Approximate Extended Formulations for Multidimensional Knapsack and the Unsplittable Flow Problem on Trees

by

Noah J.B. Weninger

B.Sc., The University of Alberta, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

August 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Approximate Extended Formulations for Multidimensional Knapsack and the Unsplittable Flow Problem on Trees

submitted by **Noah J.B. Weninger** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

F. Bruce Shepherd, Computer Science, UBC
Supervisor

Joseph Paat, Sauder School of Business, UBC
Supervisory Committee Member

Abstract

In this thesis, we study the Multidimensional Knapsack Problem (MKP) and two closely related special cases: the Unsplittable Flow Problem on Trees (UFPT), and the Unsplittable Flow Problem on Paths (UFPP). For these problems, when the natural integer programming formulation is relaxed to a linear program, the integrality gap is $O(n)$. Previous work on UFPT and UFPP has established that the addition of rank constraints to the linear program can be effective in some cases at reducing the integrality gap. However, Friggstad and Gao proved that even with all rank constraints added, the integrality gap of UFPT is $\Omega(\sqrt{\log n})$. Faenza and Sanità showed that a formulation for these problems which approximates the integer hull arbitrarily well must either have exponentially many facets or be described as an extended formulation (i.e., described in a higher dimensional space). We are interested in polynomially sized formulations, so we focus our study on extended formulations.

Our first contribution is a greedy algorithm which finds an optimal solution to the linear programming relaxation for the special case of UFPT where all requests share a common endpoint. We apply this to tighten the analysis of hard instances described in the literature. Following this, we describe two approximate extended formulations for MKP using disjunctive programming. The first is a formulation which improves upon the integrality gap of the linear relaxation using only a small number of extra variables and constraints. The second is a polyhedral $(1 - \epsilon)$-approximate extended formulation for MKP for any $0 < \epsilon \leq 1$, which was originally given by Pritchard. We then introduce a new hierarchy of strengthened formulations for MKP, which we study in the context of UFPT. The strengthened formulations are NP-Hard to separate over, but they can be $(1 - \epsilon)$-approximated using the aforementioned result. We conclude by evaluating the strength of this hierarchy when applied to the known hard instances from the literature. Our results suggest that this hierarchy may be most useful in conjunction with the rank constraints, but open questions remain.

Lay Summary

Knapsack problems are about finding the best way to select items given some capacity constraints. For example, consider the problem of finding which items to pack in a knapsack so that it isn't too heavy and the items you pack offer the greatest benefit. This thesis examines problems like this, but focuses on the case where there are multiple constraints, such as a limit on the number of cars on each street of a road network. Many of these problems are considered too difficult to solve exactly, so our focus is on finding fast ways to achieve good approximate solutions. Our approximate solutions are of a particular form: they are points on the surface of high-dimensional convex shapes. Describing the solutions in this way has many advantages which we use to derive new ways of approximating these problems.

Preface

The ideas behind the proof in Section 1.4.1 were developed with input from F. Bruce Shepherd. The results in Section 2.2.1 and Chapter 4 are the product of a paper co-authored with Adam Jozefiak and F. Bruce Shepherd, which is currently being prepared for publication. Section 3.2 is a generalization of a result by Daniel Bienstock and Benjamin McClosky [BM12]. Section 3.3 is a result by David Pritchard [Pri10], with the presentation modified for clarity. The remainder of this document is original and unpublished work by the author, Noah Weninger.

Table of Contents

List of Figures

List of Algorithms

Notation

1_S	The vector with $(1_S)_i = 1$ if $i \in S$, and $(1_S)_i = 0$ if $i \notin S$
αP	$\{\alpha x : x \in P\}$
a	Item size
$a^T b$	$\sum_i a_i b_i$
c	Knapsack capacity
$\text{conv}(S)$	The convex hull of S
d	Demand
$E(T)$	The edges of the tree T
$[k]$	$\{1, 2, \ldots, k\}$
$K_I(e)$	The KP instance arising from a single edge e of a UFPT instance
$K_I(S)$	The MKP instance arising from a set of edges S from a UFPT instance
m	Number of constraints or dimensions
n	Number of items, requests, or variables
$\frac{p}{q}$	If used in the context of a vector, this is the vector with all components equal to $\frac{p}{q}$ and with size inferred from context
P_r	The edges of the path that request r routes on
$\mathbb{R}$	Real numbers
$\text{rank}(S)$	The size of the largest cardinality feasible subset of S
R_e	The set of requests that route on edge e

$R(T)$	The requests with an endpoint in the tree T
u	Edge capacity
$V(T)$	The vertices of the tree T
w	Profit
$x(S)$	$\sum_{i \in S} x_i$
$\mathbb{Z}$	Integers

Glossary

IP	Integer Program
KP	Knapsack Problem
LP	Linear Program
m**-KP**	m-dimensional Knapsack Problem
MKP	Multidimensional Knapsack Problem
UFPP	Unsplittable Flow Problem on Paths
UFPT	Unsplittable Flow Problem on Trees

Acknowledgments

There are many people without whom this would not have been possible. First and foremost, I thank my supervisor Bruce Shepherd for his exceptional guidance and encouragement, my family and friends for their unwavering support, and my second reader Joseph Paat. I am grateful to Zachary Friggstad, Ryan Hayward, and Abram Hindle for their kindness, teaching, and encouragement, which led me to pursue graduate studies. This work was funded by the National Sciences and Engineering Research Council of Canada (NSERC), Huawei Canada, and UBC's Department of Computer Science, for which I am very thankful.

Chapter 1

Introduction

The primary motivation of this thesis is the development of improved approximation algorithms for the unsplittable flow problem on trees (UFPT). This well-studied NP-hard problem is of theoretical interest as well as practical: UFPT has many applications including bandwidth allocation, caching, optical networks, resource allocation, and scheduling [CMS07, GMWZ18]. In the context of approximation algorithms, UFPT is intriguing because the best known polytime result is an $O(\log^2 n)$-approximation, but the existence of an $O(1)$-approximation has not been ruled out. We also consider the special case of UFPT called the unsplittable flow problem on paths (UFPP), which concerns path graphs rather than trees. For UFPP, polytime $O(1)$-approximations are known and it remains open to find a polynomial-time approximation scheme (PTAS). Both UFPT and UFPP have been extensively studied. These problems can be seen as generalizations of the 0-1 knapsack problem (KP), or as special cases of the 0-1 multidimensional knapsack problem (MKP). Many of our results are, in fact, given in the more general setting of MKP, and hence the results may be useful on general MKP instances, or for other special cases of MKP. We begin by discussing some preliminaries before introducing these problems, our contributions, and related work.

1.1 Preliminaries

We assume the reader is familiar with the basics of the theory of linear and integer programming, including the existence of optimal extreme point solutions, duality, and complementary slackness, as well as basic literacy in the topics of graph theory, algorithmics, and computational complexity.

We consider only maximization problems, that is, problems where all feasible solutions are associated with some profit value, and our objective is to find a feasible solution of maximum profit. To clarify what we mean by an approximation algorithm in this context,

suppose that for some problem, we have an algorithm A which finds a solution $A(\mathcal{I})$ of maximum profit for any instance $\mathcal{I}$, and an algorithm B which finds a feasible solution $B(\mathcal{I})$ which is not necessarily optimal. For $\alpha \geq 1$, we say that B is an α-approximation algorithm if for all instances $\mathcal{I}$, the profit of $A(\mathcal{I})$ is at most α times the profit of $B(\mathcal{I})$. Due to inconsistencies in the literature, we also use the term α-approximation algorithm when $0 < \alpha \leq 1$ to mean that the profit of $B(\mathcal{I})$ is at least α times the profit of $A(\mathcal{I})$ for all instances $\mathcal{I}$. In either case, when $\alpha = 1$ the algorithm is exact, and the quality of approximation decreases as α becomes further away from 1.

As the title suggests, our approach involves the use of approximate extended formulations. Extended formulations are not needed for the material in this chapter or in Chapter 2 and are introduced in Chapter 3.

1.2 Knapsack

We begin by defining the *0-1 knapsack problem* (KP), the most fundamental problem we study. This section also introduces some general definitions that are used throughout the paper.

An instance $\mathcal{I} = (a, c, w)$ of this problem consists of a capacity c and n items, with item i having some size a_i $(0 < a_i \leq c)$ and nonnegative profit w_i. The objective is to select a subset of the items which achieves the maximum possible total profit and has total size at most the capacity. We can formulate this by the following integer program (IP), which encodes whether to select item i by the 0-1 values x_i. We write $w^T x$ to mean the dot product of vectors v and x, that is, $w^T x = \sum_{i=1}^{n} w_i x_i$. We also define $[k] = \{1, 2, \ldots, k\}$.

$$\begin{aligned} \max \quad & w^T x \\ \text{such that} \quad & a^T x \leq c \\ & 0 \leq x_i \leq 1 \qquad \forall\, i \in [n] \\ & x \in \mathbb{Z}^n. \end{aligned} \tag{K_I}$$

In general, for some formulation P (such as K_I), we say that x is in the associated *feasible region* of P if x satisfies the constraints but is not necessarily optimal. This may also be denoted by saying $x \in P$. The optimal value of some instance $\mathcal{I}$ for a formulation P is written $\mathrm{OPT}_P(\mathcal{I})$ and may be shortened to OPT_P or OPT when the instance or formulation is clear from context. Note that OPT is a scalar, not a vector, since $\mathrm{OPT} = \max\{w^T x : x \in P\}$.

Solving KP, or equivalently, optimizing over K_I, is NP-Hard [PKP13], hence our focus on approximations. To find approximations, it is common to consider the linear programming

(LP) *relaxation* K_L of K_I, where the constraint $x \in \mathbb{Z}^n$ is replaced by $x \in \mathbb{R}^n$. We sometimes call this the *natural* LP relaxation because it is derived from the IP without any additional constraints added. The number of variables and constraints in this LP is polynomial in n and thus it is solvable in polynomial time using standard linear programming algorithms.

$$\begin{aligned} \max \quad & w^T x \\ \text{such that} \quad & a^T x \leq c \\ & 0 \leq x_i \leq 1 \qquad \forall\, i \in [n] \\ & x \in \mathbb{R}^n. \end{aligned} \tag{K_L}$$

Evidently, for any instance $\mathcal{I}$, the feasible region of K_I is a subset of the feasible region for K_L and so $\mathrm{OPT}_{K_L} \geq \mathrm{OPT}_{K_I}$. To get an approximation algorithm for K_I, we find an optimal solution x for K_L, and then apply some polynomial time algorithm A which "rounds" x into a feasible solution $A(x) \in \mathbb{Z}^n$ for K_I. This two step method (solving the LP relaxation to optimality and transforming the output into an integral solution) is a common technique and has been shown to work well on the 0-1 knapsack problem.

Consider some integer program I and corresponding linear relaxation L (such as K_I and K_L). To show that this technique is effective we have two concerns: (1) that OPT_I is "close" to OPT_L, and (2) that we can find an algorithm A such that for any x which is optimal for K_L, A produces a vector $A(x) \in \mathbb{Z}^n$ that has value "close" to the value of x. We formalize (1) with the notion of *integrality gap*: we say that a formulation has an integrality gap of α on a set S of instances if for $\mathcal{I} \in S$, $\mathrm{OPT}_L(\mathcal{I})/\mathrm{OPT}_I(\mathcal{I}) \leq \alpha$. To formalize (2) we say that the *rounding ratio* of some algorithm A is the supremum of $(w^T x)/(w^T A(x))$ over all instances. If we find a rounding algorithm which achieves a rounding ratio equal to the integrality gap α, then we have an α-approximation algorithm since $\mathrm{OPT}_I(\mathcal{I}) \leq \mathrm{OPT}_L(\mathcal{I}) = w^T x \leq \alpha w^T A(x)$.

We show that the integrality gap of K_I is 2 and that a simple algorithm realizes the ideal rounding ratio of 2. Therefore, this technique gives a 2-approximation. This result is well known (e.g., see [PKP13]). To show this we first describe a greedy algorithm which produces a solution which is optimal for K_L, as a means to analyze OPT_{K_L}.

Lemma 1. *Algorithm 1.1 produces a vector x which is optimal for K_L.*

The proof of this lemma is omitted. We instead prove a more general version of this statement in Section 1.4. However, the following result is easy to show and is key to deriving the 2-approximation algorithm.

Lemma 2. *Let x be the output of Algorithm 1.1 on some instance. There is at most one index i such that $0 < x_i < 1$ (i.e., x_i is fractional).*

Order items such that $w_1/a_1 \leq w_2/a_2 \leq \cdots \leq w_n/a_n$
Let $x \in \mathbb{R}^n$
$r \leftarrow c$
for $i = 1, \ldots, n$ **do**
    $x_i \leftarrow \min(1, \max(0, r/a_i))$
    $r \leftarrow r - a_i\, x_i$
end
return x

Algorithm 1.1: The greedy algorithm for 0-1 knapsack.

Proof. Since the x_i are set such that $0 \leq x_i \leq 1$ and each $a_i > 0$, we know r is monotonically non-increasing in the loop. As long as $r \geq a_i$, x_i is set to 1. If we reach an iteration p such that $r < a_p$, we set x_p to r/a_p, and then decrease r to $r - a_p(r/a_p) = 0$. Hence, on all iterations $i > p$, $r = 0$ so x_i is set to 0. □

Theorem 1. *The integrality gap of K_I is at most* 2.

Proof. Let x be the optimal solution for K_L returned by Algorithm 1.1. If no index p (as described in Lemma 2) exists, then $x \in \mathbb{Z}^n$ and thus x is optimal for both K_L and K_I, meaning the integrality gap is 1 for this instance. Otherwise, if p exists, then let x^0 be the vector with $x^0_p = 1$ and $x^0_i = 0$ for $i \neq p$. Let x^1 be x but with $x^1_p = 0$. Since $x^0, x^1 \in \mathbb{Z}^n$, both are feasible for K_I and therefore $w^T x^0 \leq \mathrm{OPT}_{K_I}$ and $w^T x^1 \leq \mathrm{OPT}_{K_I}$. Therefore, $\mathrm{OPT}_{K_L} = w^T x \leq w^T(x^0 + x^1) \leq 2\mathrm{OPT}_{K_I}$, i.e., $\mathrm{OPT}_{K_L}/\mathrm{OPT}_{K_I} \leq 2$. □

The structure of this proof in fact immediately reveals how to create an algorithm to realize the rounding ratio of 2. We now know $w^T x^0$, $w^T x^1 \leq \mathrm{OPT}_{K_I} \leq w^T(x^0 + x^1)$, so the larger of $w^T x^0$ and $w^T x^1$ must be at least $\mathrm{OPT}_{K_I}/2$. Therefore, a 2-approximation is achieved by defining the rounding algorithm simply to return x^0 if $w^T x^0 > w^T x^1$ and return x^1 otherwise.

1.3 Multidimensional knapsack

In this section we generalize the 0-1 knapsack problem to have multiple knapsack constraints. This generalization can be thought of as adding more dimensions to the problem, where items can have a different size in each dimension and each dimension has its own capacity. For example, we may be concerned with maximizing profit while independently limiting the size, weight, and cost of items. A feasible set of items must not violate the capacity of any dimension. This problem is also known as the multi-constrained knapsack problem.

An instance $\mathcal{I} = (a, c, w)$ of the 0-1 m-dimensional knapsack problem (m-KP or MKP) consists of m nonnegative vectors of item sizes $a^j \in \mathbb{R}^n$ along each dimension $j \in [m]$,

scalars c^j denoting the capacity of each dimension $j \in [m]$, and a vector $w \in \mathbb{R}^n$ of item profits. Our goal is to select a subset of the items which maximizes profit while satisfying the knapsack constraint $\sum_{i=1}^{n} a_i^j x_i \leq c^j$ for all $j \in [m]$. Note that when $m = 1$ this is equivalent to the 0-1 knapsack problem.

$$\begin{aligned} \max \quad & w^T x \\ \text{such that} \quad & \textstyle\sum_{i=1}^{n} a_i^j x_i \leq c^j \qquad \forall\, j \in [m] \\ & 0 \leq x_i \leq 1 \qquad\qquad \forall\, i \in [n] \\ & x \in \mathbb{Z}^n. \end{aligned} \tag{m-K_I}$$

We denote the LP relaxation of this IP by m-K_L.

To simplify analysis, we make a few assumptions about the instances we examine. We assume that for each dimension j and item i, $a_i^j \leq c^j$, i.e., that each item can be feasibly packed by itself. We also assume that it is not possible to feasibly pack all items, i.e., we do not have $\sum_{i=1}^{n} a_i^j \leq c^j$ for every dimension j, since otherwise the problem is trivial.

