Efficient Algorithms Inspired by Integer Programming Formulations

by

Xu Rao

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Industrial Engineering and Operations Research

in the

Graduate Division

of the

University of California, Berkeley

Fall 2020

Efficient Algorithms Inspired by Integer Programming Formulations

Abstract

Efficient Algorithms Inspired by Integer Programming Formulations

by

Xu Rao

Doctor of Philosophy in Engineering - Industrial Engineering and Operations Research

University of California, Berkeley

Professor Dorit S. Hochbaum, Chair

Integer programming formulations play a key role in the design of efficient algorithms and approximation algorithms for many discrete linear optimization problems as found in previous research. A specific integer programming formulation of a discrete linear optimization problem may lead to algorithms with better running time or approximation algorithms with better approximation factors than the other algorithms for the same problem. Inspired by this, we introduce here new integer programming formulations to devise efficient algorithms and approximation algorithms for two classes of problems, the covariate balancing problems and the Replenishment Storage problems.

The existence of certain special integer programming formulations is an evidence of the existence of polynomial time algorithms for some discrete optimization problems. For a variant of the covariate balancing problems, our new integer programming formulation has a network structure that was not previously known. We use this new formulation to devise network flow algorithms, which have better running time than an existing polynomial time algorithm. Other integer programming formulations we introduce show that several variants of the class of covariate balancing problems are fixed-parameter tractable.

Integer programming formulations are often important in designing approximation algorithms for intractable problems. The Replenishment Storage problems were known to be NP-hard and one approximation algorithm was known for a special variant of the class. We derive for this variant a polynomial time approximation scheme using a new integer programming formulation. Our new formulation also leads to the first known approximation scheme for another variant of this class. Moreover, this resolves the complexity status of some variants of the Replenishment Storage problems as weakly NP-hard.

Contents

List of Figures

List of Tables

Acknowledgments

This work and the completion of my PhD would not have been possible without the support of numerous people. In particular, I would like to thank:

My parents, Shaomei and Bei, for guiding me over the course of my life and for their unconditional support for me; my grandparents Meiting, Dingguo, Xiancheng, Lianchan for their love and encouragement; my uncle Bo and aunt Weiwu for sending me essential materials so I can stay healthy in the pandemic while working on this dissertation.

My advisor Dorit S. Hochbaum for her guidance and advice during my graduate studies, her prompt and patient reply to all my questions, and her valuable support in my job search. I appreciate the countless hours we spent on discussing interesting problems, brainstorming research ideas, writing and revising papers for submission.

My co-author Asaf Levin for our fruitful collaborations; Professors Phil Kaminsky, Max Shen, Prasad Raghavendra for their willingness to serve on my committee.

My roommates Mo Zhou, Renyuan Xu, Meng Qi, Sha Lu, Mengqiao Yu and Xiaoke Hu for doing workout sessions, cooking and having lots of fun time together; my friend Chen Shao, for our memorable trips in many beautiful places; my fellow PhD students for our friendships, lunch time, good advice, holiday parties, and many fun conversations. I feel very fortunate meeting these amazing people and having them with me at Berkeley. My thanks go to: Erik Bertelli, Haoyang Cao, Junyu Cao, Ying Cao, Carlos Deck, Shiman Ding, Yuhao Ding, Pelagie Elimbi Moudio, Salar Fattahi, Han Feng, Andrés Gómez, Dean Grosbard, Pedro Hespanhol, Anran Hu, Arman Jabbari, Titouan Jehl, Hansheng Jiang, Heejung Kim, Kevin Li, Tianyi Lin, Heyuan Liu, Sheng Liu, Alfonso Lobos, Cheng Lu, Yonatan Mintz, Matt Olfat, Georgios Patsakis, Meng Qi, Wei Qi, Amber Richter, Rebecca Sarto Basso, Jiaying Shi, Quico Spaen, Birce Tezel, Mark Velednitsky, Renyuan Xu, Mengxin Wang, Nan Yang, Mo Zhou, Ruijie Zhou, and many others. Finally, I would like to thank all my other Berkeley friends and college friends.

Chapter 1

Introduction

Integer programming formulations play a key role in the design of efficient algorithms and approximation algorithms for many discrete linear optimization problems as found in previous research. Although integer programming problems are in general intractable (NP-hard), there are special classes of integer programs that are polynomial time solvable. In some cases, a discrete linear optimization problem is shown to be polynomial time solvable through the integer programming formulation of specific forms. For NP-hard discrete linear optimization problems, a considerable amount of approximation algorithms depend on integer programming formulations. Sometimes, different formulations of a problem make a difference in terms of approximation factor and running time. Several examples from previous studies are provided together with more background knowledge in Section 1.1 and Section 1.2.

In this dissertation, we use new integer programming formulations to devise efficient algorithms for the class of covariate balancing problems and to derive approximation algorithms for the class of Replenishment Storage problems. These two problem classes are introduced in Section 1.3.

For one variant of the covariate balancing problems, we found an integer programming formulation with a previously unknown network structure in the constraints, which is used to devise efficient network flow algorithms. These algorithms are more efficient than an existing algorithm in [3], which solves the linear programming relaxation of a different integer program. Other integer programming formulations we introduce here for the class of covariate balancing problems show that several variants of the problems are fixed-parameter tractable (FPT).

For the class of Replenishment Storage problems, we use a new integer programming formulation to derive approximation algorithms. Our algorithms are the first known approximation algorithms for one variant of the class. For another variant, we provide better approximation factors than a previously known algorithm. Our results also resolve the complexity status of some variants of the Replenishment Storage problems as weakly NP-hard.

1.1 Integer programming formulations that imply polynomial time algorithms

Some discrete linear optimization problems are shown to be polynomial time solvable by specific formulations, which belong to special classes of integer programs that have known efficient algorithms.

One class of integer programs with known polynomial time algorithms are those in which the constraint matrix is *totally unimodular* and the right-hand side of the constraints are integers. A matrix is said to be totally unimodular if all the sub-determinants are 1, 0 or 1. For any linear programming problem with totally unimodular constraint matrix and integral right-hand side, every basic feasible solution is integral. And since every linear programming problem is polynomial time solvable, if an optimal solution exists, there is an integral optimal solution and it can be attained in polynomial time. Therefore, the class of integer programs with totally unimodular constraint matrices and integral right-hand sides can be solved efficiently with their linear programming relaxations.

Among the class of totally unimodular matrices, there is a sub-class of matrices in which each column (or row) contains at most one 1 and at most one -1, with all the remaining entries being 0. This sub-class of matrices are called the *network matrices*, and an integer programming problem with a network constraint matrix is a *network flow problem*. A vast number of algorithms have been developed for the network flow problems, and these algorithms are more efficient than solving the linear programming relaxations.

