ do
3. Find (approximately) the feasible tour T with highest profit.
4. Set $price(v) = \frac{\pi(v)}{\pi(T)}$ for all newly covered nodes $v \in V(T) \setminus L$.
5. Set $\pi(v) = 0$ for $v \in V(T) \setminus L$ and $L = L \cup V(T)$

Observe that step 3 of the greedy algorithm is precisely to solve an instance of COP (where the start and the end node coincide). In the following, we will use L_v to denote the set of nodes covered before the iteration in which v was covered.

We construct the dual solution

$$y_v = \frac{1}{\alpha H_C} price(v),$$

where α is the (constant) approximation factor for COP and $H_C = 1 + \frac{1}{2} + \cdots + \frac{1}{C}$.

Before we show that this dual solution y is feasible, let us make the following useful observations:

(i) $\sum_{v \in V} price(v) = Greedy$, where $Greedy$ denotes the number of tours output by the greedy algorithm. This fact follows simply from step 4 of the greedy algorithm, where we distribute a value of 1 among the newly covered nodes.

(ii) We can upper bound the price of any node v as follows. Let Q be any feasible tour that covers v and let T denote the tour selected by the greedy algorithm to cover v. We obtain:

$$price(v) = \frac{r(v)}{r(T \setminus L_v)} \leq \alpha \frac{r(v)}{r(Q \setminus L_v)}$$

since Q had a profit of $r(Q \setminus L_v)$ in the iteration when v was covered and thus $r(T \setminus L_v) = \pi(T) \geq \frac{1}{\alpha}\pi(Q) = \frac{1}{\alpha} r(Q \setminus L_v)$.

Now consider a feasible tour Q and an ordering of its vertices $v_1, \ldots, v_{|Q|}$ according to the order in which they were covered by the algorithm. We get that the dual solution y is indeed feasible, since

$$\sum_{v \in V(Q)} y_v \leq \frac{1}{\alpha H_C} \sum_{v \in V(Q)} \alpha \frac{r(v)}{r(Q \setminus L_v)} = \frac{1}{H_C} \sum_{i=1}^{|Q|} \frac{r(v_i)}{r(Q) - \sum_{1 \leq j < i} r(v_j)} \leq 1$$

To see the last inequality, notice that for every i

$$\frac{r(v_i)}{r(Q) - \sum_{1 \leq j < i} r(v_j)} \leq \sum_{h=0}^{r(v_i)-1} \frac{1}{r(Q) - \sum_{1 \leq j < i} r(v_j) - h}$$

and therefore

$$\sum_{i=1}^{|Q|} \frac{r(v_i)}{r(Q) - \sum_{1 \leq j < i} r(v_j)} \leq \sum_{h=0}^{r(Q)-1} \frac{1}{r(Q) - h} \leq H_C,$$

since $r(Q) \leq C$.

The feasibility of the dual solution y together with observation (i) proves that the greedy algorithm is in fact a $O(\log C)$-approximation algorithm for DVRP+. Note that this result also bounds the integrality gap of *(LP)* by $O(\log C)$. □

We observe that this result holds as well if a node can be served by several vehicles. Archetti et al. [ASS06] proved that the optimal solution to the DVRP+ with split delivery can be transformed into a solution for DVRP+ with at most twice as many routes where every node is served exactly once. Therefore, our results also imply a $O(\log C)$-approximation for DVRP+ with split delivery.

4 The School Bus problem in Practice

4.1 Introduction

After studying the School Bus problem for approximation algorithms and theoretical hardness results, we present in this chapter an application of the School Bus problem in practice. Based on an integer programming formulation of the School Bus problem, we implemented a heuristic using column generation to solve a real-life instance provided by a private school in the lake Geneva region. Our results show that our approach yields a big improvement in terms of customer satisfaction when compared to the previously used bus schedule.

4.1.1 Related work

Many variants of School Bus routing problems have been studied in the literature. We refer to the surveys by Park and Kim [PK10] and the book by Toth and Vigo [TV01], as well as references therein. The PhD thesis of Spada [Spa04] considers a school bus routing problem combined with a scheduling problem. She presents approaches based on simulated annealing and tabu search and variable neighbourhood search. She further introduces the notion of regret (service level) as a quality measure, namely the difference of the time spent in the bus compared to the time needed in the parents' car. She provides results on two real-life instances from the Lausanne area. The main ideas of her work are presented in [SBL05].
An approach based on solving the LP relaxation of a set cover integer programming formulation with column generation has been shown to be very efficient and successful in practice for the vehicle routing problem with time windows [DDS92].