MKP is evidently a very general problem. It can be thought of as a general 0-1 IP but with the restriction that all coefficients in the constraint matrix are positive. The additional assumptions we make only eliminate degenerate or trivial instances and do not impact the generality of the formulation. Hence, we might expect solving MKP to be nearly as hard as solving general 0-1 IPs. Nonetheless, there is some structure we can exploit.

A simple greedy algorithm analogous to Algorithm 1.1 which solves m-K_L to optimality is not known. However, a generalization of Lemma 2 is well-known and can be proved by standard LP theory [PKP13].

Lemma 3. *There exists an optimal solution x to m-K_L which has at most m fractional variables (variables x_i such that $0 < x_i < 1$).*

Proof. We assume that $m < n$; otherwise the statement holds trivially. First, we rewrite m-K_L in standard form by introducing slack variables y and z.

$$\begin{aligned} \max \quad & w^T x \\ \text{such that} \quad & Ax + y = b \\ & x + z = 1 \\ & x, y, z \geq 0 \\ & x, z \in \mathbb{R}^n \\ & y \in \mathbb{R}^m. \end{aligned}$$

Let (x, y, z) be a basic feasible solution to this LP. The LP has $n+m$ constraints and $2n+m$ variables. So, in (x, y, z) there are at most $n + m$ non-zero variables and at least n zero variables. Among the zero variables at least $n - m$ are part of either x or z, since $y \in \mathbb{R}^m$. For any variable $x_i = 0$, it must be that $z_i = 1$, so neither is fractional. Similarly, when $z_i = 0$, it must be that $x_i = 1$. So, of the x_i, at least $n - m$ must be zero or one, leaving at most m of them with fractional values. □

We can now easily arrive at a result analogous to Theorem 1.

Theorem 2. *The integrality gap of m-K_I is at most $m + 1$.*

Proof. Let $\hat{x}$ be a basic feasible solution to m-K_L. Let $I = \{i : \hat{x}_i = 1\}$ and $F = \{i : 0 < \hat{x}_i < 1\}$. Let x^I be the vector described by I, i.e. $x^I_i = 1$ if $i \in I$ and $x^I_i = 0$ otherwise. For every $j \in F$, define a vector x^{F_j} with $x^{F_j}_i = 1$ if $i = j$ and $x^{F_j}_i = 0$ otherwise. We know that $|F| \leq m$ by Lemma 3, so there are at most $m+1$ vectors in the set $S := \{x^I, x^{F_1}, \ldots, x^{F_{|F|}}\}$. All item sizes are positive, $x^I \in m$-K_I, and since we assume each item is feasible by itself, $x^{F_j} \in m$-K_I for each j. So, for $x \in S$, $w^T x \leq \text{OPT}_{m\text{-}K_I}$. Furthermore, the sum $x' = \sum_{x \in S} x$ has $x'_i \geq \hat{x}_i$ for every i, so $w^T \hat{x} \leq w^T x'$. Since $|S| \leq m+1$, we have $w^T x' \leq (m+1)\text{OPT}_{m\text{-}K_I}$. Therefore, $w^T \hat{x}/\text{OPT}_{m\text{-}K_I} \leq m + 1$ so the integrality gap is $m + 1$. □

Generalizing from KP, we can easily conclude with an $m+1$ approximation algorithm: solve the LP relaxation m-K_L, then choose the highest profit vector from the set S as defined in the proof of Theorem 2. Since the sum of all $m + 1$ vectors has profit at least $\text{OPT}_{m\text{-}K_I}$, the best vector must have profit at least $\text{OPT}_{m\text{-}K_I}/(m + 1)$.

We have now seen how to use linear programming to $(m + 1)$-approximate m-KP. However, this approximation ratio is far from ideal and in Chapter 3 we improve upon it. This concludes our basic introduction to knapsack problems, which is sufficient background to understand our results. For a more comprehensive discussion please see [PKP13].

1.4 Unsplittable flow on paths and trees

The unsplittable flow problems on paths (UFPP) and trees (UFPT) are generalizations of KP and special cases of MKP. Since UFPP is a special case of UFPT, we use the term UFPT to refer generally to the two problems. Unlike KP and MKP, we frame UFPT in the context of routing demands along paths in a tree, rather than packing items into a container. The unsplittable flow problem was initially introduced as the problem of routing flow along a single (i.e., unsplittable) path in a general graph, but we focus on the case where the graph is a tree. For further discussion about the history of this problem, please see Section 1.6.

An instance $\mathcal{I} = (T, R)$ of UFPT consists of an undirected capacitated tree $T = (V, E, u)$ and a set of requests R, defined as follows. V is the set of vertices and each edge $e \in E$ has some positive capacity u_e. Each request $r \in R$ imposes some nonnegative demand d_r on all edges along the unique simple path P_r between $s_r \in V$ and $t_r \in V$. We denote the number of requests $|R|$ by n and the number of edges $|E|$ by m. For simplicity, we assume that $R = \{1, \ldots, n\} = [n]$ and that $s_r \neq t_r$ for all $r \in R$. We say that a subset $S \subseteq R$ of requests is *feasible* or *routable* if, for each edge $e \in E$, the total demand of all requests $r \in S$ such that $e \in P_r$ is at most the capacity u_e of that edge. The goal is to select a feasible subset $S \subseteq R$ which maximizes the profit $\sum_{r \in S} w_r$. We formalize this with the following IP.

$$\begin{aligned} \max \quad & w^T x \\ \text{such that} \quad & \textstyle\sum_{r \in R \,:\, e \in P_r} d_r x_r \leq u_e && \forall\, e \in E \\ & 0 \leq x_r \leq 1 && \forall\, r \in R \\ & x \in \mathbb{Z}^n. \end{aligned} \qquad \text{UFPT}_I$$

The LP relaxation UFPT_L of UFPT_I is defined by replacing $x \in \mathbb{Z}^n$ with $x \in \mathbb{R}^n$. UFPP has an identical formulation, with the only difference being that T must be a path, that is, the degree of every vertex must be 2, except for the two end vertices which have degree 1.

For simplicity, we make the same assumptions as for m-K_I. We assume that all requests are routable on their own, i.e., for each $r \in R$ and $e \in P_r$, $d_r \leq u_e$. Any request which does not satisfy this condition cannot be included in any feasible solution, so it can be discarded. We also assume that it is impossible to route all requests together, as the optimal solution would then be trivial.

In terms of generality and approximability, UFPT is harder than KP and easier than MKP. When T consists of only a single edge, UFPT reduces to KP. UFPT can be thought of as MKP but with the restriction that for each item i and dimension j, $a_i^j \in \{0, d_i\}$ for some $d \in \mathbb{R}_+^n$, and the sizes a_i^j which are nonzero for a particular item i must correspond to the edges of a simple path. UFPT is also closely related to 0-1 column restricted packing integer programs (CPIP), which are more general, lacking the additional constraints of which sizes can be nonzero [CEK09].

Both UFPT and UFPP have been extensively researched. Important special cases of importance are those where all items have equal profit (*unit-profit* instances), when all requests route on some common edge (*intersecting* instances), and when all requests share a common endpoint (*single-sink* instances). Other cases have been studied less extensively, such as when the input graph is a cycle or star[1] or when the profit of each request equals

[1]In the case of a star, the problem generalizes the maximum weight matching problem: make a leaf for each item in the matching, make a request connecting each pair of items that can be matched, and set all

its demand.

Similarly to MKP, we refer to the constraints for each edge e as the knapsack constraint for e. We refer to the corresponding knapsack instance for a given edge in by $K_I(e)$, and the corresponding MKP instance for a set S of edges by $K_I(S)$. Hence, $K_I(E) = \text{UFPT}_I$. Like MKP, we do not have a simple greedy algorithm for solving the LP relaxation of UFPT on paths or trees. However, for single-sink instances, which we discuss extensively, we show that such an algorithm exists.

1.4.1 LP optimum for single-sink UFPT instances

In this section we describe a greedy algorithm which finds an optimal solution to UFPT_L in the single-sink case, where all requests share a common endpoint vertex r. As far as we know, such an algorithm has not been described in the literature. We use this result in Chapter 2 to tighten the analysis of the integrality gap of some well-known UFPT instances.

Assume T is rooted at r, with the root at the top of the tree. Note that by nature of single-sink instances, we may assume that if edge e_1 lies below edge e_2 in the rooted tree, then $u(e_1) < u(e_2)$. If this is not the case, then we could contract e_1 to a single vertex without affecting the feasible region. To analyze the algorithm we use a slightly different but equivalent formulation for UFPT, where variables x_i range from 0 to d_i rather than from 0 to 1.

$$\begin{aligned} \max \quad & \textstyle\sum_i w_i x_i \\ \text{such that} \quad & \textstyle\sum_{i \in R_e} x_i \le u_e \qquad \forall e \in E \\ & 0 \le x_i \le d_i \qquad \forall i \in R \end{aligned} \tag{UFPT'_I}$$

This LP has the following dual.

$$\begin{aligned} \min \quad & \textstyle\sum_i d_i z_i + \sum_e u_e y_e \\ \text{such that} \quad & z_i + \textstyle\sum_{e \in P_i} y_e \ge w_i \qquad \forall i \in R \\ & 0 \le z, y \end{aligned} \tag{UFPT'_I-Dual}$$

Thus, the complementary slackness conditions are:

$$x_i > 0 \implies z_i + \textstyle\sum_{e \in P_i} y_e = w_i, \tag{1}$$

$$y_e > 0 \implies \textstyle\sum_{i \in R_e} x_i = u_e, \tag{2}$$

$$z_i > 0 \implies x_i = d_i. \tag{3}$$

demands and capacities to 1.

We say that an edge e is *tight* if $\sum_{i \in R_e} x_i = u_e$. Assume WLOG that we have $w_1 \geq w_2 \geq \ldots \geq w_n$. We define a greedy algorithm to produce a primal feasible $\bar{x}$ and dual feasible $(\bar{y}, \bar{z})$, and then use complementary slackness to show that this solution is optimal.

```
1  x̄ ← 0
2  for i = 1 to n do
3  |  Increase x̄_i until some some edge becomes tight or until x̄_i = d_i
4  |  If an edge e first became tight on this iteration, define t_i = e
5  end
6  w' ← w
7  ȳ ← 0
8  z̄ ← 0
9  for i = n down to 1 do
10 |  if t_i is set then
11 |  |  ȳ_{t_i} ← w'_i
12 |  |  for j ∈ R : t_i ∈ P_j do
13 |  |  |  w'_j ← w'_j − w'_i
14 |  |  end
15 |  else if x̄_i = d_i then
16 |  |  z̄_i ← w'_i
17 |  end
18 end
19 return x̄, ȳ, z̄
```

Algorithm 1.2: The greedy algorithm for single-sink instances of UFPT'_I.

Theorem 3. *The vector $\bar{x}$ returned by Algorithm 1.2 is optimal for* UFPT'_I.

Proof. To show this, we prove that the vectors $\bar{x}, \bar{y}, \bar{z}$ returned by Algorithm 1.2 are feasible and satisfy complementary slackness. Note that since we assume the edge capacities are strictly decreasing away from the root, only one edge can become tight on each iteration.

First, it is easy to see that $\bar{x}$ is feasible for UFPT'_I, because elements $\bar{x}_i$ are ≥ 0 and only increased until some capacity constraint goes tight or until $\bar{x}_i = d_i$, which are exactly the conditions for feasibility.

Next we show complementary slackness conditions (2) and (3). If $\bar{z}_i > 0$ then clearly $\bar{x}_i = d_i$, since $\bar{z}_i$ is not set otherwise, so (3) is satisfied. If $\bar{y}_e > 0$ for some e, then $e = t_i$ for some i. Therefore, e is tight, i.e., $\sum_{i \in R_e} \bar{x}_i = u_e$ and hence (2) is satisfied.

We now prove that condition (1) holds and that $\bar{y}, \bar{z}$ is dual feasible. Consider some $i \in R$ and let $T = \{e \in P_i : e \text{ is tight}\}$. Since all tight edges are associated with some t_j, we can write $T = \{t_{j_1}, \ldots, t_{j_q}\}$ for some q and sequence $j_1, \ldots, j_q$. Furthermore, since $T \subseteq P_i$, we can assume that the edges $t_{j_1}, \ldots, t_{j_q}$ are ordered by decreasing distance from r. After some edge e goes tight, no edge in the subtree below e can go tight, because no

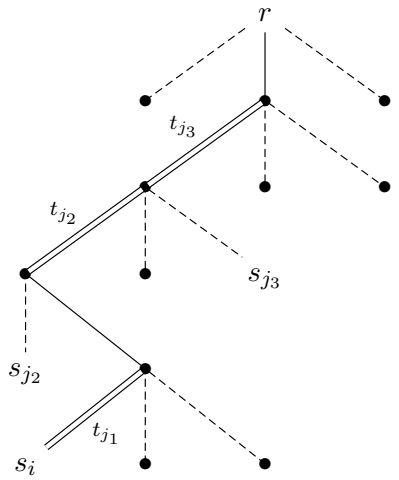

Figure 1.1: An example depicting the tight set T and path P_i for some request $i \in R$. The source of request i is s_i, the doubled edges (=) are those in T, and the dashed edges are those not in P_i. Request i made j_{t_1} go tight (so $i = t_1$), then request j_2 made t_{j_2} go tight, and finally request j_3 made t_{j_3} go tight.

more demand can be routed on e. Hence, t_{j_1} went tight first, and t_{j_q} went tight last, so we can assume $j_1 < \cdots < j_q$. See Fig. 1.1 for an illustration of this.

On any iteration ℓ not in $\{j_1, \ldots, j_q\}$, the values w'_{j_k} are not changed, because $t_\ell \notin P_{j_k}$ for any k; otherwise t_ℓ would be in T. Hence, w'_{j_q} is never modified before the iteration with $i = j_q$ of the loop on lines 9-17, so $\bar{y}_{t_{j_q}} = w_{j_q}$. We claim that on any iteration j_k with $k < q$ the algorithm sets $\bar{y}_{t_{j_k}} = w_{j_k} - w_{j_{k+1}}$.

On the iteration with $i = j_q$, the algorithm sets $w'_{j_k} \leftarrow w_{j_k} - w_{j_q}$ for all k. So, on iteration j_{q-1}, it assigns $\bar{y}_{t_{j_{q-1}}} = w'_{j_{q-1}} = w_{j_{q-1}} - w_{j_q}$ and sets

$$w'_{j_k} \leftarrow w'_{j_k} - w'_{j_{q-1}} = w_{j_k} - w_{j_q} - (w_{j_{q-1}} - w_{j_q}) = w_{j_k} - w_{j_{q-1}}$$

for all k. It follows by induction that on any iteration j_k with $k < q$ the algorithm sets $\bar{y}_{t_{j_k}} = w_{j_k} - w_{j_{k+1}}$ as desired.

Since $j_k < j_{k+1}$ for all $k < q$, we have $w_{j_k} - w_{j_{k+1}} \geq 0$, which is needed for dual feasibility. Furthermore, $\sum_{e \in P_i} \bar{y}_e = w_{j_q} + \sum_{k=1}^{q-1} w_{j_k} - w_{j_{k+1}} = w_{j_1}$. If t_i is defined, then $i = j_1$, because otherwise one of the other tight edges would be blocking request i from routing at all. So, $\bar{x}_i > 0$ and we have $\sum_{e \in P_i} \bar{y}_e = w_i$. Therefore, in this case, condition (1) is satisfied and $\bar{y}, \bar{z}$ is dual feasible. Assume instead that t_i is not defined. If $\bar{x}_i = d_i$, then the algorithm sets $\bar{z}_i = w_i - w_{j_1}$ by a similar argument to how we determined $\bar{y}_{t_k}$, and hence $\bar{z}_i + \sum_{e \in P_i} \bar{y}_e = w_i$, satisfying (1) and dual feasibility. Finally, if $\bar{x}_i = 0$, then we only need to check dual feasibility. We know $j_1 < i$ because request j_1 blocked request i, so $w_{j_1} \geq w_i$

and hence the solution is dual feasible. □

1.4.2 Rank formulations for UFPT

In this section, we discuss the use of *rank constraints* as a means for strengthening the LP formulation of UFPT. Intuitively, the rank of some set of requests is the size of the largest feasible subset, and a rank constraint enforces that limit.