The structure of the network matrices is used here for devising efficient algorithms for the class of covariate balancing problems. One variant of the covariate balancing problems was known to be polynomial time solvable by solving the linear programming relaxation of an integer programming formulation. We identify here an integer program with the network structure. As a result, two network flow algorithms are devised. This improves the running time complexity of the previously known method for this variant of covariate balancing problems.

Another class of integer programs that is polynomial time solvable is called *monotone IP3* [26]. A monotone IP3 problem is characterized by constraints that have at most two variables per inequality that appear with opposite sign coefficients, and in addition a third variable that appears only in one constraint (the coefficients of those third variables objective are required to be nonnegative in a minimization objective or nonpositive in a maximization objective). This class of integer programs is solvable with combinatorial flow algorithms in polynomial time for polynomially bounded inputs [26]. One example that was discovered as monotone IP3 is the Hochbaum's Normalized Cut (HNC) problem, a variant of the Normalized Cut problem for image segmentation. The HNC problem was mistakenly thought to be NP-hard in [51]. Hochbaum formulated the HNC as a monotone IP3 problem and thus, established that it is polynomial time solvable [26].

1.2 Integer programming formulations that imply approximation algorithms

One of the ways to cope with intractable problems is to derive *approximation algorithms*, which yield feasible solutions that approximate the optimal solution in terms of objective value. A δ-approximation algorithm for a minimization (maximization) problem is a polynomial time algorithm that for all instances of the problem produces a feasible solution whose objective value is within (at least) a factor of δ of the objective value of an optimal solution. For a δ-approximation algorithm, we will call δ the *approximation factor* of the algorithm.

Integer programming formulations have been used extensively in designing and analyzing the approximation algorithms of NP-hard discrete linear optimization problems. This topic is discussed in the books of Hochbaum [23], Williamson and Shmoys [58], and Vazirani [57]. Sometimes different choices of formulations would give different approximation factors or running time.

One example is the minimum knapsack problem, for which the standard integer programming formulation would lead to arbitrarily bad approximation factors. To remedy this, Carr et. al [8] provided a strengthened integer programming formulation that has an exponential number of constraints. This strengthened formulation was used to develop a deterministic rounding approximation algorithm [8] and a primal-dual approximation algorithm [7], both of which attain an approximation factor of 2. The deterministic rounding algorithm rounds the fractional solution of the linear programming relaxation to an integer solution based on some rules. The primal-dual method, which often gives much faster algorithms than the deterministic rounding algorithms, constructs an integral solution to the primal program and a feasible solution to the dual program iteratively. Both the deterministic rounding and the primal-dual methods compare the value of the solution with the value of the linear programming relaxation solution. So it is impossible to achieve an approximation factor that is better than the *integrality gap*, where the *integrality gap* of an integer programming formulation is defined to be the worst-case ratio over all instances of the optimal value to the linear programming relaxation to the optimal value of the integer program. The standard integer programming formulation of the minimum knapsack problem has an integrality gap that can be arbitrarily large. So it is impossible to use the standard formulation to derive good approximation algorithms using these two methods. On the other hand, the strengthened formulation of Carr et. al. has an integrality gap of 2, which allows 2-approximation algorithms to be built upon this formulation.

Many approximation algorithms that relied on the linear programming relaxation solution are restricted from getting a better approximation factor than the integrality gap. We show here that the integer programming formulation can also lead to approximation algorithms that beat the integrality gap for the class of Replenishment Storage problems. The Replenishment Storage problems are NP-hard, and no approximation algorithm was known except for a special variant [18]. The approximation factor of this known algorithm varies from 1 to 2 depending on the problem instance. We provide a new integer programming formulation here, and design *approximation schemes* (see Chapter 2 for definitions) for various variants based on our formulation. An approximation factor of $(1 + \epsilon)$ for any $\epsilon > 0$ can be attained by the approximation schemes, with the running time depending on ϵ.

Different integer programming formulations can lead to approximation algorithms with the same approximation factor but different running time. An example can be found in [17, 9] for the problem of minimizing the sum of weighted completion times of precedence-constrained jobs on a single machine. Hall et. al. [17] provided the first 2-approximation algorithm based on solving the linear relaxation of an integer programming formulation using completion time variables. This integer program of Hall et. al. contains an exponential number of inequalities, yet the linear programming relaxation can be solved in polynomial time using a separation oracle. Chudak and Hochbaum [9] later presented a different integer programming formulation using the linear ordering variables, which contains two variables per inequality in the constraints. This formulation of Chudak and Hochbaum is a special case of a class of integer programs called *IP2* [28], and as a consequence, there is a 2-approximation algorithm that solves it in polynomial time with network flow techniques (see next paragraph for details). This is more efficient than solving the linear relaxation of the formulation shown by Hall et. al.

IP2 is a class of integer programs with bounded variables over constraints with at most two variables per inequality [28]. Hochbaum et. al. [28] showed that any feasible IP2 problem has a 2-approximation solution derived in the time required to solve a minimum cut problem, which is polynomial if the range of variables are polynomially bounded. Therefore, whenever an IP2 formulation applies to a feasible discrete optimization problem, there is a 2-approximation algorithm that solves the problem with a minimum cut procedure. The 2-approximation algorithms with IP2 formulations were generated for many problems including the following ones: the MIN 2-SAT problem [28], the biclique problems in [24], the minimum satisfiability problem [25], and the complement of maximum clique problem [25].

1.3 Applications

Covariate Balancing

In an observational study, one is given disjoint samples of treatment units and control units, and the goal is to compare outcomes between the two types of samples in order to estimate a treatment effect. A complication is that the treatment assignment mechanism is in general non-random, and so the treatment and control units often differ on important pre-treatment attributes called *covariates*. These differences, referred to as covariate imbalance, can confound the estimate of the treatment effect if not properly taken into account. Therefore, it is a common theme across various observational studies on estimating treatment effects to adjust for covariate imbalance. The covariate balancing problems considered here are optimization problems that find a selection of treatment samples and a selection (subset) of control samples, which minimize the covariate imbalance over several different measurements of imbalance. Only the samples in the selections are used for the estimation.

The problem of identifying causal connections between actions (treatments) and outcomes arises in several fields, such as the social sciences [11, 15, 21], epidemiology [5], medicine [46, 59, 63], economics [31], and political science [22]. There are two different set-ups for gathering the data used for assessing the impact of a treatment, experimental and observational. In experimental studies for assessing the impact of a treatment, researchers have access to samples drawn from some population and can choose which of them receive treatment and which do not. The standard way of choosing the samples is through random assignment. This ensures an unbiased estimate of the treatment effect, which means the expected value of the estimated treatment effect is the true treatment effect. Unlike experimental studies, in an observational study, the researcher does not have the ability to determine treatment assignment and instead is only able to observe some units that were treated and some that were not. For example, when studying the effect of smoking, it is unethical to randomly select some individuals to be exposed to smoking while others are not [41, 50]. The observed treated and untreated units are called the *treatment sample* and *control sample*, respectively.