4.1.2 Our results

We describe a framework for maximizing the customer satisfaction that we measure using the notion of regret. The regret minimization and the usage of the minimum number of buses form two opposing objectives. We observe that we can equivalently assume a bound on one and minimize the other and vice versa. However, it is much simpler to solve an integer linear

programming formulation for minimizing the number of buses assuming a bound on the maximum regret. The covering integer linear program that we have discussed in Section 3.2.1 is known to have a small integrality gap of 4 on tree metrics and of 16 in general. Its resolution relies on solving an Orienteering problem (cf. Chapter 2) as a subroutine.
Eventually, we describe a solution for the bus system of the private school that uses exactly the same buses with their respective capacities and reduces the maximum regret by 20%.

4.2 A heuristic based on integer programming and column generation

We propose an approach that is based on solving an integer linear programming formulation approximately. Before presenting this method in more detail, we first describe the setting of the real-life instance and our quality measure.

4.2.1 Description of the instance

Based on the existing timetable for the school bus system of the private school in the lake Geneva area, we extracted the following general information.

REAL-LIFE INSTANCE

- 1 school and 139 different bus stops
- 283 pupils to be picked up at the bus stops
- 11 buses of various capacities

We are modelling this instance using a complete, undirected graph $G = (V, E)$ with 140 nodes, where a root node s represents the school. The number of pupils to be picked up at the bus stops is represented by a node demand function $r : V \to \mathbb{N}$. A distance function $d : E \to \mathbb{N}$ on the edges indicates the travel time between two nodes. For a path P, we will denote by $d^P(s, v)$ the distance between s and v along the path P, whereas $d(s, v)$ is the shortest distance between s and v.
Since the task of taking the pupils to school in the morning and the task of taking them home in the afternoon are symmetric, we focus in the following on the prior. Our approach can be applied in the same way also to the latter task.

4.2.2 Preliminaries

In order to establish distances between all locations, we implemented a script sending requests to Google Maps. The Google Maps API answers such a request with a distance and a duration

of an optimal route. Each request was considered in default mode, i.e. travel mode is driving with average speed and the use of highways is allowed. We store all pairwise travel times. Since Google Maps uses an objective that takes both the length and the duration of a route into account, both the distances and the durations can be non-metric (in particular, inserting a point to a route can shorten it in either distance or travel time). However, our algorithm that we will describe in the next section relies on metric weights on the edges of a graph, so the Google Maps output was adjusted to form a metric: First, it was made symmetric by keeping only the shorter of the two directions and second, the output was transformed into a metric by iteratively adapting the length of an edge that does not satisfy the triangle inequality.

Our goal is to minimize what we call the *regret* of a child, namely the comparison of the travel time to school on the bus and in a private car. This comparison can be made in terms of either subtracting or dividing the travel times using the car and the bus, respectively. The first variant is referred to as additive regret, whereas the second is called multiplicative regret. In the following, we will focus on minimizing the maximum additive regret, i.e the maximum time difference between the travel on the bus and on the shortest path.

The purpose of our approach is to keep the same setting as the previously implemented bus system. In particular, we aim at using exactly the same buses with their various capacities in order to facilitate the comparison and as well as the adoption of the proposed solution. However, if we allow modifications of the instance, it is also of economical interest to see if one can reduce the number of used buses for a limited maximum regret.

4.2.3 IP formulation

In this section, we present the integer linear programming formulation that we used for our heuristic. We recall that the goal is to minimize the maximum additive regret for a given number of buses. However, it turns out that an integer linear program for minimizing the total number of bus routes for a fixed upper bound R on the maximum regret is much simpler to formulate and solve. Since the optimum solutions of these two optimization problems coincide for the correct parameter setting, we can use this simpler formulation to devise a good solution for our actual problem.
We have to do one more modification. Namely, we will ignore the node demands for our integer linear programming formulation and instead consider a parameter C that bounds the number of bus stops on a bus route. In this way, we can ensure that every bus stop is visited exactly once.

The following covering integer linear programming formulation has variables for every feasible path and asks to minimize the total number of feasible paths that are necessary to cover all vertices. A feasible path P in this formulation is rooted at the vertex s representing the school, visits at most C bus stops and the maximum additive regret $\max_{v \in V(P) \setminus \{s\}} d^P(v,s) - d(v,s)$

is bounded by R.

$$\begin{aligned} \min \quad & \sum_{P \text{ feasible}} x_P && \text{(IP)} \\ & \sum_{P: v \in V(P)} x_P \geq 1 \quad \forall v \in V \\ & x_P \in \{0,1\} \qquad \forall \text{ feasible path } P. \end{aligned}$$