Definition 1. *Consider some UFPT instance. For $S \subseteq R$ define*

$$\text{rank}(S) = \max\{|T| : T \subseteq S \text{ and } T \text{ is feasible}\}.$$

To motivate this, we make a simple observation. Let P be the feasible region of some IP. We say that some inequality $a^T x \leq b$ is a *valid inequality* for this IP if $\text{conv}(P) \subseteq \{x : a^T x \leq b\}$.

Observation 1. *Let $S \subseteq R$. The inequality $\sum_{i \in S} x_i \leq \text{rank}(S)$ is a valid inequality for UFPT.*

Hence, including all such inequalities is a valid formulation for UFPT, which we call the rank LP or rank formulation.

$$\begin{aligned} \max \quad & \textstyle\sum_i w_i x_i \\ \text{such that} \quad & \textstyle\sum_{r \in S\,:\, e \in P_r} d_i x_i \leq u_e && \forall\, e \in E \\ & \textstyle\sum_{i \in S} x_i \leq \text{rank}(S) && \forall\, S \subseteq R \\ & 0 \leq x_r \leq 1 && \forall\, r \in R \\ & x \in \mathbb{R}^n. \end{aligned} \qquad \text{Rank-UFPT}$$

Unfortunately, we cannot compute $\text{rank}(S)$ efficiently for general instances of UFPT because $\text{rank}(S)$ is itself the optimal value of an UFPT instance where all $w_i = 1$, which is known to be NP-hard to compute [CEK09]. Formally, $rank(S)$ is the optimal value of the following IP.

$$\begin{aligned} \text{rank}(S) = \max \quad & \textstyle\sum_i x_i \\ \text{such that} \quad & \textstyle\sum_{r \in S\,:\, e \in P_r} d_i x_i \leq u_e && \forall\, e \in E \\ & 0 \leq x_r \leq 1 && \forall\, r \in R \\ & x \in \mathbb{Z}^n. \end{aligned} \qquad \text{Card-UFPT}$$

The best known approximation for Card-UFPT, and hence for $\text{rank}(S)$, is an $O(\log |S|)$-approximation [CEK09]. Although it is hard to approximate $\text{rank}(S)$ for arbitrary sets S, it

turns out that for certain sets S we can compute $\text{rank}(S)$ efficiently, and that enforcing the rank constraints only for those sets is sufficient to solve the rank LP to within a constant factor. Specifically, Friggstad and Gao show that we only need to consider rank constraints for a collection $\mathcal{F}$ of sets S with $\text{rank}(S) = 1$ in order to 9-approximate Rank-UFPT [FG15]. We call these sets S with $\text{rank}(S) = 1$ *cliques*. They use this result to define a formulation called the compact rank LP. In their paper, $|\mathcal{F}| = O(nm)$, so this formulation is solvable in polytime.

$$\begin{aligned} \max \quad & \textstyle\sum_i w_i x_i \\ \text{such that} \quad & \textstyle\sum_{r \in S : e \in P_r} d_i x_i \le u_e && \forall\, e \in E \\ & \textstyle\sum_{i \in S} x_i \le 1 && \forall\, S \in \mathcal{F} \\ & 0 \le x_i \le 1 && \forall\, r \in R \\ & x \in \mathbb{R}^n. \end{aligned} \qquad \text{Compact-Rank-UFPT}$$

Theorem 4 (Theorem 5 from [FG15])**.** *If x is feasible for Compact-Rank-UFPT then $x/9$ is feasible for Rank-UFPT.*

In Chapter 2 we present a family of single-sink tree instances for which Rank-UPFT has integrality gap $\Omega(\sqrt{\log n})$. Furthermore, adding the rank constraints only improves the integrality gap by a constant factor over the natural LP relaxation for these instances. On the other hand, for intersecting path instances it is known that Rank-UFPT has integrality gap $O(1)$ [CEK09]. For general path instances, the integrality gap of Rank-UFPT is known to be $O(\log n)$, and it is conjectured to be $O(1)$ [CEK09].

Although the rank formulation does not improve significantly on the natural LP relaxation in the general case, there are some interesting properties of the rank LP which we present in Chapter 2.

1.5 Our contributions

In Chapter 2 we tighten results related to a class of UFPT instances introduced by Friggstad and Gao [FG15], determining exactly the optimal solutions to the IP and LP (see Theorem 6). To accomplish this, we also prove a number of new structural results about the instances, some of which are used in Chapter 4.

In Chapter 3 we generalize a result by Bienstock and McClosky [BM12] from knapsack to multidimensional knapsack. Our result is a linear extended formulation which performs better than the natural LP relaxation yet only has $O(n)$ times as many variables and constraints (see Theorem 9). Following this, we present a result by Pritchard [Pri10], which, for any $0 < \epsilon \le 1$, gives a $(1 - \epsilon)$-approximate formulation for multidimensional

knapsack which does not depend on the profits (see Theorem 10). We do not improve on Pritchard's result, but we have modified the presentation for clarity.

In Chapter 4 we introduce a new hierarchy of strengthened formulations for multidimensional knapsack problems called the *knapsack intersection hierarchy* and investigate its performance in the context of UFPT. Using results from Chapter 3 we show that the strengthened formulations can be approximately separated in polynomial time when the number of rounds is constant (see Corollary 2). We then prove that after applying t rounds of this hierarchy to the natural LP relaxation for UFPT, the integrality gap is $\Omega(n/t)$ (see Theorem 11). However, this result does not generalize to the rank LP and our results suggest that this new hierarchy may be effective where the rank LP is not: for the Friggstad-Gao instances, which evidence the worst known integrality gap for the rank LP, we prove that the hierarchy performs better, achieving an integrality gap of $\Theta(1/c)$ after n^c rounds for any $c > 0$ (see Theorem 12).

1.6 Related work

The name "unsplittable flow problem" was introduced in the doctoral thesis of Kleinberg [Kle96]. In the original definition, it is not required that the underlying graph is a tree. Hence, one not only has to determine which requests to include in the solution but also which path between terminals to route those requests on. This problem was also sometimes referred to as the integer multicommodity flow problem in operations research, a name which Kleinberg remarks did not have a standardized meaning in the literature, leading him to adopt the term unsplittable flow. The idea of routing flow on a single path can be further traced back to the *ring loading problem* introduced by Cosares and Saniee [CS94], which examined the case where the graph is a simple cycle.

The name unsplittable flow would go on to refer to a wide variety of problems as well, including the special cases of paths and trees, where all flows are inherently unsplittable (the route between two vertices cannot be "split" between multiple paths since there is only one such path). Additionally, the same name has come to refer to both the cases where the no-bottleneck assumption—which states that the maximum demand must be at most the minimum capacity—is assumed and where it is not (Kleinberg assumes it in the original definition, calling it the *balance assumption*).

The term *all-or-nothing flow* (ANF) was introduced to deal with this problem [CKS13]. It captures the essential property that for any request we must route all the demand ($= d_i$) or nothing ($= 0$). However, most recent papers still use the UFP terminology. On paths, the problem has also been studied under names such as resource allocation, bandwidth allocation, resource constrained scheduling, temporal knapsack, and interval packing [BSW14].

Although the name "unsplittable flow on trees" is not ideal, for consistency with the

recent literature, we adopt it. Specifically, we define the unsplittable flow problem on paths (UFPP) and the unsplittable flow problem on trees (UFPT) without the no-bottleneck assumption. The no-bottleneck case was effective settled in [CMS07].

We now discuss related work regarding UFPT and UFPP. The literature on integer programming hierarchies, extended formulations and disjunctive programming is discussed in Chapters 3 and 4.

1.6.1 UFPT

Currently, the best known approximation for UFPT is an $O(k \log n)$-approximation where k is the pathwidth of the tree [ACEW16]. All trees have pathwidth $O(\log n)$, so this is an $O(\log^2 n)$-approximation in general. Furthermore, this result holds for arbitrary submodular objectives, and implies an $O(\log n)$ approximation for UFPP since paths have constant pathwidth. Prior to this work, an $O(\log^2 n)$-approximation was first given by Chekuri, Ene, and Korula [CEK09]. Later, Friggstad and Gao matched this with an LP formulation for UFPT with integrality gap $O(\log^2 n)$ [FG15].

When all demands and item profits are 1 it is possible to obtain a 2-approximation for UFPT, and with arbitrary profits there is a 4-approximation [CMS07]. On the other hand, when all profits are 1 but demands are arbitrary, the best known result is an $O(\log n)$-approximation [CEK09]. This suggests that much of the difficulty in this problem comes from the fact that the requests can vary in demand.

UFPT is known to be APX-hard, thus ruling out a PTAS unless P = NP [GVY97]. However, there is a quasi-PTAS under some assumptions about the number of leaves and the magnitude of demands and capacities [ACH09]. It can be shown that UFPT is APX-hard when the number of leaves is $\Omega(n^\epsilon)$ for any $\epsilon > 0$ [ACH09].

1.6.2 UFPP

Since every path graph is a tree, the results from the previous seciton on UFPT all apply to UFPP. However, UFPT is known to be a harder problem, so better results are known for UFPP. Of the two problems, UFPP has received the most attention in the literature. At the time of writing, the best known polytime approximation for UFPP is a $(1 + \frac{1}{1+e} + \epsilon)$-approximation by Grandoni, Mömke, and Wiese [GMW20]. Prior to this, Grandoni, Mömke, and Wiese also published a $(5/3 + \epsilon)$-approximation, the first paper to break the $2 + \epsilon$ barrier, which stood for a number of years [GMWZ18]. Many recent results classify requests as either *small* or *large* depending on how large their demand is relative to the smallest capacity edge that they route on (the *bottleneck* edge). For instances that contain only large or only small requests, a PTAS is known, so a $(2 + \epsilon)$-approximation can be achieved by either discarding all small requests or all large requests [AGLW18]. To achieve

a $(5/3+\epsilon)$-approximation, a dynamic programming algorithm is used that is able to partially combine the solutions for only small and only large requests.

Prior to the $(2+\epsilon)$-approximation, an extended formulation with integrality gap $(7+\epsilon)$ was defined by embedding a dynamic program into an LP [AGLW13]. This is currently the best known LP formulation for this problem. Before this, a combinatorial $(7+\epsilon)$-approximation was already known [BSW14]. This was the first $O(1)$-approximation for UFPP; the best result prior to this is an $O(\log n)$-approximation—the first known $o(n)$ approximation—which uses dynamic programming [BFKS09]. An $O(n)$-approximation is easily obtained by rounding the natural LP relaxation (see Theorem 2).

It is not known whether a PTAS for UFPP exists, and resolving this is considered an important open problem [GMWZ18]. However, a quasi-PTAS is known, which suggests a PTAS may indeed exist since the existence of a quasi-PTAS rules out APX-hardness unless NP $\subseteq$ DTIME $\left(2^{\text{polylog}(n)}\right)$ [BCES06]. Currently, the fastest quasi-PTAS for UFPP runs in time $n^{O_\epsilon(\log\log n)}$ [GMW21].

Despite this open problem, a PTAS is known for many important special cases of UFPP. We already mentioned that there is a PTAS for instances with only large or only small requests. More formally, there is a PTAS if for some $\delta > 0$ the instance contains only requests which have demand at least a δ-fraction of the capacity of the smallest edge they route on [AGLW18]. On the other hand, if no requests fit this criteria (all requests are small), then there is also a PTAS [CMS07]. A PTAS is also known for instances where all requests cross an edge from a set of $O(1)$ edges, and hence for intersecting instances as well [GMWZ17]. Other cases which admit a PTAS include the case where the profit density of the requests falls within a constant sized range [BGK$^+$14], and in the case where all profits are 1 and the optimal profit is a fixed parameter [Wie17]. When the number of edges is constant, there is a PTAS via MKP (see Section 3.3).

Chapter 2

Hard Instances

In the literature on UFPT, two notable instances were introduced which serve to evidence lower bounds on the integrality gap of UFPT_L and Rank-UFPT. The first such instance S^n has T defined as a path graph on n vertices, for which UFPT_L has an integrality gap of $\Theta(n)$ but Rank-UFPT has an integrality gap of $O(1)$. The second is a tree T^h_{FG} on $\Theta(2^{h^2})$ vertices for which UFPT_L and Rank-UFPT both have an integrality gap of $\Theta(\sqrt{\log n})$. Hence, S^n evidences how poor of an approximation the LP relaxation is, while T^h_{FG} evidences that the rank LP is not ideal either.

In this section we first define S^n and prove some of its well-known properties. We then define T^h_{FG} and prove some new structural results and tightened bounds for it. These new results are key to our contributions in Chapter 4.

2.1 Staircase instances

For $n \geq 2$, we define the *staircase*[1] instance $S^n = (T, R)$ as follows. Let T be a path graph on n vertices, that is, $V = \{1, \ldots, n\}$ and $E = \{(1,2), (2,3), \ldots, (n-1, n)\}$. We refer to vertex 1 as the root or just r. For each $i = 1, \ldots, n-1$, define $u_{(i,i+1)} = 2^{i-1}$ and create a request i with $s_i = i$, $t_i = n$, $d_i = 2^{i-1}$, and $w_i = 1$. See Fig. 2.1 for an illustration. These instances were first described by Chakrabarti, Chekuri, Gupta, and Kumar [CCGK02].

Theorem 5. *On instances S^n, the integrality gap of UFPT_L is $n/2$, but the integrality gap of Rank-UFPT is 1.*

Proof. First we show that the integrality gap of the LP relaxation is $n/2$. Notice that each request routes on an edge with capacity equal to the demand of the request. So, since all requests share a common endpoint, for any two requests there must be an edge which both

[1] In the literature, this instance is referred to as a staircase because of a common way of visualizing UFPP instances where the capacity is plotted above the vertices on the Y axis.

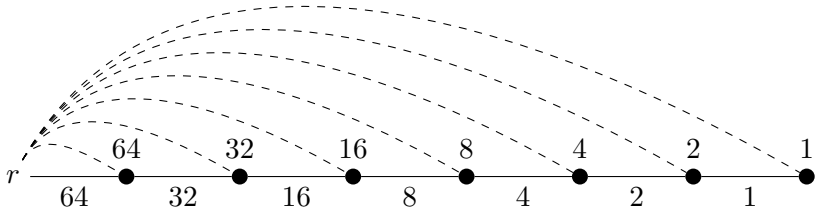

Figure 2.1: Instance S^8 is a path graph on 8 vertices. Each vertex marked with a bullet (•) is associated with a request (dashed line) which routes between that vertex and r. The value under each edge denotes the capacity of that edge, and the value above each vertex denotes the demand of the request associated with that vertex. All requests have profit 1.

requests route on, with capacity equal to the demand of the larger of the two requests. Hence, routing either of the requests does not leave enough capacity for the other, so any optimal integral solution must contain only a single request. All requests have profit 1, so therefore the optimal integral profit is 1.

We now show the LP optimum. Recall that the greedy algorithm in Section 1.4.1 selects requests in order of decreasing profit density. Request i has profit density 2^{1-i}, so the greedy algorithm starts by fully routing request 1, i.e., it sets $x_1 = 1$. Now, the remaining capacity on the edge with capacity 2 is 1; to fill this capacity we can route $1/2$ of the request with demand 2, i.e., we set $x_2 = 1/2$. Inductively for all other i, the edge with capacity 2^{i-1} is $1/2$ full when we assign to the request with demand 2^{i-1}, so we set $x_i = 1/2$ in order to fill the capacity. This gives a profit of $1 + (n-2)/2 = n/2$. Hence, the integrality gap of the LP relaxation is $(n/2)/1 = n/2$.

We now determine the rank LP integrality gap. We saw that no two requests can route together integrally. Thus, $\text{rank}(R) = 1$ so $\sum_i x_i \leq 1$ is a valid rank inequality. With this inequality there is clearly no assignment that produces a profit greater than 1, so the optimal profit for the rank formulation is 1, and hence the integrality gap of the rank formulation is also 1, i.e., the rank LP is equal to the integer hull. □

We can conclude from this that in general, the integrality gap of the LP relaxation is $\Omega(n)$. These instances evidence the worst known integrality gap for the natural LP relaxation, yet the rank formulation (in fact, only a single rank constraint) is equal to the integer hull. So, it is natural to question whether the integrality gap of the rank formulation is $O(1)$ in general. This is not currently known—for path instances the integrality gap is known to be $O(\log n)$, and is conjectured to be $O(1)$ [CEK09]. However, for tree instances, the

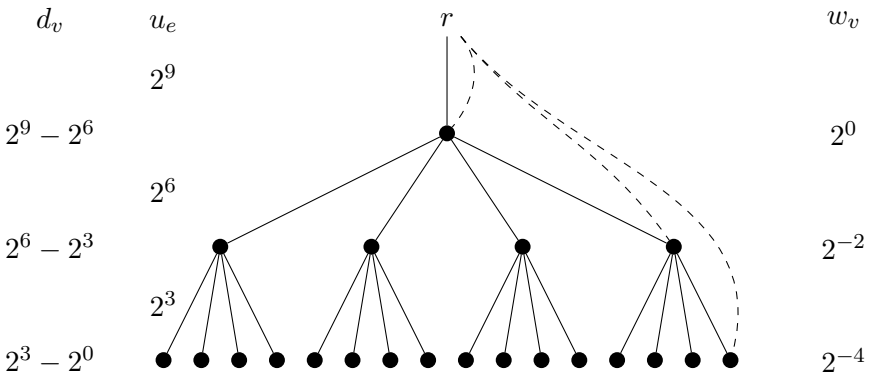

Figure 2.2: Instance T^3_{FG}. Each vertex marked with a bullet ($\bullet$) is associated with a request which terminates at r; a select few such requests are indicated by dashed lines. The values on the left indicate the demands/capacities of requests/edges in each level, respectively. The values on the right indicate the profit of the requests in each level.

integrality gap is $\Omega(\sqrt{\log n})$ [FG15].