The major obstacle in observational studies is the *selection bias*, which is the distortion in the estimated treatment effect caused by differences in the covariates (pre-treatment attributes) of the treated and control populations. Examples of covariates in medical data include age, height, weight, blood pressure, disease history, and/or genetic information. Despite this difficulty, the ease of access to ever-increasing quantities of observational data makes such studies popular across a wide range of disciplines.

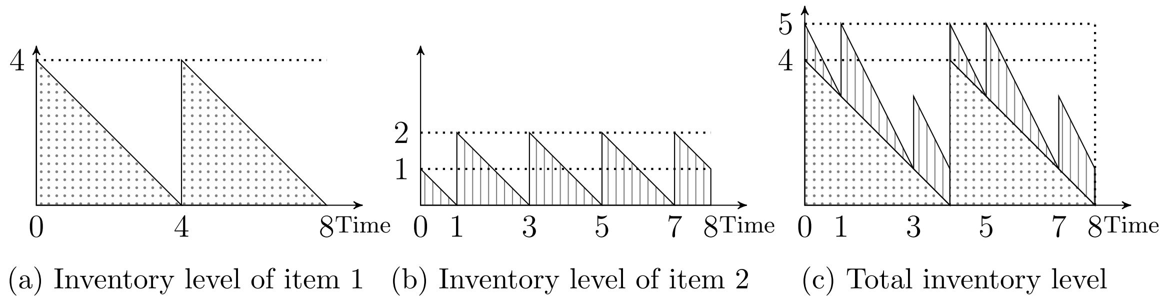
It is a common theme across various observational studies on estimating treatment effects to adjust for covariate imbalance, which refers to the differences in covariates between the treatment and control populations. We address here several variants of the covariate balancing problems with different measurements of imbalance.

We show that certain variants of the covariate balancing problems can be solved efficiently as discovered through our integer programming formulations with network structure. One of these variants was known to be polynomial time solvable with the linear programming relaxation [3]. We are able to solve this variant here with network flow algorithms, which have better running time. When the number of covariates and the number of levels of each covariate are parameters, we derive several fixed-parameter tractable (FPT) results. These FPT results are based on the integer programming formulations that require only a fixed number of variables. With the polynomial time algorithm of Lenstra [37] for integer programs on a fixed number of variables, we determine that those variants of covariate balancing problems with such formulations are FPT.

The Replenishment Schedule to Minimize Peak Storage

The Replenishment Storage problem (RSP) arises in planning a periodic replenishment schedule of multiple items so as to minimize the storage capacity required. The input to the RSP consists of a multi-item inventory system where each item has deterministic demand, a given reorder size and its own cycle length determined by its Economic Order Quantity.

Here the reorders can only take place at an integer time unit within the cycle. The problem is to determine the timing of the first replenishment of each item within its cycle so that the maximum inventory level of all items over time is minimized. Geometrically, RSP can be viewed as a problem of shifting periodic triangle functions and then packing them on top of each other so as to minimize the peak value required as shown in Figure 1.1. (This figure is used again in Chapter 4 for a more detailed introduction to the RSP.)



(a) Inventory level of item 1 (b) Inventory level of item 2 (c) Total inventory level

Figure 1.1: A geometric view of RSP

An instance of RSP consists of n items. Each item is associated with an integer individual cycle length. The *joint cycle length* of the n items, denoted by k here, is the least common multiple (lcm) of the individual cycle lengths. By the cyclical nature of the problem, the total inventory levels repeat periodically every k units of time for any reorder schedule. If all items have the same cycle time, k, the problem is said to be *single-cycle*, otherwise, it is said to be *multi-cycle*.

The RSP is NP-hard [18]. Studies of the RSP have been mainly focused on the development of heuristics [38, 60, 61, 4, 10, 49]. The only known approximation algorithm is a $(1 + 2/k)$-approximation algorithm for the single-cycle of RSP [18].

We introduce for the RSP a new integer programming formulation. Using this new formulation, we provide a pseudo-polynomial time algorithm that solves the RSP for constant joint cycle length. It follows that the RSP with constant joint cycle length is weakly NP-hard. On the other hand, the RSP with non-constant joint cycle length is proved to be strongly NP-hard here, and therefore there is no pseudo-polynomial algorithm unless P=NP. That means, the complexity of the RSP is different between the variants with constant joint cycle length and the variants with non-constant joint cycle length.

Then we use the integer programming formulation again to derive approximation schemes, which are families of $(1 + \epsilon)$-approximation algorithms for every $\epsilon > 0$. There are two types of approximation schemes, Polynomial Time Approximation Scheme (PTAS) and Fully Polynomial Time Approximation Scheme (FPTAS), which are defined in Chapter 2. We derive the first known approximation schemes for three variants of the RSP: a FPTAS for the multi-cycle RSP with constant joint cycle length, a FPTAS which is FPT in terms of the joint cycle length for the single-cycle RSP with constant joint cycle length, and a PTAS for the single-cycle RSP with non-constant joint cycle length. The question of whether there exists a PTAS for the multi-cycle RSP with non-constant joint cycle length remains open.

1.4 Overview

Preliminaries are provided in Chapter 2, which includes the description of two network flow models used here, definitions of different types of approximation algorithms and fixed-parameter tractability.

In Chapter 3 we consider several variants of covariate balancing problems. We discuss the computational complexity of these problems. And we show that with a different integer programming formulation from the one in an existing paper [3], we can develop more efficient algorithms to handle several special cases. We also use the integer programming formulations for these problems to derive the fixed-parameter tractability results.

Chapter 4 lays out our approach for designing approximation schemes for the Replenishment Storage problem. The chapter provides the problem model, our new integer programming formulation, a dynamic programming algorithm with pseudo-polynomial running time, the fully polynomial time approximation scheme for the problem with constant joint cycle length, as well as the polynomial time approximation scheme for the single-cycle problem with non-constant joint cycle length.

Chapter 2

Preliminaries

We first introduce two network flow models, the minimum cost network flow problem and the maximum flow problem, which are used in Chapter 3 for deriving efficient network flow algorithms for covariate balancing problems. Next, we provide the formal definition of two types of approximation schemes, and the strongly and weakly NP-hard terminologies, which are relevant to the existence of approximation schemes. Then we introduce the concept of fixed-parameter tractability (FPT). This concept is used in both Chapter 3 and Chapter 4.