We first observe that this formulation has an exponential number of variables. We can therefore not simply apply an integer programming solver to compute a solution. We overcome this difficulty by using a column generation approach to solve the linear relaxation of (IP) in CPLEX 12. A column generation approach consists in generating only those path variables that potentially reduce the objective value, i.e. total number of path variables set to 1. These paths can be found by iteratively solving the separation problem of the dual linear program. We refer to [Fei10] for an excellent introduction into the usage of column generation approaches for vehicle routing problems. The linear relaxation of (IP) simply considers the range $[0,1]$ for rational path variables x_P instead of binary variables, i.e. the discrete range $\{0,1\}$. In order to solve the linear relaxation of (IP), we use the interplay of the column generation approach with solving the separation problem of the dual linear program (cf. [GLS88]). The dual linear program is

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v && \text{(D)} \\ \text{s.t.} \quad & \sum_{v \in V(P)} y_v \leq 1 \quad \forall \text{ feasible path } P \\ & y_v \geq 0 \quad \forall v \in V. \end{aligned}$$

The separation problem for (D) is thus to find a feasible path with maximum y-value. In order to solve the separation problem, we solve several Orienteering problems. For every node $v \in V \setminus \{s\}$, we consider the Capacitated Orienteering problem between s and v in the graph G with profits y, unit demands and capacity bound C, where the length bound is given by $d(s,v) + R$. Since the approximation algorithm that we devised in Chapter 2.2 is not very practical, we implemented the local search heuristic for the Orienteering problem presented by Chao et al. [CGW96] that repeatedly tries to insert and remove nodes to improve the total profit. The only modification was to limit the number of nodes in a solution to C.

After finding a solution to the linear relaxation of (IP), we let CPLEX find a solution to (IP) on the generated set of variables.

Using this solution of (IP), we reintroduce the node demands and assign to each path of the solution a bus with a suitable capacity.

We note that the solution that we compute with our approach is in general not optimal. This is mostly due to the fact that we use a fast heuristic as a subroutine for the Capacitated Orien-

teering problem. Furthermore, even with an exact algorithm for Capacitated Orienteering as a subroutine, we would obtain an optimal solution of the linear relaxation of (IP), but not of (IP) itself. This is because the variables that we generated for the solution may not contain the variables needed for the optimal solution of (IP).

4.2.4 Practical results

In our previous formulation, we set the parameters $R = 28$ (maximum regret in minutes) and $C = 25$ (maximum number of visited bus stops). Our approach yields a solution that we can translate to a set of bus routes for the real-life instance using the same given set of buses. The solution was computed in less than 15 hours on a MacBook Pro with 2.6 Ghz and 4GB RAM. The bus routes are described in table 4.1. Based on the distances obtained from GoogleMaps,

Bus no.	1	2	3	4	5	6	7	8	9	10	11
# children	32	18	26	26	14	21	34	24	49	21	18
Bus capacity	47	22	45	33	22	23	49	24	49	24	23
Max. add. regret [min]	25	25	27	28	19	26	28	26	24	23	26
Max. mult. regret	2.1	2.1	2.2	1.9	1.9	3.0	2.5	2.2	3.0	2.3	2.3

Table 4.1 – Overview of the computed bus routes

we can compare the previously existing timetable to our solution. We remark that in the previous timetable, the maximum additive regret is 35 minutes compared to 28 minutes in our solution. This implies an improvement of 20% on the maximum difference between the bus tour and the shortest path. A similar result is even achieved for the maximum multiplicative regret, where we have a worst ratio of 3.0 compared to 3.66 in the previous schedule, although the regret ratio was not part of the objective. Table 4.2 summarizes the results.

	previous solution	new solution
max. additive regret	35 min	28 min
max. regret ratio	3.66	3.0
number of buses	11	11

Table 4.2 – Comparison of the previous and our solution

We note that the average additive regret per child is below 11 minutes in our solution, whereas the multiplicative regret is 1.66 on average. Figure 4.2 shows an excerpt of the proposed output based on the locations of the bus stops indicated in Figure 4.1.

We observe that for a smaller regret value than $R = 28$ minutes (and $C = 25$), a solution would need at least 12 bus routes. However, it is of course also possible to reduce the number of bus routes below 11 by allowing for a higher maximum regret. Finally, we remark that service times, i.e waiting times at the bus stops, are not considered, since the number of stops on the bus routes with maximum regret are roughly the same for both the previous and the new

timetable.

Our approach has to ignore several obstacles like country borders and bus provider specific requirements, since their incorporation would render the problem too complicated and too large for an efficient handling using an integer programming formulation. The bus schedule that we computed with our heuristic could thus not be implemented directly in practice. The private school opted for a commercial implementation that used our solution as a guideline for an improved bus service.