2.2 Friggstad-Gao instances

In this section, we describe the family of Friggstad-Gao UFPT instances, and prove that they have an integrality gap of $\Theta(\sqrt{\log n})$ for both the LP relaxation and the rank formulation.

We define the tree T^h_{FG} with height $h \geq 2$ as follows. There is a root vertex r which has a single child v_1. Apart from r, all vertices have 2^{h-1} children. We denote the set of vertices with distance k from r by $level_k$, that is, $level_0 = \{r\}$, $level_1 = \{v_1\}$, and for $k \in [h]$, $|level_k| = 2^{(h-1)(k-1)}$. For each edge $e = uv$ with $u \in level_{k-1}$ and $v \in level_k$, define $u_e = 2^{h(h-k+1)}$. For all $k \geq 1$ and each vertex $v \in level_k$, create a request associated with v with $s_v = v$, $t_v = r$, demand $d_v = 2^{h(h-k+1)} - 2^{h(h-k)}$, and profit $w_v = 2^{-(h-1)(k-1)}$. See Fig. 2.2 for an example. This defines a *single-sink* instance since every request terminates at r. Moreover, since the profit of each request in any level is the inverse of the number of requests in that level, the total profit of requests in any level is exactly 1. A simple calculation shows that the number of requests—and equivalently the number of edges or number of vertices—in levels 0 through ℓ is

$$n(\ell) = \sum_{i=0}^{\ell-1} (2^{h-1})^i = \frac{2^{(h-1)\ell} - 1}{2^{h-1} - 1} = \Theta\left(2^{(h-1)(\ell-1)}\right). \tag{2.1}$$

Thus, $h = \Theta(\sqrt{\log n})$, where $n := n(h)$ is the total number of requests/edges, and for any ℓ, we have $\ell = \Theta\left(\frac{\log n(\ell)}{h}\right)$.

We denote by $K(e)$ the set of vectors $x \geq 0$ which satisfy the edge capacity constraint for e. Hence, a subset J of requests is routable if and only if $1_J \in K(e)$ for every edge e. For convenience, when referring to this instance, we often associate a request R_v routing between r and v with the vertex v or variable x_v. We write $T^{<v}$ to denote the subtree rooted at v with v itself removed. We use 1_S to denote the indicator vector of the set S, that is, the vector with $(1_S)_i = 1$ if $i \in S$ and $(1_S)_i = 0$ if $i \notin S$.

2.2.1 Basic properties

Our main results rely on the following lemmas.

Lemma 4. *For any edge $e = uv$ where $u \in level_{\ell-1}$ and $v \in level_\ell$ for some ℓ, the set of requests in $T^{<v}$ is routable on e. That is, $1_{T^{<v}} \in K_I(e)$.*

Proof. In any level $k > \ell$, $2^{h(h-k+1)}$ is an upper bound for the demand $2^{h(h-k+1)} - 2^{h(h-k)}$ of the requests. The number of vertices in $level_k$ that are also in the subtree below e is $2^{(h-1)(k-\ell)}$. Thus, the demand on e from routing all requests in $level_k$ is at most $2^{(h-1)(k-\ell)}2^{h(h-k+1)} = 2^{h^2-h\ell-k+\ell+h}$. Therefore, summing over all $k > \ell$, we have

$$\begin{aligned}\textstyle\sum_{k=\ell+1}^{h} 2^{h^2-h\ell-k+\ell+h} &= 2^{h^2-h\ell+\ell+h} \textstyle\sum_{k=\ell+1}^{h} 2^{-k} \\ &= 2^{h^2-h\ell+\ell+h}(2^{-\ell} - 2^{-h}) \\ &\leq 2^{h(h-\ell+1)} = u_e.\end{aligned}$$

□

Lemma 5. *The vector $\frac{1}{2}$ is in $K_I(e)$ for every edge e.*

Proof. Consider any edge $e = uv$ where $u \in level_{\ell-1}$ and $v \in level_\ell$. The only requests which route on e are those in the subtree rooted at v. Therefore it is sufficient to show that $b := \frac{1}{2}(1_{\{v\}} + 1_{T^{<v}}) \in K_I(v)$. Note that $1_{\{v\}}$ is in $K_I(e)$ since

$$d_v = 2^{h(h-\ell+1)} - 2^{h(h-l)} < 2^{h(h-\ell+1)} = u_e,$$

and by Lemma 4, we have that $1_{T^{<v}} \in K_I(e)$. It follows that any convex combination of these vectors, and hence b, lies in $K_I(e)$. □

Lemma 6. *Let r be the root of the tree T_{FG} and P be any path from r to a leaf. Then the demands for the requests of P form a routable set.*

Proof. The requests associated with level k have demand $2^{h(h-k+1)} - 2^{h(h-k)}$, so the total

demand of such a path is

$$\sum_{k=1}^{h} 2^{h(h-k+1)} - 2^{h(h-k)} = 2^{h^2} - 2^{h(h-1)} + 2^{h(h-1)} - 2^{h(h-2)} + \cdots + 2^{2h} - 2^h + 2^h - 2^0$$
$$= 2^{h^2} - 1.$$

This is less than 2^{h^2}, the capacity of the topmost edge. A similar argument shows that no other edges are violated by leveraging the self-similar structure of the tree. □

Lemma 7. *For any $\ell \geq 1$, any request $v \in level_\ell$, and any $k > \ell$, the set of requests*

$$\{v\} \cup \{u : u \in level_k \text{ is in the subtree below } v\}$$

is not feasible.

Proof. Recall that the demand of v is $2^{h(h-\ell+1)} - 2^{h(h-\ell)}$ and the demand of $u \in level_k$ is $2^{h(h-k+1)} - 2^{h(h-k)}$. The number of requests $u \in level_k$ which are in the subtree below v is $\left(2^{h-1}\right)^{k-\ell} = 2^{(h-1)(k-\ell)}$. So, in total the demand is

$$\begin{aligned}
&2^{h(h-\ell+1)} - 2^{h(h-\ell)} + 2^{(h-1)(k-\ell)} \cdot \left(2^{h(h-k+1)} - 2^{h(h-k)}\right) \\
&= 2^{h(h-\ell+1)} - 2^{h(h-\ell)} + 2^{h^2-h\ell+h+\ell-k} - 2^{h^2-h\ell+\ell-k} \\
&= 2^{h(h-\ell+1)} + 2^{h(h-\ell)} \cdot \left(2^{h+\ell-k} - 2^{\ell-k} - 1\right) \\
&> 2^{h(h-\ell+1)}.
\end{aligned}$$

Since $2^{h(h-\ell+1)}$ is the demand of the edge immediately above v, this set is not feasible. □

2.2.2 Integrality gap of the natural LP relaxation

In their paper, Friggstad and Gao proved an upper bound of 2 for the integral optimum and a lower bound of $h/2$ for the linear optimum on instances T^h_{FG}, implying an integrality gap of $h/4$ [FG15]. Here, we tighten these results by deriving closed form expressions for the optimal solutions to the IP and LP formulations, and hence for the integrality gap. We show that the true integral optimum is at most 1.5625 and the linear optimum is at least $(h+1)/2$, implying an integrality gap of $(h+1)/3.125$.

Theorem 6. *For instances T^h_{FG}, the integrality gap of the natural LP relaxation is*

$$\frac{\frac{(h+1)2^h-1}{2^{h+1}-2}}{1 + 2^{-(h-1)^2}\left\lfloor \frac{2^{h(h-1)}}{2^h-1} \right\rfloor} \geq \frac{h+1}{3.125} = \Theta(\sqrt{\log n}).$$

To prove this, we first establish the LP optimum and then the IP optimum.

Lemma 8. *The optimal profit of the LP relaxation for instances T_{FG}^h is*

$$\frac{(h+1)2^h - 1}{2^{h+1} - 2} \geq \frac{h+1}{2} = \Theta(\sqrt{\log n}).$$

Proof. In this proof we consider instances T_{FG}^h to be "hanging down" from the root vertex r, with the leaves at the bottom, as in Fig. 2.2. By Theorem 3, the LP optimum is achieved by an algorithm which greedily routes requests to the maximum (fractionally) feasible extent in order of decreasing profit density w_i/d_i. The solution we construct assigns the same value to every request in a given level, so we represent it by a vector $x \in \mathbb{R}^h$.

Recall that the root is in level 0, and the leaves are in level h. In level k of the tree, requests have profit $2^{-(h-1)(k-1)}$ and demand $2^{h(h-k+1)} - 2^{h(h-k)}$, so their profit density increases monotonically towards the leaves:

$$\frac{2^{-(h-1)(k-1)}}{2^{h(h-k+1)} - 2^{h(h-k)}} = \Theta\left(2^{k-1-h^2}\right).$$

By Lemma 4, we can route all the requests from any individual level together, so the greedy algorithm first assigns a value of 1 to each request in the lowest level of the tree ($k = h$), i.e., it sets $x_h = 1$.

By Lemma 7, we know that we cannot fully route any two levels of the tree together. Hence, the other levels must be fractionally routed. We now determine how much we can fractionally route the requests in level $h-1$, given how much capacity is occupied by the requests in level h. The requests in level h have demand $2^h - 1$. Since the branching factor is 2^{h-1}, the demand imposed by the requests in level h on the edges immediately above level $h-1$ is $2^{h-1} \cdot \left(2^h - 1\right)$. Since these edges have capacity $2^{h(h-(h-1)+1)} = 2^{2h}$, the amount of remaining capacity is $2^{2h} - 2^{h-1} \cdot \left(2^h - 1\right) = 2^{h-1} \cdot (2^h - 1)$.

By Lemma 4, even if all of these edges are tight, none of the other edges can be violated by these requests alone, so we only need to consider the capacity of these edges at this step. The requests in level $h-1$ have demand $2^{h(h-(h-1)+1)} - 2^{h(h-(h-1))}$, so to fill the remaining capacity the greedy algorithm assigns

$$x_{h-1} = \frac{2^{h-1} \cdot (2^h - 1)}{2^{h(h-(h-1)+1)} - 2^{h(h-(h-1))}} = \frac{2^h + 1}{2^{h+1} - 2}.$$

We now show by induction that for $k \leq h-2$, the greedy algorithm assigns $x_k = 2^{h-1}/(2^h - 1)$. Assume $k \leq h-2$ and that $x_{k+1} < 1$. This assumption holds at the base case of $k = h-2$, since $x_{h-1} < 1$. It also holds in the inductive case because $2^{h-1}/(2^h - 1) < 1$.

Since $x_{k+1} < 1$, the edges below level k must be tight; otherwise the greedy algorithm

would have assigned more value. Thus, the demand imposed so far on the edges above level k is equal to the branching factor 2^{h-1} times the capacity of the edges below level k. Simplifying, this is $2^{h-1} \cdot 2^{h(h-(k+1)+1)} = 2^{h(h-k+1)-1}$. Since the edges above level k have capacity $2^{h(h-k+1)}$, the amount of remaining capacity is $2^{h(h-k+1)} - 2^{h(h-k+1)-1} = 2^{h(h-k+1)-1}$. Hence, the requests in level k can be assigned value

$$x_k = \frac{2^{h(h-k+1)-1}}{2^{h(h-k+1)} - 2^{h(h-k)}} = \frac{2^{h-1}}{2^h - 1},$$

concluding the inductive argument.

Recall that in any level k, the profit of each request is $2^{-(h-1)(k-1)}$ and the number of requests is $2^{(h-1)(k-1)}$. Since we assign the same value to all requests in a level, the profit contributed by the requests in level k is $x_k \cdot 2^{-(h-1)(k-1)} \cdot 2^{(h-1)(k-1)} = x_k$, i.e., the profit of those requests is simply equal to the value assigned to them. Therefore, the total profit of the constructed solution is

$$\sum_{i=1}^{h} x_i = 1 + \frac{2^h + 1}{2^{h+1} - 2} + (h-2) \cdot \frac{2^{h-1}}{2^h - 1} = \frac{(h+1)2^h - 1}{2^{h+1} - 2} = \Theta(h). \qquad \square$$

Lemma 9. *The optimal integral solution profit for instances T^h_{FG} is*

$$1 + 2^{-(h-1)^2} \left\lfloor \frac{2^{h(h-1)}}{2^h - 1} \right\rfloor \leq 1.5625.$$

Proof. We consider instances T^h_{FG} to be "hanging down" from the root vertex r as in Fig. 2.2. To prove this, we show a stronger statement: for any request/vertex v which is in level k of the tree, the maximum profit of any feasible subset S of the requests in the subtree rooted at v is

$$2^{-(h-1)(k-1)} + 2^{-(h-1)^2} \left\lfloor \frac{2^{h(h-k)}}{2^h - 1} \right\rfloor.$$

The case $k = 1$ proves the lemma. The proof is by induction from $k = h$ to 1. For the base case where $k = h$, v is a leaf, so the subtree rooted at v contains only v itself. Since $h \geq 2$, we have

$$\left\lfloor \frac{2^{h(h-k)}}{2^h - 1} \right\rfloor = 0,$$

so it suffices to show that the profit of $S = \{v\}$ is at most $2^{-(h-1)(k-1)}$, and indeed, $w_v = 2^{-(h-1)(k-1)}$, concluding the base case. We now assume that $k < h$ and that the

claim holds for all children of v, and show that it holds for v itself. We consider two cases: where $v \in S$ and where $v \notin S$.

Case 1: $v \in S$. Here, we gain profit $2^{-(h-1)(k-1)}$ from v itself, so it remains to show that the rest of the requests in S contribute profit

$$2^{-(h-1)(h-1)} \left\lfloor \frac{2^{h(h-k)}}{2^h - 1} \right\rfloor.$$

After routing v, the amount of capacity remaining on the edge immediately above v is

$$2^{h(h-k+1)} - \left(2^{h(h-k+1)} - 2^{h(h-k)}\right) = 2^{h(h-k)}.$$

The requests in level h have the highest profit density (see Lemma 8), so filling the remaining capacity fractionally with requests from that level achieves the maximum possible fractional profit. We show, in fact, that filling the remaining capacity integrally with those requests is the optimal choice, even if there is a small amount of remaining capacity.

Recall that the requests in level h have demand $2^h - 1$. Level h contains $2^{(h-1)(h-1)}$ requests and $2^{h(h-k)}/(2^h - 1) \leq 2^{(h-1)(h-1)}$, so there are enough such requests to completely fill the remaining capacity. We add to S as many of these requests as possible while maintaining feasibility. However, since we need an integral solution, there may be some remaining capacity. Since the demand of requests in levels $< h$ is even higher than for level h, there is not enough capacity left to include any requests from other levels in S. Furthermore, removing any set of level h requests from S in order to include a request from a level $< h$ would reduce the profit of S: since demand increases by a factor of 2^h for each level towards the root, we would need to remove at least $2^h - 1$ level h requests from S to fit one request from level $h - 1$. If we performed this operation, the difference in profit would be at most

$$-(2^h - 1)2^{(h-1)(h-1)} + 2^{(h-1)(h-2)} < 0.$$

Therefore, including only v and as many level h requests as possible in S is optimal, and this produces the desired profit.