We abbreviate in this dissertation linear programs or linear programming by the acronym LP, and we abbreviate integer program or integer programming by the acronym IP.

2.1 The minimum cost network flow problem and the maximum flow problem

The minimum cost network flow (MCNF) problem is to determine a least cost flow through a network in order to satisfy demands at certain nodes from available supplies at other nodes. The input to the problem is a graph $G = (V, A)$ with a set of nodes V and a set of arcs A, where each arc $(i, j) \in A$ is associated with a cost c_{ij} that denotes the cost per unit flow on that arc, capacity upper bound u_{ij}, and capacity lower bound l_{ij}. Each node $i \in V$ has supply b_i which is interpreted as demand if negative, and can be 0. Let x_{ij} be the amount of flow on arc $(i, j) \in A$. The sum of flows on arcs directed to node k is the inflow of k, and the sum of flows on arcs directed from node k is the outflow of k. The flow vector $\mathbf{x}$ is said to be feasible if it satisfies:
(1)**Flow balance constraints:** For every node $k \in V$ $Outflow(k) - Inflow(k) = b_k$
(2) **Capacity constraints:** For each arc $(i, j) \in A$, $l_{ij} \leq x_{ij} \leq u_{ij}$.

The linear programming formulation of the problem is:

$$\begin{array}{lll} \text{(MCNF)} & \min & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \text{subject to} & & \sum_{j:(k,j)\in A} x_{kj} - \sum_{i:(i,k)\in A} x_{ik} = b_i \ \forall k \in V \\ & & l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in A. \end{array}$$

The flow balance constraints coefficients form a $\{0, 1, -1\}$-matrix where in each column there is exactly one 1 and one -1. Such matrix is a special case of matrices where each column (or row) has at most one 1 and at most one -1, which are known to be totally unimodular. The addition of the capacity constraints retains the total unimodularity property. Note that the MCNF constraint matrix is linearly dependent (with the sum of all balance constraints leading to $0 = 0$) and one constraint can be eliminated as it is the negative sum of the others.

The flow balance constraints coefficients form a $\{0, 1, -1\}$-matrix where in each column there is exactly one 1 and one -1. Such matrix is known to be totally unimodular. So the linear programming relaxation of any MCNF problem has all basic solutions, and in particular the optimal solution, integral. However, there are many specialized algorithms for the MCNF problem which are more efficient than solving its linear programming relaxation.

The maximum flow problem, *max-flow*, is a special case of MCNF. The max-flow problem seeks a feasible solution that sends the maximum amount of flow from a specified source node to another specified sink node. The max-flow problem is defined on a directed graph $G = (V, A)$, with arc capacities u_{ij} and two distinguished nodes: a source node, $s \in V$, and a sink node, $t \in V$. There is no lower bound or cost associated to each arc. The total outflow from s, or the total inflow into t, is called the value of the flow. The objective is to find the maximum value feasible flow leaving s and reaching t that satisfy the arc capacities. A linear programming formulation of max-flow is:

$$\begin{array}{lll}
\text{(max-flow)} & \max & f \\
\text{subject to} & & \sum_{(s,j)\in A} x_{sj} = f \\
& & \sum_{(k,j)\in A} x_{kj} - \sum_{(i,k)\in A} x_{ik} = 0 \ \forall k \in V \setminus \{s,t\} \\
& & 0 \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in A.
\end{array}$$

As a special case of MCNF there are specialized algorithms for max-flow which are more efficient than algorithms for MCNF.

2.2 Approximation schemes and strong NP-hardness

First, we define approximation algorithms.

Definition 2.1. *A δ-approximation algorithm for a minimization (maximization) problem is a polynomial time algorithm that for all instances of the problem produces a feasible solution whose objective value is within (at least) a factor of δ of the optimal objective value.*

An approximation scheme is a family of approximation algorithms with some special property. There are two types of approximation schemes and their definitions are given below.

Definition 2.2. *A polynomial time approximation scheme (PTAS) is a family of algorithms, in which for every ϵ there is a $(1 + \epsilon)$-approximation algorithm (for minimization problem) or a $(1 - \epsilon)$-approximation algorithm (for maximization problem)*

The running time of an algorithm in a PTAS is allowed to depend arbitrarily on $1/\epsilon$. So the running time could be exponential in $1/\epsilon$. In the special case where the running time of a PTAS is polynomial in $1/\epsilon$, it is said to be fully polynomial.

Definition 2.3. *A fully polynomial time approximation scheme (FPTAS) is an approximation scheme such that the running time of the $(1 \pm \epsilon)$-approximation algorithm is polynomial in $1/\epsilon$ for every ϵ.*

Not all NP-hard optimization problems have a FPTAS. Under the assumption that P≠NP, class of problems that do not admit a FPTAS is the class of strongly NP-hard problems, which is defined next.

Definition 2.4. *A problem is strongly NP-hard if it is NP-hard even when its numeric data is encoded in unary.*

The following theorem proven in [57] shows that it is impossible to find a FPTAS for any strongly NP-hard problems unless P=NP.

Theorem 2.1. *[57] Any strongly NP-hard problem with a polynomially bounded objective function does not have a FPTAS, assuming P≠NP.*

In contrast to strongly NP-hard problems, there are weakly NP-hard problems which are defined below:

Definition 2.5. *A problem is weakly NP-hard if it has a pseudo-polynomial time algorithm (that is, it has polynomial time algorithm if its numeric data is encoded in unary).*

It is possible that a weakly NP-hard problem has a FPTAS, but not every weakly NP-hard problem has one. For example, the m-dimensional knapsack problem for any fixed $m \geq 2$ is weakly NP-hard, but has no FPTAS even when the optimal objective is polynomially bounded [36].

2.3 Parameterized complexity

Compared with classical complexity theory, parametrized complexity theory measures complexity not only in terms of the input size, but also in terms of a parameter, which is a numerical value that may depend on the input.

Definition 2.6. *An algorithm with input size n is fixed-parameter tractable (FPT) with respect to parameter k if the running time of the algorithm is at most $f(k) \cdot (n)^{O(1)}$, where $f(k)$ is a computable function of parameter k which is independent of n. A problem that has an FPT algorithm is said to be a fixed-parameter tractable problem.*

FPT problems that we use here are integer programs with fixed numbers of variables, as well as mixed-integer linear programs with fixed numbers of integer variables. These problems can be solved in FPT time with respect to the number of integral variables, by the algorithms developed by Lenstra [37] (see also [34] for improved time complexity of these algorithms). We use new integer programming formulations, mixed and not mixed, that include fixed numbers of integer variables to derive FPT results for several variants of the covariate balancing problems.