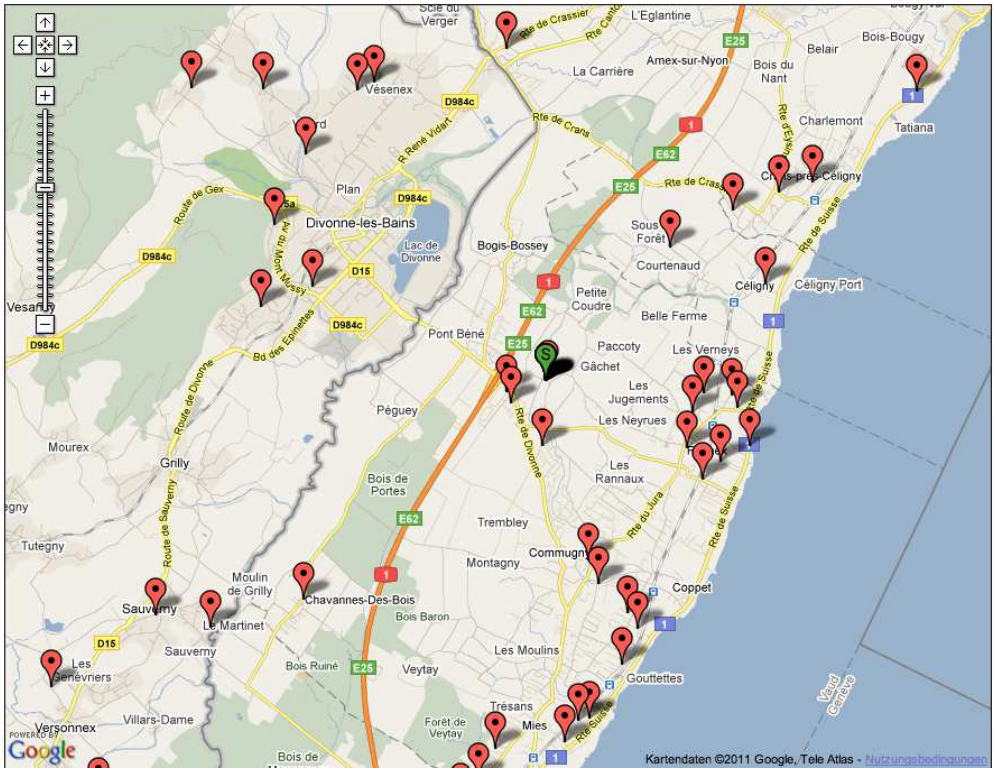


Figure 4.1 – Screenshot from Google Maps showing bus stops (red markers) and the school (green S-marker).