Case 2: $v \notin S$. By the induction hypothesis, for each of the 2^{h-1} children of v we can gain profit

$$2^{-(h-1)((k+1)-1)} + 2^{-(h-1)(h-1)} \left\lfloor \frac{2^{h(h-(k+1))}}{2^h - 1} \right\rfloor.$$

Therefore, the maximum profit of a feasible subset of the subtree rooted at v is

$$2^{h-1} \cdot \left(2^{-(h-1)((k+1)-1)} + 2^{-(h-1)^2} \left\lfloor \frac{2^{h(h-(k+1))}}{2^h - 1} \right\rfloor \right) \leq 2^{-(h-1)(k-1)} + 2^{-(h-1)^2} \left\lfloor \frac{2^{h(h-k)}}{2^h - 1} \right\rfloor$$

as desired. □

Proof of Theorem 6. From the previous two lemmas it is straightforward to show that the integrality gap is

$$\frac{\frac{(h+1)2^h - 1}{2^{h+1} - 2}}{1 + 2^{-(h-1)^2} \left\lfloor \frac{2^{h(h-1)}}{2^h - 1} \right\rfloor} \geq \frac{\frac{h+1}{2}}{1.5625} = \frac{h+1}{3.125} = \Theta(h) = \Theta(\sqrt{\log n}).$$ □

2.2.3 Integrality gap of the rank LP

It remains to show that the integrality gap of the rank formulation is at most a constant factor better than the natural LP relaxation, that is, the rank formulation also has integrality gap $\Theta(\sqrt{\log n})$. To show this we follow a similar proof to one presented by Friggstad and Gao [FG15].

Theorem 7. *For instances T^h_{FG}, the integrality gap of the rank LP is $\Theta(\sqrt{\log n})$.*

Proof. First, we show that given any set $S \subseteq R$, at least two of the requests in the set can be routed together. Let $i, j \in R$ be arbitrary requests and suppose $i \in level_a$ and $j \in level_b$, WLOG with $a < b$ (if $a = b$, then the requests could not conflict). If i and j were to conflict, they would do so on the parent edge of i because capacity increases towards the root. The capacity of this edge is $2^{h(h-a+1)}$, the demand of i is $2^{h(h-a+1)} - 2^{h(h-a)}$, and the demand of j is $2^{h(h-b+1)} - 2^{h(h-b)} \leq 2^{h(h-a)} - 2^{h(h-a-1)}$. Hence, the total demand is less than the capacity, so the two requests can route together. So, given any $S \subseteq R$, any two requests in S can route together and thus $\text{rank}(S) \geq 2$. Hence, there are no clique constraints for this instance and thus Compact-Rank-UFPT is equivalent to UFPT_L.

Therefore, the optimal solution x^* to UFPT_L, given in Lemma 8, is also optimal for Compact-Rank-UFPT. By Theorem 4, $x^*/9$ is feasible for Rank-UFPT, and hence the integrality gap of Rank-UFPT on this instance is $\Theta(\sqrt{\log n})/9 = \Theta(\sqrt{\log n})$. □

Instances T^h_{FG} evidence the worst known lower bound for Rank-UFPT, i.e., the integrality gap of Rank-UFPT is $\Omega(\sqrt{\log n})$. However, the integrality gap of Rank-UFPT is known to be $O(\log n)$ on general instances [CEK09]. It remains an open problem to close this gap, either by tightening the upper bound to $O(\sqrt{\log n})$ or finding an instance with integrality gap $\omega(\sqrt{\log n})$.

Chapter 3

Extended Formulations

In the previous section, we saw that the rank formulation for UFPT, while somewhat effective at strengthening the natural LP relaxation, still has an integrality gap of $\Omega(\sqrt{\log n})$. To improve this gap, one might consider generalizing the rank inequalities, for example to those of the form $\sum a_i x_i \leq b$ for $a_i \in \{0, 1, 2\}$. Although these directions may in fact lead somewhere, some negative results are known which suggest there are limits to this approach—which *extended formulations* can overcome.

Definition 2. *Let $w \in \mathbb{R}^n_+$ and $0 \leq \epsilon \leq 1$. A $(1-\epsilon)$-approximate extended formulation for $P \subseteq \mathbb{R}^n$ is a polyhedron $Q \subseteq \mathbb{R}^N$ with $N > n$, such that for all $x \in P$ there is some $x' \in \mathbb{R}^{N-n}$ such that $(x, x') \in Q$, and*

$$\max\{w^T x : x \in P\} \geq (1-\epsilon) \max\{w^T x : \exists x' \in \mathbb{R}^{N-n} \text{ s.t. } (x, x') \in Q\}.$$

We may also refer to $(1+\epsilon)$-approximate extended formulations where it is more convenient to do so. For $\epsilon > 0$ we have a $(1+\epsilon)$-approximation if

$$\max\{w^T x : \exists x' \in \mathbb{R}^{N-n} \text{ s.t. } (x, x') \in Q\} \leq (1+\epsilon) \max\{w^T x : x \in P\}.$$

Evidently, there is a $(1+\epsilon)$-approximation if and only if there is a $(1-\epsilon')$-approximation; just take $\epsilon' = \epsilon/(1+\epsilon)$ or $\epsilon = \epsilon'/(1-\epsilon')$. Since we only deal with maximization problems, this is not ambiguous; we can determine what is meant by x-approximation depending on whether $x > 1$. We motivate this definition using the following result of Faenza and Sanità [FS15].

Theorem 8. *For sufficiently small $\epsilon > 0$, there is no $(1-\epsilon)$-approximate formulation $Q \subseteq \mathbb{R}^n$ for a 0-1 knapsack polytope $P \subseteq \mathbb{R}^n$ such that the number of facets of Q is polynomial in n.*

Therefore, if we want to define a $(1-\epsilon)$-approximate formulation for KP which has a polynomial number of facets for every fixed ϵ, we need to use an extended formulation; no formulation in the original space $\mathbb{R}^n$ exists.

In this section we explore approximate extended formulations for multidimensional knapsack using a technique called disjunctive programming; in the next section we apply these formulations to derive strengthened formulations for UFPT.

3.1 Disjunctive programming

Let $P \subseteq [0,1]^n$ be an arbitrary set, $w \in \mathbb{R}^n_+$, and suppose we wish to solve $\max\{w^T x : x \in P\}$. We say that $Q^1, \ldots, Q^L \subseteq [0,1]^n$ is a *disjunction* for P if $P \subseteq Q^1 \cup \cdots \cup Q^L$. The Q^i are called the *disjuncts*. Clearly, $Q := \text{conv}(Q^1 \cup \cdots \cup Q^L) \supseteq P$ and therefore

$$\max\{w^T x : x \in P\} \leq \max\{w^T x : x \in Q\},$$

so Q is a relaxation for P. *Disjunctive programming* effectively amounts to enumerating a number of cases (the Q^i), but the enumeration is implicit—we end up with a single convex optimization problem with extra variables (an extended formulation). By the definition of $\text{conv}(\cdot)$, we can optimize over Q with the following formulation:

$$\begin{array}{lll} \max & w^T x & \\ \text{such that} & \sum_{i=1}^L \lambda^i x^i = x & \\ & \sum_{i=1}^L \lambda^i = 1 & \\ & \lambda^i \geq 0 & \forall\, i \in [L] \\ & x^i \in Q^i & \forall\, i \in [L] \end{array} \qquad \text{DP1}$$

However, this formulation is nonlinear because of the $\lambda^i x^i$ terms. This can be alleviated if we have a linear description of the Q^i, that is, if there is some A^i and b^i such that $Q^i = \{x \in [0,1]^n : A^i x \leq b^i\}$ for all i. If we have such a linear description, then we can linearize the optimization problem over Q by making each term $\lambda^i x^i$ into a variable which we denote by z^i. This is called *homogenization* and is achieved by adding an extra dimension z^i_0 for each z^i. Optimization over Q can thus be formulated as an LP which uses the extra

variables z^i:

$$\begin{array}{rll} \max & w^T x & \\ \text{such that} & \sum_{i=1}^{L} z_j^i = x_j & \forall\, j \in [n] \\ & \sum_{i=1}^{L} z_0^i = 1 & \\ & z_0^i \geq 0 & \forall\, i \in [L] \\ & A^i z^i \leq z_0^i b^i & \forall\, i \in [L] \\ & z_j^i \leq z_0^i & \forall\, i \in [L], \forall\, j \in [n] \end{array} \qquad \text{DP2}$$

Lemma 10. *DP1 and DP2 are equivalent.*

Proof. If we take $(x, \lambda) \in$ DP1, then we can define $z_0^i = \lambda^i$ and $z_j^i = \lambda^i x_j^i$ for $i \in [L]$ and $j \in [n]$, so that $(x, z) \in$ DP2. Conversely, for any $(x, z) \in$ DP2, we can take $\lambda^i = z_0^i$ for all i, so that $(x, \lambda) \in$ DP1. □

The following result is key for the application of disjunctive programming to approximation algorithms.

Lemma 11. *Let* $OPT = \max\{w^T x : x \in P\}$. *If there is a constant* $0 \leq \gamma < 1$ *such that for each* $i \in [L]$ *we have* $OPT \geq (1-\gamma)\max\{w^T x : x \in Q^i\}$, *then* $OPT \geq (1-\gamma)\max\{w^T x : x \in Q\}$.

Proof. Let (x, z) be optimal for DP2. Then for all $i \in [L]$, we have $z^i \in z_0^i\, Q^i$, so

$$z_0^i \,\mathrm{OPT} \geq z_0^i\,(1-\gamma)\max\{w^T x : x \in Q^i\} \geq (1-\gamma) w^T z^i.$$

If we sum this over all $i \in [L]$ we find, as desired, that

$$\mathrm{OPT} = \sum_{i=1}^{L} z_0^i \,\mathrm{OPT} \geq (1-\gamma) \sum_{i=1}^{L} w^T z^i = (1-\gamma) w^T x. \qquad \square$$

This simple result is enough for us to begin applying disjunctive programming. For a more comprehensive introduction to disjunctive programming, please refer to [Bal79]. We first give a simple disjunctive programming formulation for m-KP which improves upon the $(m+1)$-approximation seen in the introduction. Following this we present a polynomially sized $(1-\epsilon)$-approximate formulation for m-KP for any $0 < \epsilon \leq 1$.

3.2 Simple disjunctive programming approximation for m-KP

In this section we give a $(1+0.79m)$-approximate extended formulation for m-KP which uses just $O(n)$ disjuncts. This result is generalized from a 1.79-approximate extended formulation for KP given by Bienstock and McClosky [BM12]. Their result was provided in response to the question of whether there was any "simple" relaxation for KP with polynomially separable inequalities which has an integrality gap of less than 2 (the integrality gap of the natural LP relaxation). We show that their result generalizes very cleanly to m-KP. Their result uses only $O(1)$ disjuncts while we require $O(n)$, but the construction is still relatively simple.

Theorem 9. *There is a $(1 + m \cdot r(m))$-approximate extended formulation for m-KP with size polynomial in n and m, where*

$$r(m) = \frac{\sqrt{8m^4 + 28m^3 + 29m^2 + 10m + 1} - 3m - 1}{4m^2 + 2m} \leq 0.79.$$

If desired for simplicity, the analysis also works with $r = (\sqrt{19} - 2)/3$, but the above definition of r gives a tighter bound for large m, since $\lim_{m\to\infty} r(m) = \sqrt{2}/2$.

Proof. Fix some instance $\mathcal{I}$ and assume WLOG that the optimal profit OPT_L for m-K_L is $m+1$. If not, we can scale all the profits so that this is the case. Let OPT_I be the optimal profit for m-K_I. Since m is fixed, we write r instead of $r(m)$. As we showed in Section 1.3, the integrality gap of m-K_L is $m+1$, i.e., $\mathrm{OPT}_L/\mathrm{OPT}_I \leq m+1$, and therefore $\mathrm{OPT}_I \geq 1$. We now define an extended formulation relaxation for m-K_I of size polynomial in n and show that it has optimal profit at most $(1 + mr)\mathrm{OPT}_I$.

We divide the disjuncts we define into three types: Q^2_h, Q^1_h, and Q^0. Let $\Omega = \{i : w_i \geq r\}$ and $\Omega_h = \{i \in \Omega \setminus \{h\} : a^j_i + a^j_h \leq c^j \ \forall j \in [m]\}$. First, for each $h \in \Omega$ such that $\Omega_h \neq \emptyset$, we make a disjunct Q^2_h as follows:

$$\begin{aligned}
&\textstyle\sum_{i=1}^n a^j_i x_i \leq c^j && \forall\, j \in [m] \\
&x_h = 1 && \\
&\textstyle\sum_{i\in\Omega_h} x_i \geq 1 && \\
&0 \leq x_i \leq 1 && \forall\, i \in [n]
\end{aligned} \tag{Q^2_h}$$

Next, for every $h \in \Omega$ we make a disjunct Q_h^1:

$$\begin{array}{ll} \sum_{i=1}^n a_i^j x_i \le c^j & \forall\, j \in [m] \\ x_h = 1 & \\ x_i = 0 & \forall\, i \in \Omega \setminus \{h\} \\ x_i = 0 & \forall\, i \text{ where } \exists\, j \text{ s.t. } a_i^j + a_h^j > c^j \\ 0 \le x_i \le 1 & \forall\, i \in [n] \end{array} \tag{Q_h^1}$$

And finally we make a single disjunct Q^0:

$$\begin{array}{ll} \sum_{i=1}^n a_i^j x_i \le c^j & \forall\, j \in [m] \\ x_i = 0 & \forall\, i \in \Omega \\ 0 \le x_i \le 1 & \forall\, i \in [n] \end{array} \tag{Q^0}$$

We now show that this construction has the desired integrality gap. Let Q be the convex hull of the union of all of these disjuncts. We claim that $m\text{-}K_I \subseteq Q \subseteq m\text{-}K_L$. Let S be such that $1_S \in m\text{-}K_I$. If $|S \cap \Omega| \ge 2$, then $1_S \in Q_h^2$ for every $h \in S \cap \Omega$. If $|S \cap \Omega| = 1$, then it is easily verified that $1_S \in Q_h^1$ for $\{h\} = S \cap \Omega$. If $|S \cap \Omega| = 0$, then clearly $1_S \in Q^0$. So, $m\text{-}K_I \subseteq Q$. Now let $x \in Q$. Since all disjuncts have the constraints $\sum_{i=1}^n a_i^j x_i \le c^j$ for all $j \in [m]$, we have $x \in m\text{-}K_L$ by definition. So, $Q \subseteq m\text{-}K_L$.

If there is some $h \in \Omega$ such that $\Omega_h \neq \emptyset$, then for every $i \in \Omega_h$, $1_{\{h,i\}} \in m\text{-}K_I$. Furthermore, $1_{\{h,i\}}$ achieves profit at least $2r$ because $h, i \in \Omega$ so both h and i have profit at least r. Since $\mathrm{OPT}_L = m+1$, as desired the integrality gap of Q is at most $(m+1)/(2r) \le 1 + mr$.

Now we assume that $\Omega_h = \emptyset\ \forall h \in \Omega$, i.e., no two items of profit at least r can be packed together feasibly. So, we are only concerned with the disjuncts Q_h^1 and Q^0 from this point on.

Let $h \in \Omega$ be arbitrary and let $\hat{x}$ be an extreme point optimal solution to $\max\{w^T x : x \in Q_h^1\}$. Observe that if $w^T \hat{x} \le 1 + mr$, then the integrality gap is at most $1 + mr$ as desired, since $\mathrm{OPT}_I \ge 1$. Therefore, we assume $w^T \hat{x} > 1 + mr$.