Chapter 3

Efficient Algorithms for Variants of Covariate Balancing Problems

In an observational study, one is given disjoint samples of treatment units and control units, and the goal is to compare outcomes between the two types of samples in order to estimate a treatment effect. A complication is that the treatment assignment mechanism is in general non-random, and so the treatment and control units often differ on important pre-treatment attributes called *covariates*. These differences, referred to as covariate imbalance, can confound the estimate of the treatment effect if not properly taken into account. Therefore, it is a common theme across various observational studies on estimating treatment effects to adjust for covariate imbalance. The covariate balancing problems considered here are optimization problems that find a selection of treatment samples and a selection (subset) of control samples, which minimize the covariate imbalance over several different measurements of imbalance. Only the samples in the selections are used for the estimation.

Although these covariate balancing optimization problems are all NP-Hard in general, certain special cases can be solved efficiently as discovered through our network flow IP formulations. In addition to the polynomial time algorithms, we present FPT results for several variants of the covariate balancing problems. Part of these results is included in two unpublished manuscripts [29, 27].

Chapter Overview

We consider five families of covariate balancing problems in this chapter. The definition of the problems, related literature, and a summary of our results are presented in Section 3.1. In Section 3.2 we consider the special case in which there is only 1 covariate for each of the problem families and provide a compact representation of the solutions. For the case of 2 or more covariates, the complexity and algorithmic results for four of the five problem families are presented in Section 3.3, Section 3.4, Section 3.5, and Section 3.6 respectively. The remaining one is implied to be NP-hard with 2 or more covariates from the known

NP-hardness of another problem, which is mentioned in Section 3.1. The fixed-parameter complexity results for the five problem families are all provided in Section 3.7.

3.1 The class of covariate balancing problems

To formalize the discussion we introduce notation. Let the number of treatment samples be n and the number of control samples be n' (assuming that $n \leq n'$). Let the set of all treatment samples be denoted by $\mathcal{T}$, $|\mathcal{T}| = n$. Each sample, in either group, has P observed nominal covariates. The values, or categories, of each covariate partition the treatment and control samples to a number of subsets referred to as *levels* where the samples at every level share the same covariate value. For $p = 1, ..., P$, covariate p partitions both treatment and control groups into k_p levels each. Let the partition of the treatment group under covariate p be $L_{p,1}, L_{p,2}, ..., L_{p,k_p}$ of sizes $\ell_{p,1}, \ell_{p,2}, ..., \ell_{p,k_p}$. Similarly, let the partition of the control group under covariate p be $L'_{p,1}, L'_{p,2}, ..., L'_{p,k_p}$ of sizes $\ell'_{p,1}, \ell'_{p,2}, ..., \ell'_{p,k_p}$. Let κ be an integer specifying the ratio of the number of matched control samples to the number of matched treatment samples. The pre-specified integer κ should be within the range $[1, n'/n]$, otherwise the problems are infeasible.

To restrain group-level covariate imbalance, Rosenbaum et al. [46] introduced the concept of *fine balance*, which is a set constraints on the selection of treatment and the selection of control samples.

Definition 3.1 (κ-fine-balance). *For an integer κ, a selection $S \subseteq \mathcal{T}$ of the treatment group and a selection S' of the control group, we say that (S, S')-κ-fine-balance is satisfied if $\kappa \cdot |S \cap L_{p,i}| = |S' \cap L'_{p,i}|$ for $p = 1, ..., P$ and $i = 1, ..., k_p$.*

The definition of the five problems are given as follows.

The *minimum κ-imbalance* (min κ-imbalance) problem is to find a selection S' of control samples of size κn, so as to minimize the κ-imbalance, which is the violation of the $(\mathcal{T}, S')$-κ-fine-balance. We denote this κ-imbalance for selection S' as

$$IM(S') = \sum_{p=1}^{P} \sum_{i=1}^{k_p} ||S' \cap L'_{p,i}| - \kappa \ell_{p,i}|.$$

The *maximum κ-fine-balance selection* (κ-FBS) problem is to select a subset $S \subseteq \mathcal{T}$ and a subset S' of the control group so as to maximize the size of the selection S (equivalent to maximizing the size of S' since $|S'| = \kappa|S|$) where the (S, S')-κ-fine-balance constraints are satisfied.

The third problem studied here is the *κ-fine-balance matching* (κ-BM) problem, first introduced by Rosenbaum et al. [46] for one covariate. Here we are given a distance, or cost, measure between each treatment and each control sample representing the covariate dissimilarity. The κ-BM problem is to minimize the total cost of the assignment of each treatment sample in $\mathcal{T}$ to κ control samples such that the selection of matched control samples S' satisfies $(\mathcal{T}, S')$-κ-fine-balance.

The last two problems could be view as the combination of the third problem, the κ-BM problem, and each of the first two problems respectively. The combination takes the objective from the κ-BM problem and defines the feasible sets to be the optimal solution of the first two problems respectively.

Formally, the *minimum κ-imbalance matching* (κ-MIBM) problem finds the optimal selections to the min κ-imbalance problem in the first stage. In the second stage, among all selections that attain the minimum κ-imbalance, find the selection that minimizes the total distance of an assignment of each selected treatment sample to exactly κ selected control samples.

We define the *maximum selection κ-fine-balance matching* (κ-MSBM) problem in a similar way. In the first stage, the goal is to find the optimal selections for the κ-FBS problem. In the second stage, among all maximum-sized selections, find the selection that minimizes the total distance of an assignment of each selected treatment sample to exactly κ selected control samples.

For the case of $\kappa = 1$, we ignore the prefix κ so (S, S')-κ-fine-balance is called (S, S')-fine-balance, κ-FBS problem is called FBS problem, etc. We summarize ten problems, differentiated between $\kappa = 1$ and $\kappa \geq 2$ for each of the five problem families, in Table 3.1.

Table 3.1: Summary of problems studied here.

Problem name	Objective	Constraints
min imbalance	$\min IM(S)$	$(\mathcal{T}, S')$-fine-balance
min κ-imbalance	$\min IM(S)$	$(\mathcal{T}, S')$-κ-fine-balance
max fine-balance selection (FBS)	$\max \lvert S\rvert$	(S, S')-fine-balance
max κ-fine-balance selection (κ-FBS)	$\max \lvert S\rvert$	(S, S')-κ-fine-balance
fine-balance matching (BM)	min assignment cost	$(\mathcal{T}, S')$-fine-balance
κ-fine-balance matching (κ-BM)	min assignment cost	$(\mathcal{T}, S')$-κ-fine-balance
min imbalance matching (MIBM)	min assignment cost	S' optimal for min imbalance
min κ-imbalance matching (κ-MIBM)	min assignment cost	S' optimal for min κ-imbalance
max selection fine-balance matching (MSBM)	min assignment cost	(S, S') optimal for FBS
max selection κ-fine-balance matching (κ-MSBM)	min assignment cost	(S, S') optimal for κ-FBS

Related literature

There are two models used here for balancing the covariates. The matching model (e.g. [52]) attempts to reduce covariate imbalance by pairing each treatment sample with a similar control sample, or a set of κ similar control samples, for κ a pre-specified integer. The treatment effect estimate is then the average difference in outcomes across all matched pairs or matched sets, where unmatched samples are ignored. The balance optimization subset selection (BOSS) model [41] identifies a selection (subset) of the treatment samples and a selection (subset) of the control samples that minimizes a group-level covariate imbalance measure.