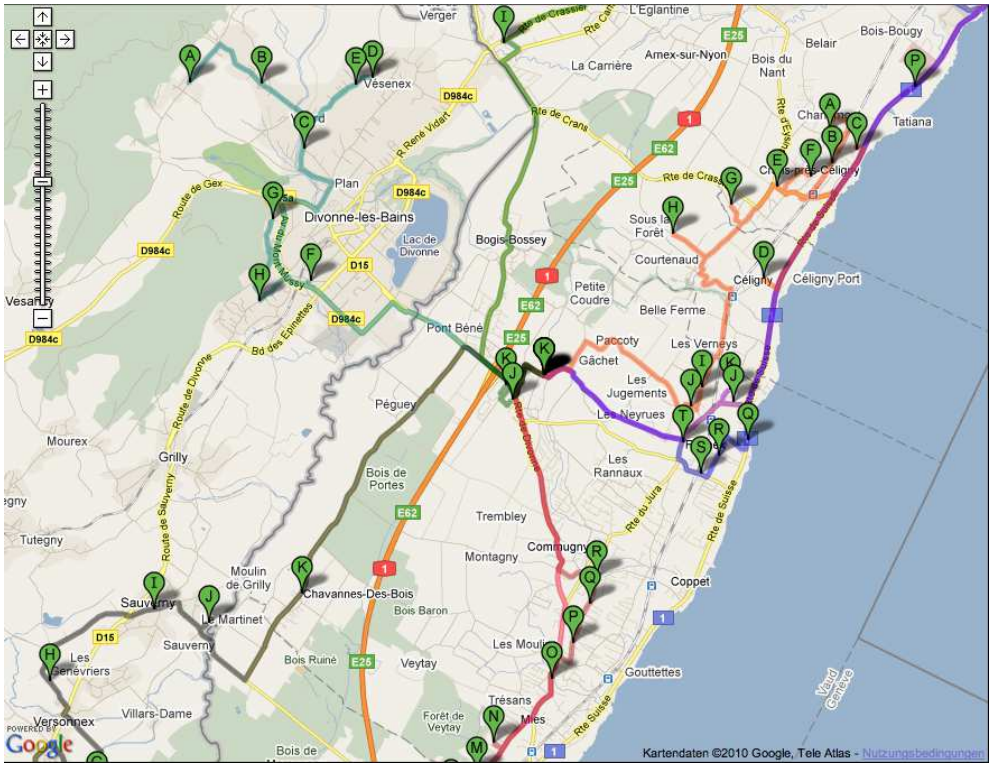


Figure 4.2 – Screenshot from Google Maps showing the suggested bus routes in different colors.

5 Summary and Conclusion

In this thesis, we studied several vehicle routing problems related to public transportation. Using the notion of regret as measure for customer satisfaction, we discussed variants of the Orienteering problem (Chapter 2) and the School Bus problem (Chapter 3). We finally described in Chapter 4 an integer linear programming approach for solving the School Bus problem in practice and presented results on a real-life instance.

For the studied problems, we designed approximation algorithms and proved hardness of approximation results. We are the first to analyse in a mathematically rigorous way the computational complexity of capacitated variants of Orienteering problems and their extensions. Previous works on these variants were looking for heuristics to solve them in practice. A general framework was developed to extend an approximation algorithm for any variant of Orienteering to respect an additional capacity constraint at a small loss in the approximation guarantee. Furthermore, we proved that the polynomial time approximation schemes for the Orienteering problem on trees and in the plane can be generalized to the capacitated case. Finally, new approximation algorithms for Orienteering with delay factor and Orienteering with deadlines were obtained.

We considered several variations of the School Bus problem. Subject to a regret threshold, the goal can be to minimize the number of bus routes or their total length. Conversely, the maximum regret is to be minimized using a given number of buses in an additional variant. For each of those problem variants, we provided theoretical lower and upper bounds on the approximation guarantee an algorithm can achieve; though there sometimes remains a gap between the hardness results and the best known approximation algorithm. Furthermore, we observed that one can often improve the approximation guarantee significantly if the focus is directed to interesting special cases of the more general problem. For instance, the School Bus problem with regret minimization can be approximated in case of unlimited capacity within $O(N^2)$ in general graphs (N denotes the given number of buses), whereas we have shown an elegant 13.5-approximation-algorithm on trees. We refer to Table 3.2 for an overview of our results for the variants of the School Bus problem on tree metrics.

In addition to the theoretical analysis of approximation algorithms, we also investigated their applicability to real-life scenarios. For a real-life instance provided by a private school in the lake Geneva region, we computed an improved bus schedule using the natural set cover integer linear programming formulation of the School Bus problem that we solved using a column generation approach. However, we note that our results are preliminary and have to be seen as a proof of concept, since the flexibility and the applicability of our approach in practice are rather limited.

At this point, we mention also a few questions for further research. It would be very interesting to study if a constant factor approximation algorithm can be found for the Orienteering problem with deadlines, maybe first considering the special case where the deadlines are proportional to the root distances. Furthermore, we have outlined a possible approach to obtain a polynomial time approximation scheme for the Orienteering problem on planar graphs, but the outcome is still unclear. Concerning the School Bus problem where we minimize the total length of the walks, the question for approximation algorithms in general graphs is left for future studies. Finally, it is open whether the School Bus problem with multiplicative regret admits a constant factor approximation on tree metrics.

Summing up, we gave combinatorial approximation algorithms and polynomial time approximation schemes based on dynamic programming for variations of the Orienteering and the School Bus problem. Additionally, we provided simple constructions that prove the hardness of approximation for some variants. We obtained several nice theoretical insights and thus gained a deeper understanding of these problems. Yet, many questions are still open and new questions came up during our studies, leaving room for further investigation and research.

Bibliography

[ACL09] Anna Adamaszek, Artur Czumaj, and Andrzej Lingas. PTAS for k-Tour Cover Problem on the Plane for Moderately Large Values of k. In *Algorithms and Computation*, LNCS 5878, pages 994–1003. Springer, 2009.

[ABST14] Enrico Angelelli, Cristina Bazgan, Maria Grazia Speranza, and Zsolt Tuza. Complexity and approximation for Traveling Salesman Problems with profits. *Theoretical Computer Science*, 531(0):54 – 65, 2014.

[ABSH14] Claudia Archetti, Nicola Bianchessi, Maria Grazia Speranza, and Alain Hertz. The split delivery capacitated team orienteering problem. *Networks*, 63(1):16–33, 2014.