Furthermore, observe that if for some i we have $w_i \ge (m+1)/(1+mr)$, then we are done because then packing item i alone implies that $\mathrm{OPT}_I \ge (m+1)/(1+mr)$, so the integrality gap of $m\text{-}K_L$—and hence the integrality gap of Q—is at most

$$\frac{m+1}{(m+1)/(1+mr)} = 1 + mr.$$

So, we assume from here on that $w_i < (m+1)/(1+mr)$. Now, since $\hat{x} \in Q_h^1$, $\hat{x}$ contains

exactly one nonzero item h from Ω, so we have the following:

$$\sum_{i\in\Omega} w_i\hat{x}_i = w_h < \frac{m+1}{1+mr},$$
$$\text{and} \qquad \sum_{i\in\Omega} a_i^j\hat{x}_i = a_h^j \qquad \forall\, j\in[m].$$

Let $\Gamma = \{i \notin \Omega : \forall j,\ a_i^j + \min_{k\in\Omega} a_k^j \le c^j\}$. Clearly $\Gamma\cap\Omega = \emptyset$. Recall that Q_h^1 has the constraint that $x_i = 0$ for all i such that $a_i^j + a_h^j > c^j$ for some j. For $i \notin \Gamma\cup\Omega$, there is some j such that $a_i^j + \min_{k\in\Omega} a_k^j > c^j$. Since $h\in\Omega$, we have $a_i^j + a_h^j \ge a_i^j + \min_{k\in\Omega} a_k^j > c^j$, so $\hat{x}_i = 0$. So, $\Gamma\cup\Omega = \{i : x_i > 0\}$. Therefore, since we assumed $w^T\hat{x} > 1+mr$, we have

$$\sum_{i\in\Gamma} w_i\hat{x}_i = w^T\hat{x} - \sum_{i\in\Omega} w_i\hat{x}_i > 1+mr-\frac{m+1}{1+mr},$$
$$\text{and} \qquad \sum_{i\in\Gamma} a_i^j\hat{x}_i = \sum_i a_i^j\hat{x}_i - \sum_{i\in\Omega} a_i^j\hat{x}_i \le c^j - a_h^j \qquad \forall\, j\in[m].$$

Since $\hat{x}\in m\text{-}K_L$, which has integrality gap $m+1$, we can use the rounding algorithm from Section 1.3 to find a feasible set $S\subseteq\Gamma$ which achieves profit at least $(\sum_{i\in\Gamma} w_i\hat{x}_i)/(m+1)$. So, there exists a set $S\subseteq\Gamma$ with $1_S\in m\text{-}K_I$ which satisfies the following:

$$\sum_{i\in S} w_i \ge \frac{1}{m+1}\left(1+mr-\frac{m+1}{1+mr}\right),$$
$$\text{and} \qquad \sum_{i\in S} a_i^j \le c^j - a_h^j \qquad \forall\, j\in[m].$$

There is enough space left along each dimension to pack item h along with S, so there is a feasible integral solution $S\cup\{h\}$ with profit at least

$$\frac{1}{m+1}\left(1+mr-\frac{m+1}{1+mr}\right) + r \ge \frac{m+1}{1+mr},$$

i.e., $\mathrm{OPT}_I \ge (m+1)/(1+mr)$. Note that the above inequality requires a significant amount of algebra to prove and depends on the value of r. We have omitted this for brevity. Since $w^T\hat{x} \le \mathrm{OPT}_L = m+1$, the integrality gap of Q_h^1 is $1+mr$.

Finally, if $Q_h^1 = \emptyset$ for all h, then $Q = Q^0$. In the definition of Q^0 we force $x_i = 0$ for all items with $w_i \ge r$. By Lemma 3 any extreme point optimal solution has at most m fractional items, so the profit of the LP is less than $\mathrm{OPT}_I + mr \le (1+mr)\mathrm{OPT}_I$. Therefore, the integrality gap of Q^0 is at most $1+mr$, as desired.

We have now shown that every disjunct has optimal value at most $(1+mr)\mathrm{OPT}_I$, so if we take $\gamma = mr/(1+mr)$, then by Lemma 11, Q itself also has optimal value at most

$(1+mr)\mathrm{OPT}_I$ and therefore is a $(1+mr)$-approximate extended formulation. □

3.3 Disjunctive programming $(1-\epsilon)$-approximation for m-KP

In this section we present a $(1-\epsilon)$-approximate extended formulation for m-KP for any $0 < \epsilon \leq 1$ which has size polynomial in n for fixed ϵ and m. This construction was first given by Pritchard [Pri10]. In his paper, Pritchard presents a non-polyhedral version of this result in detail and notes how to modify the construction to achieve a polyhedral approximation. Here, we present the full details of the polyhedral version. We do not improve upon the original result, but have modified the presentation for clarity. Our proof also borrows ideas from an earlier result for the 1-dimensional case by Bienstock [Bie08], which was the inspiration behind Pritchard's result.

This section partially answers a question posed by Van Vyve and Wolsey [VW06], which asked whether there exist $(1+\epsilon)$-approximate extended formulations for knapsack polytopes which have size polynomial in n and ϵ^{-1}. The formulation we present here is polynomial in n for each fixed ϵ but is not polynomial in ϵ^{-1}. It remains an open problem to find an extended formulation with size polynomial in both n and ϵ^{-1}, even for the 1-dimensional case. We discuss this further in the conclusion of this section.

In this section we denote the integral multidimensional knapsack polytope m-KP by P_I. The main result is the following:

Theorem 10. *Let $0 < \epsilon \leq 1$. There exists some $A \in \mathbb{R}^{s\times n}$, $A' \in \mathbb{R}^{s\times t}$, and $b \in \mathbb{R}^s$ with $t = O(n^{1+m^3\epsilon^{-1}})$ and $s = O(n^{1+m^3\epsilon^{-1}})$, such that*

$$P_I \subseteq P_\epsilon := \left\{x \in \mathbb{R}^n : \exists\, x' \in \mathbb{R}^t \text{ s.t. } Ax + A'x' \leq b\right\},$$

and for any $w \in \mathbb{R}^n_+$,

$$\max\left\{w^T x : x \in P_I\right\} \geq (1-\epsilon)\max\left\{w^T x : x \in P_\epsilon\right\}.$$

It is easy to see that the extended formulation P_ϵ has size polynomial in n for any fixed ϵ and m, and thus can be optimized over in polynomial time. Since P_ϵ does not depend on the profits w, we have the following additional desirable property. We show in Chapter 4 that this implies that the intersection of many polytopes P_ϵ is itself a $(1-\epsilon)$-approximation for the intersection of the corresponding integral polytopes P_I.

Corollary 1. *P_ϵ is a polyhedral $(1-\epsilon)$-approximation for P_I, that is,*

$$(1-\epsilon)P_\epsilon \subseteq P_I \subseteq P_\epsilon.$$

The idea behind this construction mirrors a well known approach for achieving a PTAS for multidimensional knapsack. In essence, we want to guess a set of the largest sized items that are assumed to be in the solution, and then solve the LP relaxation for the remaining small items. Since the LP only decides whether to include relatively small items, it can be rounded to an integral solution without much loss. As long as the size of the large item set we guess is constant, there are only polynomially many such sets to try, so the algorithm run in polytime.

We now describe how to construct P_ϵ, before proving that it achieves the desired approximation.

3.3.1 Construction

The construction uses disjunctive programming to effectively enumerate all possible guesses up to a certain size. Let H be an integer depending on ϵ and m which we define later. Our guesses are of size H.

First, we handle the cases where the number of items in the solution is smaller than H. To do this we create a disjunct Q^S_{small} for all sets $S \subseteq [n]$ such that $|S| < H$ and $1_S \in P_I$. These disjuncts are trivial: they only contain the single point 1_S. There are $O(n^H)$ such disjuncts.

Next, we create a disjunct $Q^{\mathcal{S}}_{large}$ for each set family $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ such that for all j, $|S_j| = H$ and $1_{S_1 \cup \cdots \cup S_m} \in P_I$. Hence, the number of these disjuncts is $O(n^{mH})$. $\mathcal{S}$ is analogous to the guess described above and $Q^{\mathcal{S}}_{large}$ is analogous to the LP for the remaining small items after the guessed items are fixed. For convenience we say $S = S_1 \cup \cdots \cup S_m$. Essentially, we want the feasible solutions for this disjunct to include all items in S, and exclude items not in S which have size larger than those in S_j for some dimension j. In other words, we only allow our guess S along with fractionally feasible items that are smaller on every dimension than the items we guessed. So, we define $Q^{\mathcal{S}}_{large}$ as follows:

$$x_i = 1 \qquad \forall\, i \in S \tag{3.1}$$

$$x_i = 0 \qquad \forall\, i \notin S : \exists\, j \; a^j_i > \min_{k \in S_j} a^j_k \tag{3.2}$$

$$x \in P \tag{3.3}$$

3.3.2 Analysis

First, we show that this disjunctive formulation is indeed a relaxation for P_I.

Lemma 12. *Let P_ϵ be the convex hull of the union of all disjuncts. Then $P_I \subseteq P_\epsilon$.*

Proof. For $x \in [0,1]^n$ define $\text{suppt}(x) = \{i : x_i = 1\}$. We claim that if $x \in P_I \cap \{0,1\}^n$, then x is in one of the disjuncts. If $|\text{suppt}(x)| < H$, then trivially $x \in Q_{small}^{\text{suppt}(x)}$. Otherwise, we show $x \in Q_{large}^{\mathcal{S}}$ for some $\mathcal{S} = \{S_1, \ldots, S_m\}$. For each dimension j, let S_j contain the H largest items in $\text{suppt}(x)$. Then, we have $x_i = 1$ for each $i \in S$, satisfying Eq. (3.1). For each dimension j, we picked S_j to contain the H largest items in $\text{suppt}(x)$, so no other item in $\text{suppt}(x)$ can have size greater than $\min_{k \in S_j} a_k^j$, and hence Eq. (3.2) is satisfied. Finally, Eq. (3.3) is trivially satisfied because $x \in P_I$. □

It remains to prove that this formulation has the desired integrality gap.

Lemma 13. *Let $w \in \mathbb{R}_+^n$ and let $OPT_I = \max\left\{w^T x : x \in P_I\right\}$. For any disjunct Q,*

$$OPT_I \geq (1-\epsilon)\max\{w^T x : x \in Q\}.$$

Proof. First consider the case of the small disjuncts. For any S, Q_{small}^S contains only the single point 1_S, which is in P_I by definition, so the inequality trivially holds.

Now we consider the large disjuncts $Q_{large}^{\mathcal{S}}$. Let $\mathcal{S}$ be arbitrary. It suffices to prove that for any extreme point optimal solution x for $\max\{w^T x : x \in Q_{large}^{\mathcal{S}}\}$, we have $\text{OPT}_I \geq (1-\epsilon)w^T x$. By Lemma 3, x contains at most m fractional elements. Let $X = \{i : x_i > 0\}$ be the set obtained by rounding up all of these fractional elements. For each dimension j, let $D_j = \arg\min\{w(D) : D \subseteq X,\ |D| = m,\ \sum_{i \in X \setminus D} a_i^j \leq c^j\}$. In other words, D_j is the smallest profit set of m items from X such that $X \setminus D$ satisfies the knapsack constraint for dimension j. Let $I = X \setminus (D_1 \cup \cdots \cup D_m)$ be the feasible set resulting from deleting all D_j from X.

Fractional elements of x are not in S and thus are subject to Eq. (3.2). So, for $i \in S_k$ and i' fractional, we have $a_i^j \geq a_{i'}^j$, i.e., the elements in S_j are all larger than the fractional elements. Deleting the fractional elements of x from X must yield a feasible set. Hence, for any $D \subseteq S_j$ with $|D| = m$, deleting D from X yields a set which is feasible for dimension j, that is, we have $\sum_{i \in X \setminus D} a_i^j \leq c^j$. Therefore, such sets D are among those considered by the arg min when selecting the sets D_j.

There must exist some such set D which has profit at most $\frac{m}{H} w(S_j)$, the average profit of m items from S_j. Furthermore, since $S_j \subseteq X$, we have $\frac{m}{H} w(S_j) \leq \frac{m}{H} w(X)$. Hence, each set D_j also has $w(D_j) \leq \frac{m}{H} w(X)$. There are m sets $D_1, \ldots, D_m$, so the profit of their union

is at most $\frac{m^2}{H} w(X)$. Therefore,

$$\mathrm{OPT}_I \geq w(I) \geq w(X) - \frac{m^2}{H} w(X) = \left(1 - \frac{m^2}{H}\right) w(X) \geq \left(1 - \frac{m^2}{H}\right) w^T x,$$

and taking $H = \lceil m^2/\epsilon \rceil$ finishes the proof. □

We can now conclude with Theorem 10.

Proof of Theorem 10. In Section 3.1, we saw how we can optimize over disjunctive programs using the linear formulation DP2, which is equivalent to optimizing over $P_\epsilon := \mathrm{conv}(Q^1 \cup \cdots \cup Q^L)$ for disjuncts Q^i. By Lemma 12, P_ϵ is a relaxation for P_I, and given Lemma 13 we can apply Lemma 11 to conclude that P_ϵ is indeed a $(1-\epsilon)$-approximation. From the construction it is clear that P_ϵ does not depend on the profits w as desired.

There are at most $\binom{n}{mH} = O(n^{mH})$ disjuncts $Q^{\mathcal{S}}_{large}$ and at most $\sum_{k=1}^{H} \binom{n}{k} = O(Hn^H)$ disjuncts Q^S_{small}, so the homogenized LP DP2 has $O((n+m)(n^{mH} + Hn^H)) = O(n^{1+m^3\epsilon^{-1}})$ constraints and $O(n^{1+mH} + Hn^{1+H}) = O(n^{1+m^3\epsilon^{-1}})$ variables. □

In comparison with the simple disjunctive programming formulation given in Section 3.2, this construction is significantly larger when ϵ is set to achieve a similar approximation factor. In Section 3.2 we achieved a $(1 + r(m) \cdot m)$-approximation, or equivalently, a $(1 - r(m) \cdot m/(1 + r(m) \cdot m))$-approximation. If we take $\epsilon = r(m) \cdot m/(1 + r(m) \cdot m)$, the construction here sets $H = O(m^2)$ and thus uses $O(n^{m^3})$ disjuncts, in stark contrast to the $O(n)$ disjuncts used by the simple formulation.

However, the construction in this section is a polyhedral approximation (i.e., not depending on the cost function; see Corollary 1), while the construction in Section 3.2 is not. There is a non-polyhedral variant of this construction given by Pritchard [Pri10], which requires only $O(n^m)$ disjuncts to achieve the same approximation factor. Still, the simple construction does better, which suggests that there may be some room for improvement here. In particular, it may be possible to use ideas from the simple formulation to find a $(1-\epsilon)$-approximate extended formulation with size polynomial in both n and ϵ^{-1}, answering the open question posed by Van Vyve and Wolsey [VW06].

3.4 An extended formulation for UFPT?

We have now seen a $(1-\epsilon)$-approximate extended formulation for MKP, and by extension for UFPT. However, the size of the formulation for UFPT that this approach gives is exponential in the number of edges. Since MKP is such a general problem, it is natural to question whether there is an extended formulation which exploits the structure of UFPT instances to get a $(1-\epsilon)$-approximate formulation with size polynomial in the number of

edges and requests. This would imply a PTAS, so given that UFPT is APX-hard, we do not expect this to be possible. However, UFPP is not known to be APX-hard, so a formulation may exist for path instances—although, we expect this to be quite hard to resolve, because no PTAS for UFPP is currently known despite the significant effort put towards finding one.

Chapter 4

The Knapsack Intersection Hierarchy

In this section we introduce a method of strengthening LP formulations for multidimensional knapsack which we call the *knapsack intersection hierarchy*. We study the application of this hierarchy to UFPT, but it is a general framework that may have applications in other settings.

Computational solution strategies for MKP often use some form of branch and cut method. One of the most effective approaches is to rely on cuts for the knapsack polytopes associated with individual constraints [CJP83]. For each $j \in [m]$, let $K(j)$ denote the polytope $\{x \in [0,1]^n : \sum_i a_i^j x_i \leq c^j\}$, i.e., the polytope associated with constraint j. The *knapsack cuts* for $K(i)$ are the inequalities which are valid for the integer hull $K_I(j) = \text{conv}(K(j) \cap \{0,1\}^n)$. On each iteration of a branch and cut approach (e.g., see [Sch98]), one has a feasible—but fractional—solution $\tilde{x}$ for a current relaxation P' of m-K_I. One then generates knapsack cuts for some constraint. That is, for some j, we find a valid inequality $b^T x \leq d$ for $K_I(j)$ for which $b^T \tilde{x} > d$. Adding these inequalities to P' gives a tighter formulation for m-K_I on which to recurse.

This approach has also been extended to *multi-row cuts*. This can be done in two distinct ways: (1) By aggregating multiple constraints to form a single knapsack and using it to generate cutting planes [DLTW14, Xav17, DM18] and (2) by considering cuts associated with the integer hull of the intersection of several knapsack polytopes [LW08, KPP04]. The latter set-up is potentially stronger in the following sense: there are instances where adding all cuts of type (2) defines the integer hull but adding (any number of) cuts of type (1) does not.[1]

[1] A well-known example for the Chvátal rank actually shows that one may need an unbounded number of rounds of aggregated cuts in order to obtain the integer hull (e.g., see Section 23 in [Sch98]).