Matching has long been the standard for observational studies because of its potential to balance the entire joint distribution of covariates. In an ideal situation, the samples of the treatment and the control in each matched pair or matched set belong to the same levels over all covariates, which is referred to as *exact matching*. However, satisfying the exact matching requirement may be too restrictive in many cases and it typically results in a very small selection from the treatment and control group, which is not desirable. Thus, the matching model has to compromise on exact matches by permitting inexact matches or by ignoring treatment individuals for whom no close matches exist (known as incomplete matching). Within the realm of inexact matching, one notion of well-matched samples is some distance measure (e.g. the Mahalanobis distance [48], the propensity score [47]) based on the covariate values for each treatment-control pair. A detailed review of matching-related methods used for covariate balancing problems is given by [52].

The BOSS model takes a different approach for observational studies [41, 53, 50]. Instead of seeking matched treatment-control pairs, BOSS seeks a set of control samples that features optimal covariate balance (however that is measured) with respect to the treatment group. This shift from matched pairs to balanced groups makes BOSS more versatile than matching and also provides an optimality guarantee on the balance that is attained, something that matching methods often fail to provide [50]. However, the BOSS model does not have the individual samples matching component which might be desired in some contexts.

The concept of fine balance was first introduced by Rosenbaum et al. [46], who studied the κ-BM problem for the 1-covariate problem and proposed a network flow algorithm. No polynomial running time algorithm has been known for the κ-BM problem with two or more covariates. In fact, the 2-covariate κ-BM problem was shown to be NP-hard in [50].

It is not always feasible to find a selection S' of the control samples that satisfies the $(\mathcal{T}, S')$-κ-fine-balance constraints in the κ-BM problem. To that end, several papers considered the κ-MIBM problem [59, 63, 44]. The κ-MIBM problem is implied to be NP-hard for 2 covariates since the 2-covariate κ-BM problem can be reduced to a 2-covariate κ-MIBM problem. So no polynomial algorithm exists for the 2-covariate κ-BM problem unless P=NP. Yang et al. [59] proposed two network flow algorithms for the case of the 1-covariate problem; Pimental et al. [44] proposed a network flow algorithm for the case in which the covariates form a nested sequence. Zubizarreta [63] considered a different variant which minimizes the total assignment cost of the matched sets with a penalty on the imbalance, and presented a mixed integer programming formulation for an arbitrary number of covariates.

Sauppe et al. [50] and Bennett et al. [3] considered the min κ-imbalance problem, which focuses only on the selection of the control group. The problem is trivial to solve for the 1-covariate problem (see Section 3.2 for details); but for three or more covariates, the problem is NP-hard [50]. Sauppe et al. [50] reformulated the problem as a generalized set cover problem and presented an $(1 - 1/e)$-approximation algorithm. For the case of 2-covariate, Bennett et al. [3] provided an IP formulation for the problem and proved that it is polynomial time solvable by solving the linear programming relaxation.

Summary of our results

We introduce the κ-FBS problem and the κ-MSBM problem here for the first time. Instead of permitting inequality in the $(\mathcal{T}, S')$-κ-fine-balance, they relax the requirement of selecting *all* treatment samples (as in [45]) and replaces it with a maximum size selection possible, while enforcing the κ-fine-balance constraints for the two subsets.

Since the κ-BM problem can be reduced to the κ-MIBM problem or the κ-MSBM problem with the same number of covariates, the NP-hardness of the κ-BM problem for two or more covariates implies the NP-hardness for these two problems for two or more covariates as well. So we mainly discuss the complexity of these problems here. We prove that 1-covariate the κ-MIBM problem is NP-hard when $\kappa \geq 3$. A polynomial algorithm is presented for the 1-covariate MSBM problem, but we leave the complexity status of the 1-covariate 2-MSBM problem open.

Our algorithmic results focus mainly on the two problems under the BOSS model: the min κ-imbalance problem and the κ-FBS problem.

We derived efficient algorithms for the min κ-imbalance problem with two covariates. We present an IP formulation that is slightly different from the one in [3], and show that the constraint matrix is a network matrix. We then show that the two-covariate min κ-imbalance problem can be solved much more efficiently than as a linear program with network flow techniques. We show how to solve the problem as a minimum cost network flow problem with complexity of $O(\kappa n(\min\{n', k_1k_2\} + (k_1 + k_2)\log(k_1 + k_2)))$. We then devise a more efficient algorithm based on a maximum flow formulation of the two-covariate min κ-imbalance problem that runs in $O(n' \cdot \min\{n^{\frac{2}{3}}, n'^{\frac{1}{2}}\} \cdot \log n \cdot \log \kappa n)$ steps.

We prove that for three or more covariates, the κ-FBS problems are NP-hard for any value of κ. For the case of the 2-covariate problem, we present here an efficient algorithm for the FBS problem. The algorithm is based on an integer programming formulation of the problem in which the constraint matrix, for two covariates, has the structure of network flow constraints. For the resulting minimum cost network flow problem we apply an algorithm with running time $O(n \cdot (\min\{n + n', k_1k_2\} + (k_1 + k_2)\log(k_1 + k_2))$. We also prove that for $\kappa \geq 3$, the 2-covariate κ-FBS problem is NP-hard. For the remaining case in which $\kappa = 2$ and the number of covariates is two, the complexity status of the 2-FBS problem is left open.

We observe here that, for *any* number of covariates, if the selections of treatment and control samples are fixed, then the optimal assignment among the selected samples, and therefore the optimal solution to the κ-MIBM and the κ-MSBM problem, can be attained by solving a minimum cost network flow problem. See Section 3.2 for details.

A summary of the complexity results for the five problem families is given in Table 3.2.

Beyond these complexity results, we also address here *fixed-parameter tractable* (FPT) results. We prove that the κ-FBS, κ-BM and MSBM problems are solvable in fixed-parameter tractable time for constant numbers of covariates levels, yet the κ-MSBM problem is NP-hard for constant $\kappa \geq 3$ even when the numbers of covariates levels are constant. It remains an open problem whether the 2-MSBM problem is NP-hard for constant numbers of covariates levels.