[AFHS09] Claudia Archetti, Dominique Feillet, Alain Hertz, and Maria Grazia Speranza. The capacitated team orienteering and profitable tour problems. *JORS*, 60(6):831–842, 2009.

[AHS07] Claudia Archetti, Alain Hertz, and Maria Grazia Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, 2007.

[ASS06] Claudia Archetti, Martin W. P. Savelsbergh, and Maria Grazia Speranza. Worst-Case Analysis for Split Delivery Vehicle Routing Problems. *Transportation Science*, 40(2):226–234, 2006.

[AMN98] Esther M. Arkin, Joseph S. B. Mitchell, and Giri Narasimhan. Resource-constrained Geometric Network Optimization. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, pages 307–316, New York, NY, USA, 1998. ACM.

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of Finding Embeddings in a K-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, April 1987.

[Aro98] Sanjeev Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems. *J. ACM*, 45(5):753–782, September 1998.

[AKTT97] Tetsuo Asano, Naoki Katoh, Hisao Tamaki, and Takeshi Tokuyama. Covering points in the plane by k-tours: towards a polynomial time approximation scheme for

general k. In *ACM symposium on Theory of computing (STOC)*, pages 275–283, 1997.

[Bak94] Brenda S. Baker. Approximation Algorithms for NP-complete Problems on Planar Graphs. *J. ACM*, 41(1):153–180, January 1994.

[BBCM04] Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *ACM symposium on Theory of computing (STOC)*, pages 166–174, 2004.

[BCE+11] MohammadHossein Bateni, Chandra Chekuri, Alina Ene, Mohammad T. Hajiaghayi, Nitish Korula, and Dániel Marx. Prize-collecting Steiner Problems on Planar Graphs. In *Proceedings of the Symposium on Discrete Algorithms*, SODA '11, pages 1028–1049. SIAM, 2011.

[BT97] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[BCK+03] Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 46–, Washington, DC, USA, 2003. IEEE Computer Society.

[BC14] Adrian Bock and Alfonso Cevallos. Orienteering on planar graphs. Technical report, in preparation, EPFL, 2014.

[BGKS11] Adrian Bock, Elyot Grant, Jochen Könemann, and Laura Sanità. The School Bus Problem on Trees. In *ISAAC*, pages 10–19, 2011.

[BGKS13] Adrian Bock, Elyot Grant, Jochen Könemann, and Laura Sanità. The School Bus Problem on Trees. *Algorithmica*, 67(1):49–64, 2013.

[BS13] Adrian Bock and Laura Sanità. On the approximability of two capacitated vehicle routing problems. *Electronic Notes in Discrete Mathematics*, 41:519–526, 2013.

[BS14] Adrian Bock and Laura Sanità. Capacitated Orienteering. *submitted*, 2014.

[Bod96] Hans L. Bodlaender. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.*, 25(6):1305–1317, December 1996.

[BFG07] Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5(3):211–230, 2007.

[BHN04] Björn Brodén, Mikael Hammar, and Brengt J. Nilsson. Online and Offline Algorithms for the Time-Dependent TSP with Time Zones. *Algorithmica*, 39(4):299–319, 2004.

[BC94] Steven E. Butt and Tom M. Cavalier. A Heuristic for the Multiple Tour Maximum Collection Problem. *Comput. Oper. Res.*, 21(1):101–111, January 1994.

[CGW96] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.

[CGRT03] Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees, and minimum latency tours. In *Foundations of Computer Science (FOCS)*, pages 36 – 45, 2003.

[CKP12] Chandra Chekuri, Nitish Korula, and Martin Pál. Improved Algorithms for Orienteering and Related Problems. *ACM Trans. Algorithms*, 8(3):23:1–23:27, July 2012.

[CK04] Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *PROC. OF APPROX, SPRINGER LNCS*, pages 72–83, 2004.

[CP05] Chandra Chekuri and Martin Pál. A Recursive Greedy Algorithm for Walks in Directed Graphs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 245–253, Washington, DC, USA, 2005. IEEE Computer Society.

[CH08] Ke Chen and Sariel Har-Peled. The Euclidean Orienteering Problem Revisited. *SIAM Journal on Computing*, 38(1):385–397, 2008.

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[DM10] Aparna Das and Claire Mathieu. A quasi-polynomial time approximation scheme for Euclidean capacitated vehicle routing. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 390–403, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[DDS92] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *OPERATIONS RESEARCH*, 40(2):342–354, 1992.

[DLSS88] Martin Desrochers, Jan K Lenstra, Martin WP Savelsbergh, and Francois Soumis. Vehicle routing with time windows: optimization and approximation. *Vehicle routing: Methods and studies*, 16:65–84, 1988.