We discuss a framework to measure the strength of cuts in the latter setting. For some $S \subseteq [m]$, we study the intersection of the knapsack polytopes $K(S) := \cap_{j \in S} K(j)$. The paradigm generates cuts for the associated integer hull $K_I(S)$. We define a *knapsack intersection hierarchy* of relaxations for P_I as follows. For each $t = 1, \ldots, m$, define

$$P^t := \bigcap_{|S|=t} K_I(S).$$

In other words, P^t is obtained from P by adding, for each $S \subseteq [m]$ with $|S| = t$, all valid inequalities for $K_I(S)$. Clearly, $P^{t+1} \subseteq P^t$ and $P^m = P_I$, so we have the following hierarchy:

$$P \supseteq P^1 \supseteq \ldots \supseteq P^{m-1} \supseteq P^m = P_I.$$

Since separating over $K_I(S)$ is NP-Hard, it is already hard to separate over P^1 (a fate shared by the Chvátal closure of a polyhedron [Eis99]). However, we show that a polynomially sized, but approximate, formulation can be used when t is constant.

This hierarchy appears to differ from other integer programming hierarchies in the sense that the levels of the hierarchy are parameterized by the number of constraints (m), rather than the number of variables (n). For example, the hierarchies introduced by Lovász-Schrijver [LS91], Sherali-Adams [SA90], Parillo [Par03] and Lasserre [Las01] are all equal to the integer hull P_I at level n. However, it can be shown that for any UFPT instance there exists an equivalent instance with $m \leq 4n$ (see Appendix A.3 in [FG15]), so our hierarchy is equal to the integer hull at level $4n$.

4.1 Primary results

We show how to approximate P^t using the formulation presented in Section 3.3, which gives a polyhedral $(1-\epsilon)$-approximate extended formulation for multidimensional knapsack problems. This leads to the following result:

Proposition 1. *For $0 < \epsilon \leq 1$, there is a $(1-\epsilon)$-approximate extended LP formulation for P^t of size $O(n^{t^3\epsilon^{-1}+t+1})$, i.e., a formulation for which the value of an optimal solution is at most a $(1/(1-\epsilon))$-factor larger than the optimal solution to P^t.*

Corollary 2. *For fixed t there is a PTAS for $\max\{w^T x : x \in P^t\}$.*

Let P_{rank} and P^t_{rank} be the polytopes P and P^t with all rank constraints added. We determine the integrality gap of P^t and P^t_{rank} for the "staircase" instances S^n and the Friggstad-Gao instances T^h_{FG} which we used to lower bound the integrality gap of LP formulations in Chapter 2. The following result establishes a lower bound using the instances S^n—for which P_{rank}, and hence P^t_{rank}, is known to have an integrality gap of $O(1)$.

Theorem 11. *The integrality gap of P^t is $\Omega(n/t)$, even for path graphs where all requests share a common endpoint.*

The proof of this result is simple, and possibly even expected, given that the same lower bound holds for the Sherali-Adams hierarchy [CEK09]. With a bit more work, we can also lower bound the integrality gap for tree instances by using the Friggstad-Gao instances. We were not able to resolve a general upper bound on the integrality gap for P^t; however, specifically for the Friggstad-Gao instances, we give an upper bound matching our lower bound.

Theorem 12. *For constant t, the integrality gap of both P^t and P^t_{rank} is $\Omega(\sqrt{\log n})$. However, on the Friggstad-Gao instances, for any $c > 0$ the integrality gap of both P^{n^c} and $P^{n^c}_{\text{rank}}$ is $\Theta(1/c)$.*

Unfortunately, for $t = o(n^c)$, the integrality gap of P^t_{rank} is super-constant, and hence this does not lead to a quasi-PTAS. The Friggstad-Gao instances have no cliques, that is, each pair of requests is routable. So, we speculate whether this improved integrality gap upper bound could hold for general tree instances which do not have any cliques, and whether the rank formulation could be effective against instances which do have cliques.

4.1.1 Proof of Proposition 1

Proof of Proposition 1. In Section 3.3, following the work of Pritchard [Pri10], a $(1-\epsilon)$-approximate extended formulation is given for $K_I(S)$ with size $O(n^{1+t^3\epsilon^{-1}})$. For $0 < \epsilon \leq 1$, denote the projection of this extended formulation onto $\mathbb{R}^n$ by $K_\epsilon(S)$. Furthermore, denote by P^t_ϵ the polytope $\bigcap_{|S|=t} K_\epsilon(S)$. Since $K_\epsilon(S)$ is a polyhedral approximation, as shown in Corollary 1, we have $(1-\epsilon)K_\epsilon(S) \subseteq K_I(S) \subseteq K_\epsilon(S)$. It follows that

$$(1-\epsilon)P^t_\epsilon = (1-\epsilon)\bigcap_{|S|=t} K_\epsilon(S) = \bigcap_{|S|=t}(1-\epsilon)K_\epsilon(S) \subseteq \bigcap_{|S|=t} K_I(S) = P^t$$

and

$$P^t = \bigcap_{|S|=t} K_I(S) \subseteq \bigcap_{|S|=t} K_\epsilon(S) = P^t_\epsilon.$$

Therefore, P^t_ϵ is a polyhedral $(1-\epsilon)$-approximate extended formulation for P^t. There are $\binom{n}{t} = O(n^t)$ sets S with $S = |T|$, so since each $K_\epsilon(S)$ has size $O(n^{1+m^3\epsilon^{-1}})tn^{O(t^2/\epsilon)}$, P^t_ϵ has size $O(n^{1+t^3\epsilon^{-1}}) \cdot O(n^t) = O(n^{t^3\epsilon^{-1}+t+1})$ as desired. □

4.2 Integrality gap lower bound

In this section, we prove Theorem 11 and the lower bound part of Theorem 12. In Section 4.2.1, we start by showing that the knapsack intersection hierarchy may be an ineffective approach for approximating the solution to certain path instances—for which adding the rank constraints yields a constant integrality gap. However, in contrast we show in Section 4.2.2 that on the Friggstad-Gao tree instances, for any $c > 0$ the integrality gap is reduced to $\Omega(1/c)$ for both P^{n^c} and $P^{n^c}_{\text{rank}}$, despite that the rank LP has integrality gap $\Omega(\sqrt{\log n})$ for these instances.

4.2.1 Path instances

For path instances, it is known that the integrality gap of P_{rank} is $O(\log n)$ and it is conjectured to be $O(1)$ [CEK09]. However, the natural LP formulation has an integrality gap of $\Omega(n)$, as evidenced by the staircase instances S^n. We now prove Theorem 11 by showing that the integrality gap of P^t is $\Omega(n/t)$.

Proof of Theorem 11. Let $t > 1$. We show that $\frac{1}{t+1} \in P^t$ for instances S^n, as defined in Section 2.1. Let $S \subseteq E(S^n)$ with $|S| = t$. For each edge $(i, i+1) \in S$, request i is routable alone. All other requests can route together without violating this edge's capacity, because any other request j which routes on $(i, i+1)$ has demand 2^j, edge $(i, i+1)$ has capacity 2^i, and $\sum_{j=0}^{i-1} 2^j < 2^i$. This defines a partition of $R(S^n)$ into $t+1$ sets: a set for each of the requests with the same indices as the t edges of S and a set of all other requests. Since all of these sets are routable, the indicator vector for each of these sets lies in $K_I(S)$. Since these sets partition $R(S^n)$, the vector $\frac{1}{t+1}$ is a convex combination of these sets, and hence $\frac{1}{t+1} \in K_I(S)$. Since this holds for every such S, we have $\frac{1}{t+1} \in P^t$ and its total profit is $\Omega(n/t)$, thus establishing the integrality gap. □

4.2.2 Tree instances

In this section, we prove the first part of Theorem 12 which gives a lower bound on the integrality gap of P^t on instances $T := T^h_{FG}$. Recall that Lemma 5 establishes $1/2 \in P^1$ by proving that for each edge e, the $1/2$ vector can be written as a convex combination of (incidence vectors of) two sets, each of which is routable on e. We generalize this to any value of t by showing that for $1/c \in P^t$ for sufficiently small c, and thus the integrality gap is $\Omega(\sqrt{\log n}/c)$.

Let $S \subseteq E(T)$. We call a set $X \subseteq R(T)$ *S-routable* if $\forall e \in S$, $\sum_{i \in X \cap R(e)} d_i \leq \mu_e$. Our key structural result gives a condition when we can express a vector $1/c$ as a convex combination of S-routable sets.

We cast this convex combination question as a question of colouring the set of all requests. For $S \subseteq E(T)$, we define the *S-chromatic number*, denoted by $\chi(S)$, to be the minimum value c such that $R(T)$ can be partitioned into c sets, each of which is S-routable. Given such a partition, the vector $1/c$ is trivially a convex combination of the indicator vectors of the S-routable sets in the partition. Thus, if we can show that $\chi(S) \leq c$ for every $|S| = t$, we have guaranteed that $1/c \in P^t$. Hence, the integrality gap established by Friggstad and Gao decreases by at most a factor of $c/2$ for P^t, since the result of Friggstad and Gao is associated with the feasible vector $1/2 \in P^0$. In fact, the following holds even if we start with the stronger formulation P_{rank}; we explain why at the end of this section.

Observation 2. *The integrality gap of P^t is $\Omega(h/c)$, where $c(t) := \max\{\chi(S) : S \subseteq R, |S| = t\}$.*

The lower bound half of Theorem 12 follows from the next results.

Proposition 2. *If $|S| \leq 2^{h(c-1)}$, then $\chi(S) \leq c + 1$.*

Corollary 3. *For constant t and $S \subseteq R$ with $|S| = t$, $\chi(S) \leq 2$ for sufficiently large h. Thus, the integrality gap of P^t is $\Omega(h)$.*

Our proof is based on the following colouring result. The tree T' plays the role of a subtree essentially induced by the edges from some set S with $|S| = t$.

Lemma 14. *Let T' be a subtree of T rooted at some vertex v. If each level of T' has at most $2^{h(c-1)}$ vertices, then $V(T')$ can be partitioned into at most c sets which are $E(T')$-routable.*

Proof. We prove this by induction using a stronger induction hypothesis. Specifically, not only does the colouring exist but we may use the following special type of colouring. We define layers L_k of T' inductively where $L_1 = \{v\}$. For each $k \geq 1$, L_{k+1} consists of the children of the requests in layer L_k which are contained in T'. Then for each $i = 1, 2, \ldots, c$ we claim that $X_i = L_i \cup L_{i+c} \cup L_{i+2c} \cup \ldots$ is $E(T')$-routable. Hence, $X_1, X_2, \ldots, X_c$ is a valid c-colouring which we call *layered*. We claim that a layered colouring always exists for any such subtree T'. The base case is a single-vertex tree which is trivially true for any $c \geq 1$.

Now consider the children of v in T'. Call these $v_1, v_2, \ldots v_p$ and let T_i be the subtrees of T' associated with each v_i. By induction, each T_i has a layered colouring which uses at most c colours. Assume we have such a colouring and without loss of generality that each v_i has colour class 2, the next layer below that has color class 3, and so on up to color class c, after which the next layer has color class 1. We show that v can be added to color class 1. Let X_i denote the union of the colour classes i which occur for the T_j. Each

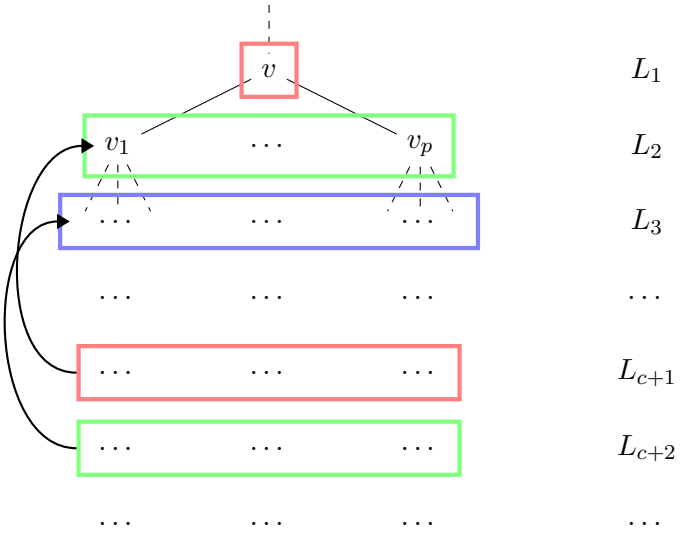

Figure 4.1: A diagram to aid with understanding the proof of Lemma 14. The key observation is illustrated by the arrows on the left: the total demand of every request in the box at the tail of an arrow is at most the demand of a single request at the tip of that arrow.

layered colour class X_i is $E(T_i)$-routable and thus is also $E(T')$-routable. Hence, it only remains to show that $X_1 \cup \{v\}$ is also $E(T')$-routable. Note that $X_1 \cup \{v\}$ consists of layers $L_1 \cup L_{c+1} \cup L_{2c+1} \cup \ldots \cup L_{qc+1}$ of T' for some choice of q. Recall that Lemma 6 asserts that the requests along any path from v to the leaves of T is routable. We show that for all i, the total demand of requests of L_{ic+1} is at most the demand of a single request in $L_{(i-1)c+2}$, so the total demand from requests in X_1 on any edge is at most the demand from routing a path from v to a leaf, and thus is routable. See Fig. 4.1 for a visual depiction of this. Suppose this is not the case. By the self similarity of the tree, we can assume that the demand of a request in L_{ic+1} is

$$2^{h(h-(ic-1)+1)} - 2^{h(h-(ic+1))} = (2^h - 1)2^{h(h-ic-1)}$$

and the demand of a request in $L_{(i-1)c+2}$ is

$$2^{h(h-((i-1)c+2)+1)} - 2^{h(h-((i-1)c+2))} = (2^h - 1)2^{h(h-(i-1)c-2)}.$$

Then, we have

$$|L_{ic+1}| \cdot (2^h - 1)2^{h(h-ic-1)} > (2^h - 1)2^{h(h-(i-1)c-2)}$$
$$\Leftrightarrow$$
$$|L_{ic+1}| > \frac{2^{h(h-(i-1)c-2)}}{2^{h(h-ic-1)}} = 2^{h(c-1)},$$

which contradicts our hypothesis. □

We now complete the proof of Proposition 2.

Proof. Let v be the least common ancestor of the vertices which are incident to the edges in S. We now create a subtree T' which is a sort of closure of S. T' is obtained by adding edges to S of any path between v and some vertex incident to an edge $e \in S$. We also include the parent edge of v. We claim that T' satisfies the hypothesis of Lemma 14. To see this, consider some level of T' consisting of vertices $a_1, \ldots, a_p$. Let E_i denote the set of edges which are either incident to vertex a_i or lie in its subtree. Note that the E_i are disjoint. Since each a_i is either incident to an edge of S, or is the internal vertices of some path used to define the closure T', it follows that $E_i \cap S \neq \emptyset$ for each i, and hence $p \leq \sum_{i=1}^{p} |E_i \cap S| \leq |S| \leq 2^{h(c-1)}$.

We now colour all the requests of T. We first invoke Lemma 14 to colour $R(T')$ using c colours. We can partition $R(T) \setminus R(T')$ as $A \cup B$, where B denotes the requests "below" T' (their paths to the root of T intersect T') and A denotes the remaining "above" requests. The set B is S-routable by Lemma 4. Requests in the set A do not even route on any edge of S. Hence, $A \cup B$ can be the $(c+1)^{st}$ colour class. □

To establish that this lower bound holds even when all rank inequalities are added, we use Theorem 4, a result by Friggstad-Gao [FG15] which shows that $x/9$ satisfies all rank constraints if x satisfies all valid constraints of the form $x_i + x_j \leq 1$, which are trivially satisfied by the vector $1/c$.

4.3 Integrality gap upper bound

In this section, we prove the upper bound part of Theorem 12, namely that for instances T^h_{FG} and $c > 0$, the integrality gap of both P^{n^c} and $P^{n^c}_{\text{rank}}$ is $O(1/c)$.

Theorem 13. *Let ℓ be the largest integer such that $n(\ell) \leq t$. The integrality gap for optimizing over P^t (with profits defined in Section 2.2) for instances T^h_{FG} is $O(h/\ell)$.*

We saw in Section 2.2 that $\ell = \Theta\left(\log(n(\ell))/h\right)$ and $h = \Theta\left(\sqrt{\log n}\right)$. For $c > 0$ and $t = n^c$, the theorem statement chooses $\ell = \Theta\left(\log(n^c)/h\right)$, so the integrality gap is $O(h/\ell) = O(1/c)$ as desired.

We show a particular way to partition the requests of the tree into $O(h/\ell)$ sets, and then show that for each set the profit of any $x \in P^t$ which uses only the requests in that set is $O(1)$. Since the integral optimum for instances T^h_{FG} is $\Theta(1)$ (see Lemma 9), it follows that the integrality gap of P^t is $O(h/\ell)$ on these instances. The proof relies on the self similar structure of the Friggstad-Gao instances, namely that every vertex except for the leaves and the root has exactly 2^{h-1} children, and capacities and demands scale down by 2^h for each step away from the root.