Table 3.2: Summary of the complexity and algorithmic results. All based on our results here except for the four noted references.

(Here n is the size of treatment group and n' is the size of control group)

Problem	1-covariate	2-covariate	≥ 3 covariates
min κ-imbalance	$O(n+n')$	$O(\kappa n(\min\{n', k_1k_2\} + (k_1+k_2)\log(k_1+k_2)))$ or $O(n' \cdot \min\{n^{\frac{2}{3}}, n'^{\frac{1}{2}}\} \cdot \log n \cdot \log \kappa n)$	NP-hard [50]
FBS	$O(n+n')$	$O(n(\min\{n+n', k_1k_2\} + (k_1+k_2)\log(k_1+k_2)))$	NP-hard
κ-FBS	$O(n+n')$	NP-hard for $\kappa \geq 3$, open for $\kappa = 2$	NP-hard
κ-BM	$O((n+n')^3)$ [46]	NP-hard [50]	NP-hard
κ-MIBM	$O((n+n')^3)$[59]	NP-hard	NP-hard
MSBM	$O((n+n')nn')$	NP-hard	NP-hard
κ-MSBM	NP-hard for $\kappa \geq 3$ open for $\kappa = 2$	NP-hard	NP-hard

We also consider the 2-covariate BM problem and the 2-covariate κ-BM problem where one of the covariates has a constant number of levels (whereas the other one may have a linear number of levels), and we show that the complexity status of these special cases is tied to the complexity status of the exact matching problem, a problem that is known to have a randomized polynomial time algorithm [39] but the existence of a deterministic polynomial time algorithm is a long-standing open problem.

3.2 The 1-covariate problems and the level-intersections representation

Consider first the case of a single covariate, $P = 1$, that partitions the control and treatment groups into, say, k levels each. Let the sizes of levels of the treatment group be $\ell_1, ..., \ell_k$, and the sizes of levels of the control group be $\ell'_1, ..., \ell'_k$. It is easy to see that there exists a selection S' of control samples that satisfies the $(\mathcal{T}, S')$-κ-fine-balance if and only if $\ell'_i \geq \kappa \ell_i$ for $i = 1, ..., k$. If this condition is satisfied then any subset S^* of the control group with $\kappa \cdot \ell_i$ samples in level i, $i = 1, ..., k$, satisfies the $(\mathcal{T}, S^*)$-κ-fine-balance, and as such is a feasible selection for the κ-BM problem. With these known numbers of control samples to be selected in each level, the optimal solution to the 1-covariate κ-BM problem is found using a minimum cost network flow formulation, as shown next.

The MCNF problem the solution to which is an optimal solution to κ-BM is constructed on a bipartite graph with the treatment samples each represented by a node on one side, and the control samples each represented by a node on the other side. The cost on each arc between a treatment sample and a control sample is the "distance" value between the two, and the arc capacity is 1. Each treatment sample has a supply of κ. To account for the requirement that in each level i of control samples there will be $\kappa \cdot \ell_i$ samples matched we add to the bipartite graph a third layer of k nodes, one for each level. The ith node in the third layer has demand of $\kappa \cdot \ell_i$ and there are arcs to this demand node from all control samples in level i with capacity 1 and cost of 0. In an optimal solution to this MCNF problem the control sample nodes through which there is a positive flow (of one unit) are the ones selected and matched to the respective treatment sample nodes from which they have a positive flow.

If $\ell'_i < \kappa\ell_i$ for some i then there is no selection S' of control samples that satisfies the $(\mathcal{T}, S')$-κ-fine-balance. To compromise on the $(\mathcal{T}, S')$-κ-fine-balance requirement, we can either relax the equality constraints, which leads to the min κ-imbalance problem, or dropping out some treatment samples, which leads to the κ-FBS problem.

The solution to the 1-covariate min κ-imbalance problem is straightforward: in step 1, select $\min\{\kappa \cdot \ell_i, \ell'_i\}$ control samples in level i; if the number of control samples selected is less than κn in step 1, then we select random additional control samples such that the selection is of size κn.

The solution to the 1-covariate κ-FBS problem is also straightforward: select $\bar{\ell}_i = \min\{\ell_i, \lfloor \ell'_i/\kappa \rfloor\}$ treatment samples of level i and $\kappa \cdot \bar{\ell}_i$ control samples of level i.

For the 1-covariate problem of finding an optimal matching, or assignment, among all optimal selections for either the min κ-imbalance or the FBS problem, we solve a MCNF problem for the known number of samples to select from each level, similar to the one defined above with the following modifications. For the minimum κ-imbalance, we first need to change the demand of the demand nodes in the above MCNF problem from $\kappa \cdot \ell_i$ to $\min\{\kappa \cdot \ell_i, \ell'_i\}$ for each level i. We also add a dummy demand node in the third layer with demand $\kappa \cdot n - \sum_{i=1}^{k} \min\{\kappa \cdot \ell_i, \ell'_i\}$, which connects with all control nodes each with capacity 1 and cost of 0. For the optimal selections of FBS, in addition to changing the demand from ℓ_i to $\bar{\ell}_i$ for each level i, we also remove the supply on each treatment sample, add for every level i a supply node with supply $\bar{\ell}_i$, and add arcs from this supply node to all treatment samples in level i with capacity 1 and cost of 0. The best assignment found with a selection that is optimal for the FBS problem is an optimal solution for the MSBM problem. However, this method does not apply to the κ-MSBM problem with $\kappa \geq 2$. We further show that even the 1-covariate κ-MSBM problem is NP-hard for $\kappa \geq 3$ (see Section 3.6).

Hence, all problems discussed here except for the κ-MSBM problem are polynomial time solvable for the 1-covariate case. In Section 3.6 we show that the 1-covariate κ-MSBM does not admit a polynomial time algorithm for $\kappa \geq 3$ unless P=NP.

Consider next the case of multiple covariates. For the min κ-imbalance and the κ-FBS problem, we observe that the selections from the treatment and control groups can be represented compactly in terms of *level-intersections*. For P covariates, the intersection of the level sets $L_{1,i_1} \cap L_{2,i_2} \cap \ldots \cap L_{P,i_P}$, $i_p = 1, \ldots, k_p$, $p = 1, \ldots, P$, form a partition of the treatment group. Similarly, the intersection of the level sets $L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_P}$, $i_p = 1, \ldots, k_p$, $p = 1, \ldots, P$, form a partition of the control group. Therefore, instead of specifying which sample belongs to the selection, it is sufficient to determine the *number* of selected samples in each level intersection for the two groups, since the identity of the specific selected samples has no effect on the fine balance requirement. With this discussion we have a theorem on the representation of the solution to the min κ-imbalance and the κ-FBS problems in terms of the level-intersections sizes.