[FHR07] Jittat Fakcharoenphol, Chris Harrelson, and Satish Rao. The K-traveling Repairmen Problem. *ACM Trans. Algorithms*, 3(4), November 2007.

[Fei10] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

[FDG05] Dominique Feillet, Pierre Dejax, and Michel Gendreau. Traveling Salesman Problems with Profits. *Transportation Science*, 39(2):188–205, 2005.

[FL01] Fedor V. Fomin and Andrzej Lingas. Approximation Algorithms for Time-Dependent Orienteering. In *Fundamentals of Computation Theory*, LNCS 2138, pages 508–515. Springer, 2001.

[FHK76] Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation algorithms for some routing problems. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:216–227, 1976.

[FW07] Greg N. Frederickson and Barry Wittman. Approximation algorithms for the traveling repairman and speeding deliveryman problems with unit-time windows. In *In APPROX-RANDOM, volume 4627 of LNCS*, pages 119–133. Springer, 2007.

[FS14] Zachary Friggstad and Chaitanya Swamy. Approximation Algorithms for Regret-Bounded Vehicle Routing and Applications to Distance-Constrained Vehicle Routing. *Symposium on Theory of Computing (STOC)*, 2014.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GLV87] Bruce L. Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[GMNR11] Inge Li Gørtz, Marco Molinaro, Viswanath Nagarajan, and R. Ravi. Capacitated Vehicle Routing with Non-uniform Speeds. In *Integer Programming and Combinatoral Optimization (IPCO)*, LNCS 6655, pages 235–247. Springer, 2011.

[GNR11] Inge Li Gørtz, Viswanath Nagarajan, and R. Ravi. Minimum Makespan Multi-vehicle Dial-a-Ride. *CoRR*, abs/1102.5450, 2011.

[GRSZ13] Fabrizio Grandoni, R. Ravi, Mohit Singh, and Rico Zenklusen. New approaches to multi-objective optimization. *Mathematical Programming*, pages 1–30, 2013.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[GKNR12] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Approximation Algorithms for Stochastic Orienteering. In *Symposium on Discrete Algorithms (SODA)*, pages 1522–1538, 2012.

[HRK85] M. Haimovich and Alexander H. G. Rinnooy Kan. Bounds and Heuristics for Capacitated Routing Problems. *MATHEMATICS OF OPERATIONS RESEARCH*, 10(4):527–542, 1985.

[Hoc82] Dorit S. Hochbaum. Approximation Algorithms for the Set Covering and Vertex Cover Problems. *SIAM J. Comput.*, 11(3):555–556, 1982.

[HP98] Dorit S. Hochbaum and Anu Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.

[KR92] Marisa G. Kantor and Moshe B. Rosenwein. The Orienteering Problem with Time Windows. *The Journal of the Operational Research Society*, 43(6):pp. 629–635, 1992.

[KLS13] Marek Karpinski, Michael Lampis, and Richard Schmied. New Inapproximability Bounds for TSP. *CoRR*, abs/1303.6437, 2013.

[KS12] Marek Karpinski and Richard Schmied. On Approximation Lower Bounds for TSP with Bounded Metrics. *CoRR*, abs/1201.5821, 2012.

[Kle05] Philip N. Klein. A linear-time approximation scheme for planar weighted TSP. In *FOCS*, pages 647–657, 2005.

[KV12] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 5th edition, 2012.

[LLM91] Martine Labbe, Gilbert Laporte, and Helene Mercure. Capacitated Vehicle Routing on Trees. *OPERATIONS RESEARCH*, 39(4):616–622, 1991.

[LDN84] Gilbert Laporte, Martin Desrochers, and Yves Nobert. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, 14(1):161–172, 1984.

[LSLD92] Chung-Lun Li, David Simchi-Levi, and Martin Desrochers. On the Distance Constrained Vehicle Routing Problem. *Operations Research*, 40(4):pp. 790–799, 1992.

[Mit99] Joseph S. B. Mitchell. Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems. *SIAM J. Comput.*, 28(4):1298–1309, March 1999.

[NR12] Viswanath Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, 2012.

[Pap77] Christos H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237 – 244, 1977.

[PK10] Junhyuk Park and Byung-In Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311 – 319, 2010.

[Sav85] Martin W. P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1):285–305, 1985.

[SV12] András Sebö and Jens Vygen. Shorter Tours by Nicer Ears:. *CoRR*, abs/1201.1870, 2012.

[Sit02] René Sitters. The Minimum Latency Problem Is NP-Hard for Weighted Trees. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 230–239. Springer, 2002.