For $v \neq r$ let T^ℓ_v be the subtree consisting of the first ℓ levels of the children of vertex v along with the edge immediately above v. The edge immediately above v has its upper endpoint outside of the subtree. We denote the edges of the subtree, vertices of the subtree, and requests with an endpoint inside the subtree by $E(T^\ell_v)$, $V(T^\ell_v)$, and $R(T^\ell_v)$, respectively. For Friggstad-Gao instances, $|E(T^\ell_v)| = |V(T^\ell_v)| = |R(T^\ell_v)|$; we denote this size simply by $|T^\ell_v|$. Notice that we have $|T^\ell_v| \leq n(\ell)$ by self similarity—and this holds with equality—unless v is less than ℓ levels from the leaves. Since we assumed $n(\ell) \leq t$, we have $|T^\ell_v| \leq t$. For vectors $x \in \mathbb{R}^n$, we denote by $x_{T^\ell_v}$ the restriction of x to those requests with an endpoint in T^ℓ_v.

We now define, for each $0 \leq i < \lceil h/\ell \rceil$, a set of subtrees $\mathcal{P}_i = \left\{T^\ell_v : v \in level_{i\ell+1}\right\}$. Let $x_{\mathcal{P}_i}$ denote the restriction of x to those requests with an endpoint in some $T^\ell_v \in \mathcal{P}_i$. Observe that the union $\mathcal{P} = \bigcup \mathcal{P}_i$ of these subtrees is a partition of $T^h_{FG} \setminus \{r\}$ into edge and vertex disjoint subtrees. See Fig. 4.2 for a visual depiction of this. The following lemma bounds the profit obtainable using requests with an endpoint in some $\mathcal{P}_i$.

Lemma 15. *For any feasible vector $x \in P^t$ we have $w^T_{\mathcal{P}_i} x_{\mathcal{P}_i} \leq 2$ for all $0 \leq i < \lceil h/\ell \rceil$.*

Proof. Let $T^\ell_v \in \mathcal{P}_i$. First we show that every feasible subset of $R(T^\ell_v)$ has profit at most $2^{-(h-1)i\ell+1}$. This follows by the self similarity of the instance; scaling all demands and capacities in T^ℓ_v by $2^{hi\ell}$ and all profits by $2^{(h-1)i\ell}$ produces a tree identical to $T^\ell_{v_1}$ (recall v_1 is the single child vertex of the root r). From Lemma 9 we know that every routable set has profit at most 1.5625 so if we only use requests in $T^\ell_{v_1}$ the profit certainly must be less than 2. By scaling as necessary, it then follows that any feasible subset of $R(T^\ell_v)$ has profit at most $2^{-(h-1)i\ell+1}$, as desired.

Now, we show that to determine feasibility of a subset of $R(T^\ell_v)$ it is sufficient to check only the capacity constraints of the edges $E(T^\ell_v)$. If S is routable, then clearly no capacity constraints are violated, so assume conversely that S is not routable. By Lemma 4, no edge which is outside of $E(T^\ell_v)$ and is an ancestor (towards the root) of any edge in S has its

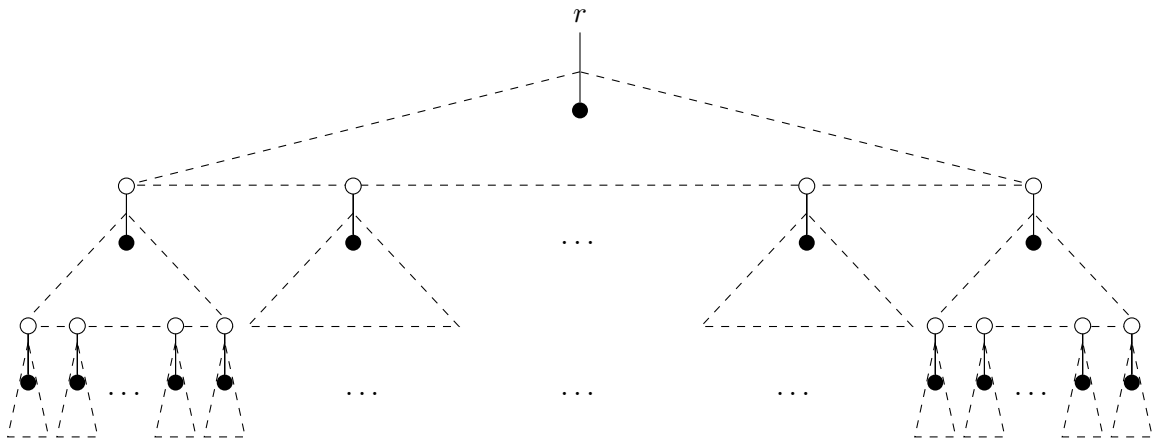

Figure 4.2: The partition of T_{FG}^h used to upper bound the integrality gap of the knapsack intersection hierarchy. Each vertex marked by $\bullet$ is associated with a subtree T_v^ℓ, indicated here by a dashed triangle. Each triangle spans ℓ layers of the tree, i.e., if a vertex marked by $\bullet$ is in level k, then the vertex marked by $\circ$ immediately below it are in level $k + \ell - 1$. The set $\mathcal{P}_i$ contains the i^{th} level of subtrees. For example, $\mathcal{P}_0$ contains the single triangle under r and $\mathcal{P}_1$ contains all the triangles immediately below that.

capacity violated by routing all requests in T. Furthermore, any other edge which is outside of $E(T_v^\ell)$ is not routed on by the requests in S and thus cannot be violated. Thus, in order for S to not be routable, the capacity of one of the edges in $E(T_v^\ell)$ must be violated.

Since $K_I(E(T_v^\ell))$ is an integer hull, any $x \in K_I(E(T_v^\ell))$ can be written as a convex combination of integral vectors in $K_I(E(T_v^\ell))$. We saw that to determine feasibility of a subset of $R(T_v^\ell)$ it is sufficient to check the capacity constraints of edges in $E(T_v^\ell)$. Thus, for $x \in K_I(E(T_v^\ell))$ such that $x \leq 1_{R(T_v^\ell)}$, we can write x as a convex combination of integral vectors 1_S for routable sets $S \subseteq R(T_v^\ell)$, which we know all have profit at most $2^{-(h-1)i\ell+1}$. Given $|T_v^\ell| \leq t$, any $x \in P^t$ has $x \in K_I(E(T_v^\ell))$, so $w_{T_v^\ell}^T x_{T_v^\ell} \leq 2^{-(h-1)i\ell+1}$. Finally, $|\mathcal{P}_i| = |level_{i\ell+1}| = 2^{(h-1)i\ell}$, so we can conclude that $w_{\mathcal{P}_i}^T x_{\mathcal{P}_i} \leq 2^{-(h-1)i\ell+1} \cdot 2^{(h-1)i\ell} = 2$. □

Proof of Theorem 13. Let $x \in P^t$. From Lemma 15, we know that for each $\leq i \leq \lfloor h/\ell \rfloor$ we have $w_{\mathcal{P}_i}^T x_{\mathcal{P}_i} \leq 2$. Summing over all i, we find that $w^T x \leq 2\lfloor h/\ell \rfloor \leq 2h/\ell$. We know that the integral optimum is $\Omega(1)$, so the integrality gap of P^t is $O(h/\ell)$. Since the rank formulation is stronger than the natural LP formulation, the integrality gap of P_{rank}^t is $O(h/\ell)$ as well. □

4.4 Comparison with other integer programming hierarchies

The use of hierarchies for integer programs dates back to the notion of Chvátal rank [Chv73]. The *Chvátal closure* of a polyhedron P is the polyhedron $P' \subseteq P$ which is defined by the system of Chvátal-Gomory cutting planes obtainable from P. If we denote by $P_C^1 = P'$, then the hierarchy is generated by $P_C^{t+1} = (P_C^t)'$. Chvátal proved that $P \supseteq P_C^1 \supseteq \ldots \supseteq P_C^{n-1} \supseteq P_C^n = P_I$. Other hierarchies have also been introduced and widely studied, such as those introduced by Lovász-Schrijver [LS91], Sherali-Adams [SA90], Parillo [Par03], and Lasserre [Las01].

There are few studies on the effectiveness of classical integer programming hierarchies on UFPT. Friggstand and Gao showed that the Lovász-Schrijver hierarchy is ineffective at reducing the integrality gap of UFPT after 2 rounds, and amounts to adding the clique constraints [FG15]. Additionally, Chekuri, Ene, and Korula prove that after applying t rounds of the Sherali-Adams hierarchy to the natural LP relaxation, the integrality gap is $\Omega(n/t)$ [CEK09], matching the result for our hierarchy. For the case of 0-1 knapsack, Karlin, Mathieu, and Nguyen show that t^2 rounds of Lasserre reduce the integrality gap to $t/(t-1)$ [KMN10], but there does not appear to be any work done on whether this would generalize to UFPT or m-KP.

Chapter 5

Conclusion

Our investigation of extended formulations for MKP and UFPT has led to new results and revealed some interesting new directions for future research.

In Chapter 2 we re-examined the hard instances for UFPT introduced by Friggstad and Gao and presented a tight analysis of the integrality gap for those instances. We could not find any way to modify the Friggstad-Gao instances to make the integrality gap larger.

In Chapter 3 we gave a new $(1 + 0.79m)$-approximate extended formulation for m-KP which uses only a linear number of extra variables and constraints. At a matching approximation ratio, this formulation is significantly smaller than Pritchard's $(1 - \epsilon)$-approximate extended formulation, which we presented in Section 3.3. We speculate that it may be possible to use ideas from our result to derive an improved $(1-\epsilon)$-approximate formulation. Future work in this area would also include modifying our result to not depend on the objective function.

In Chapter 4 we introduced a new hierarchy of strengthened formulations for multidimensional knapsack and related problems which measures the ultimate power of adding "t-row cuts". While separating over the t^{th} level P^t is NP-hard, there is a $(1-\epsilon)$-approximate version based on results of Pritchard. We examined the efficacy of this hierarchy for the well-studied unsplittable flow problem on trees. An important problem we did not resolve is to analyze the integrality gap of our hierarchy for general instances when applied to the rank LP. We have yet to establish any strong link between this new hierarchy and others, such as those introduced by Lasserre, Parillo, Lovász-Schrijver, or Sherali-Adams. Our hierarchy may also be useful in strengthening formulations for other special cases of multidimensional knapsack. We speculate that it will be most beneficial in the case when each item has a relatively large size in only a few dimensions, because then a formulation close to the integer hull could be captured by adding t-row cuts for small t.

Bibliography

[ACEW16] Anna Adamaszek, Parinya Chalermsook, Alina Ene, and Andreas Wiese. Submodular unsplittable flow on trees. *Mathematical Programming*, 172, 2016. → page 14

[ACH09] Chrisil Arackaparambil, Amit Chakrabarti, and Chien-Chung Huang. Approximability of the unsplittable flow problem on trees. *Dartmouth Computer Science Technical Report*, (642), 2009. → page 14

[AGLW13] Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. Constant integrality gap LP formulations of unsplittable flow on a path. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 25–36. Springer, 2013. → page 15

[AGLW18] Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing 2+ϵ approximation for unsplittable flow on a path. *ACM Transactions on Algorithms*, 14, 2018. → pages 14, 15

[Bal79] Egon Balas. Disjunctive programming. *Annals of discrete mathematics*, 5:3–51, 1979. → page 27

[BCES06] Nikhil Bansal, Amit Chakrabarti, Amir Epstein, and Baruch Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, page 721–729, New York, NY, USA, 2006. Association for Computing Machinery. → page 15

[BFKS09] Nikhil Bansal, Zachary Friggstad, Rohit Khandekar, and Mohammad R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. *ACM Transactions on Algorithms*, 10, 2009. → page 15

[BGK+14] Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 47–58. SIAM, 2014. → page 15

[Bie08] Daniel Bienstock. Approximate formulations for 0-1 knapsack sets. *Operations Research Letters*, 36, 2008. → page 31

[BM12] Daniel Bienstock and Benjamin McClosky. Tightening simple mixed-integer sets with guaranteed bounds. *Mathematical Programming*, 133, 2012. → pages v, 12, 28

[BSW14] Paul Bonsma, Jens Schulz, and Andreas Wiese. A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM journal on computing*, 43(2):767–799, 2014. → pages 13, 15

[CCGK02] Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. Approximation algorithms for the unsplittable flow problem. *Lecture Notes in Computer Science*, 47, 2002. → page 16

[CEK09] Chandra Chekuri, Alina Ene, and Nitish Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 42–55. Springer, 2009. → pages 7, 11, 12, 14, 17, 24, 38, 39, 45

[Chv73] Vasek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete mathematics*, 4(4):305–337, 1973. → page 45

[CJP83] Harlan Crowder, Ellis L Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983. → page 36

[CKS13] Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. The all-or-nothing multicommodity flow problem. *SIAM Journal on Computing*, 42(4):1467–1493, 2013. → page 13

[CMS07] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3, 2007. → pages 1, 14, 15

[CS94] Steve Cosares and Iraj Saniee. An optimization problem related to balancing loads on SONET rings. *Telecommunication Systems*, 3(2):165–181, 1994. → page 13

[DLTW14] Santanu S Dey, Andrea Lodi, Andrea Tramontani, and Laurence A Wolsey. On the practical strength of two-row tableau cuts. *INFORMS Journal on Computing*, 26(2):222–237, 2014. → page 36

[DM18] Santanu S Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170(1):237–266, 2018. → page 36

[Eis99] Friedrich Eisenbrand. Note on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999. → page 37

[FG15] Zachary Friggstad and Zhihan Gao. On Linear Programming Relaxations for Unsplittable Flow in Trees. In Naveen Garg, Klaus Jansen, Anup Rao, and

José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 265–283, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. → pages 12, 14, 18, 20, 24, 37, 42, 45

[FS15] Yuri Faenza and Laura Sanità. On the existence of compact ϵ-approximated formulations for knapsack in the original space. *Operations Research Letters*, 43, 2015. → page 25

[GMW20] Fabrizio Grandoni, Tobias Mömke, and Andreas Wiese. Unsplittable flow on a path: The game! 2020. → page 14

[GMW21] Fabrizio Grandoni, Tobias Mömke, and Andreas Wiese. Faster $(1 + \epsilon)$-approximation for unsplittable flow on a path via resource augmentation and back. 2021. → page 15

[GMWZ17] Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. To augment or not to augment: solving unsplittable flow on a path by creating slack. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2411–2422. SIAM, 2017. → page 15

[GMWZ18] Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. A $(5/3 + \epsilon)$-approximation for unsplittable flow on a path: placing small tasks into boxes. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 607–619, 2018. → pages 1, 14, 15

[GVY97] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. → page 14

[Kle96] Jon M Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Massachusetts Institute of Technology, 1996. → page 13

[KMN10] Anna R. Karlin, Claire Mathieu, and C. Thach Nguyen. Integrality gaps of linear and semi-definite programming relaxations for knapsack. *arXiv: Computational Complexity*, 2010. → page 45

[KPP04] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Multidimensional knapsack problems. In *Knapsack problems*, pages 235–283. Springer, 2004. → page 36

[Las01] Jean B Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 293–303. Springer, 2001. → pages 37, 45

[LS91] László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM journal on optimization*, 1(2):166–190, 1991. → pages 37, 45

[LW08] Quentin Louveaux and Robert Weismantel. Polyhedral properties for the intersection of two knapsacks. *Mathematical programming*, 113(1):15–37, 2008. → page 36

[Par03] Pablo A Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003. → pages 37, 45

[PKP13] David Pisinger, H Kellerer, and U Pferschy. Knapsack problems. *Handbook of Combinatorial Optimization*, page 299, 2013. → pages 2, 3, 5, 6

[Pri10] David Pritchard. An LP with integrality gap $1 + \epsilon$ for multidimensional knapsack. *arXiv: Discrete Mathematics*, 2010. → pages v, 12, 31, 34, 38

[SA90] Hanif D Sherali and Warren P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990. → pages 37, 45

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998. → page 36

[VW06] Mathieu Van Vyve and Laurence A. Wolsey. Approximate extended formulations. *Mathematical Programming*, 105, 2006. → pages 31, 34

[Wie17] Andreas Wiese. A $(1 + \epsilon)$-approximation for unsplittable flow on a path in fixed-parameter running time. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. → page 15

[Xav17] Alinson Santos Xavier. *Computing with multi-row intersection cuts*. PhD thesis, The University of Waterloo, 2017. → page 36