[Sit14] René Sitters. Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. In *SODA*, pages 604–616, 2014.

[Spa04] Michela Spada. *Sur la planification de tournées de véhicules scolaires*. PhD thesis, EPFL, Lausanne, 2004.

[SBL05] Michela Spada, Michel Bierlaire, and Thomas M. Liebling. Decision-Aiding Methodology for the School Bus Routing and Scheduling Problem. *TRANSPORTATION SCIENCE*, 39(4):477–490, 2005.

[TLZO01] Kay Chen Tan, Loo Hay Lee, Kenny Qili Zhu, and Ke Ou. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering*, 15(3):281 – 295, 2001.

[TMH05] Hao Tang and Elise Miller-Hooks. A {TABU} search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379 – 1407, 2005.

[Tha95] Sam R. Thangiah. Vehicle Routing with Time Windows using Genetic Algorithms, 1995.

[Tho14] Franka Tholen. The Traveling Salesman Problem with deadlines and its subproblems. Master's thesis, EPFL, Lausanne, Switzerland, 2014.

[TV01] Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[Tov84] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85 – 89, 1984.

[Tsi92] John N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:263–282, 1992.

[TSP] The Traveling Salesman Problem. http://www.math.uwaterloo.ca/tsp/.

[VSO11] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10, 2011.

[Vaz01] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

Adrian Bock

CONTACT INFORMATION

Route de Genève 101
1026 Denges
Switzerland

office: +41 21 693 27 39
mobile: +41 76 622 96 24
e-mail: adrianaloysius.bock@epfl.ch

EDUCATION

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

PhD candidate, Mathematics **March 2010 – present**

- Thesis Topic: Approximation algorithms for regret minimization in vehicle routing problems
- Advisor: Professor Friedrich Eisenbrand

University of Bonn, Bonn, Germany

Diploma in Mathematics **February 2010**

SCIENTIFIC PUBLICATIONS

(1) A. Bock, L. Sanità, *Capacitated Orienteering*, submitted.

(2) A. Bock, Y. Faenza, C. Moldenhauer, A. Ruiz Vargas, *Approximating stable set in terms of the odd cycle packing number*, submitted.

(3) A. Bock, K. Chandrasekaran, J. Könemann, B. Peis, L. Sanità, *Finding small stabilizers for unstable graphs*, Integer Programming and Combinatorial Optimization (IPCO), 2014.

(4) A. Bock, L. Sanità, *On the approximability of two capacitated vehicle routing problems*, International Network Optimization Conference (INOC), 2013.

(5) A. Bock, E. Grant, J. Könemann, L. Sanità, *The School Bus Problem on Trees*, Algorithmica, 2012.

(6) A. Bock, E. Grant, J. Könemann, L. Sanità, *The School Bus Problem on Trees*, International Symposium on Algorithms and Computation (ISAAC), 2011.

RESEARCH VISITS

University of Waterloo, Kitchener-Waterloo, Ontario, Canada

Hosts: Laura Sanità, Jochen Könemann **July 2013**

University of Waterloo, Kitchener-Waterloo, Ontario, Canada

Host: Laura Sanità **April 2012 – Mai 2012**

TEACHING EXPERIENCE

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Teaching Assistant **March 2010 – present**

- Fall 2013
  - Lineare Algebra (main assistant)
  - master thesis project on Orienteering with deadlines
- Spring 2013
  - Optimisation discrète (main assistant)
  - semester project on time tabling (ctd.)
- Fall 2012
  - Algèbre linéaire pour GC, MX (main assistant)
  - semester project on time tabling
- Fall 2011
  - Combinatorial Optimization (main assistant)
  - semester project on real root finding

- Spring 2011
  - Optimisation discrète (main assistant)
  - semester project on Support Vector Machines
- Fall 2010
  - Algèbre linéaire pour GC, SIE (assistant)
- Spring 2010
  - Algèbre linéaire II pour mathématiciens (assistant)
  - Discrete Optimization (assistant)

Technische Universität Berlin, Berlin, Germany
Teaching Assistant **March 2012 – August 2012**

- Spring 2012
  - Algorithmic (real-) algebraic geometry

Professional Experience

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Assistant-doctorant **September 2012 – present**

Technische Universität Berlin, Berlin, Germany
Wissenschaftlicher Mitarbeiter **March 2012 – August 2012**

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Assistant-doctorant **March 2010 – February 2012**

Research Institute for Discrete Mathematics, Bonn, Germany
Studentische Hilfskraft (SHK) **October 2008 – February 2010**

Siemens Corporate Technology, Munich, Germany and Beijing, China
Werkstudent **2005 – 2007**