Theorem 3.1. *The level-intersections sizes $s'_{i_1,i_2,\ldots,i_P}$ are an optimal solution to the min κ-imbalance problem if there exists an optimal selection S' of control samples such that $s'_{i_1,i_2,\ldots,i_P} = |S' \cap L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_P}|$, for $p = 1,\ldots,P$, $i_p = 1,\ldots,k_p$.*

Theorem 3.2. *The level-intersections sizes $s_{i_1,i_2,\ldots,i_P}$ and $s'_{i_1,i_2,\ldots,i_P}$ are an optimal solution to the κ-FBS problem if there exists an optimal selection S of treatment samples and S' of control samples such that $s_{i_1,i_2,\ldots,i_P} = |S \cap L_{1,i_1} \cap L_{2,i_2} \cap \ldots \cap L_{P,i_P}|$ and $s'_{i_1,i_2,\ldots,i_P} = |S' \cap L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_P}|$, for $p = 1,\ldots,P$, $i_p = 1,\ldots,k_p$.*

We will say that the optimal selection for the covariates problems here is *unique* if for any optimal selection S and S', the numbers $s_{i_1,i_2,\ldots,i_P} = |S \cap L_{1,i_1} \cap L_{2,i_2} \cap \ldots \cap L_{P,i_P}|$ and $s'_{i_1,i_2,\ldots,i_P} = |S' \cap L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_P}|$ are unique. In order to derive an optimal selection given the optimal level-intersections sizes, one selects any $s_{i_1,i_2,\ldots,i_P}$ treatment samples from the intersection $L_{1,i_1} \cap L_{2,i_2} \cap \ldots \cap L_{P,i_P}$ and any $s'_{i_1,i_2,\ldots,i_P}$ control samples from the intersection $L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_P}$ for $i_p = 1,\ldots,k_p$, $p = 1,\ldots,P$.

We observe here that, for any number of covariates, if the optimal selection of treatment and control samples in terms of level-intersections is known and unique, then the optimal assignment among the selected samples, and therefore the optimal solution to the κ-MSBM problem, can also be attained by solving an MCNF problem as follows. For each non-zero level intersection of treatment samples there is a source node with supply of $s_{i_1,i_2,\ldots,i_P}$. This source node is connected to all treatment samples in the intersection $L_{1,i_1} \cap L_{2,i_2} \cap \ldots \cap L_{P,i_P}$ with arcs of capacity 1 and cost of 0. For each non-zero level intersection of control samples there is a demand node with supply of $s'_{i_1,i_2,\ldots,i_P}$. This demand node is connected from all control samples in the intersection $L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_P}$ with arcs of capacity 1 and cost of 0. The treatment and control sample nodes through which there is a positive flow (of some unit) are the ones selected, and a positive flow between a treatment node and a control node indicates the two samples are matched. This is a minimum cost network flow problem with a total demand (or supply) bounded by $\min\{n, n'\}$, and $O(nn')$ arcs and $O(n + n')$ nodes. Therefore the successive shortest paths algorithm, discussed below in Section 3.4, solves this problem in $O((n + n')nn')$ steps.

3.3 The minimum κ-imbalance (min κ-imbalance) problem

The min κ-imbalance problem is trivial for 1-covariate (see Section **??**), and the problem is NP-hard with 3 covariates [50]. The 2-covariate problem was shown to be polynomial time solvable by the linear relaxation of the IP formulation presented by Bennett el. al. [3].

In this section, we present a different IP formulation, related to that of Bennett et al [3], and show that the constraint matrix is a network matrix. That implies the two-covariate min κ-imbalance problem can be solved much more efficiently than as a linear program with network flow techniques. We first show how to solve the problem as a minimum cost network flow problem with complexity of $O(\kappa n(\min\{n', k_1k_2\}+(k_1+k_2)\log(k_1+k_2)))$. We then devise a more efficient algorithm based on a maximum flow formulation of the two-covariate min κ-imbalance problem that runs in $O(n' \cdot \min\{n^{\frac{2}{3}}, n'^{\frac{1}{2}}\} \cdot \log n \cdot \log \kappa n)$ steps.

We start by introducing additional notation for the min κ-imbalance problem. Let the levels of the treatment group under covariate $p = 1, \dots, P$ be $L_{p,1}, L_{p,2}, ..., L_{p,k_p}$ of sizes $\ell_{p,1}, \ell_{p,2}, ..., \ell_{p,k_p}$, and let the levels of the control group under covariate p be $L'_{p,1}, L'_{p,2}, ..., L'_{p,k_p}$ of sizes $\ell'_{p,1}, \ell'_{p,2}, ..., \ell'_{p,k_p}$. For a selection S' of control samples we define the discrepancy at level i under covariate p as,

$$dis(S',p,i) = |S' \cap L'_{p,i}| - \ell_{p,i}.$$

The discrepancy of a level can be positive or negative. If the discrepancy is positive we refer to it as excess which is defined as $e_{p,i}(S') = \max\{0, dis(S',p,i)\}$, and if negative, we refer to it as deficit $d_{p,i}(S') = \max\{0, -dis(S',p,i)\}$. With this notation the imbalance of a selection S', $IM(S')$ is,

$$IM(S') = \sum_{p=1}^{P}\sum_{i=1}^{k_p}\left(e_{p,i}(S) + d_{p,i}(S)\right).$$

Notice that this quantity is identical to the imbalance form presented in the introduction: $IM(S') = \sum_{p=1}^{P}\sum_{i=1}^{k_p} ||S' \cap L'_{p,i}| - \ell_{p,i}|$.

We now present the IP formulation that was given by Bennett et al. in [3] for the min-imbalance problem. Although Bennett et al. presented the formulation is for the case of $\kappa = 1$, it can be easily adjusted for the case with general κ. That integer program involves two sets of decision variables:

z_j: a binary variable equal to 1 if control sample j is in the selection S', and 0 otherwise, for $j = 1, \dots, n'$;

$y_{p,i} = |dis(S',p,i)| = ||S' \cap L'_{p,i}| - \ell_{p,i}|$: the absolute value of discrepancy at level i under covariate p, for $p = 1, \dots, P$, and $i = 1, \dots, k_p$. We note that this variable can only assume integer values.

With these variables the formulation is:

$$\min \quad \sum_{p=1}^{P}\sum_{i=1}^{k_p} y_{p,i} \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{j\in L'_{p,i}} z_j - \kappa\ell_{p,i} \le y_{p,i} \qquad p = 1,\dots,P, \quad i = 1,\dots,k_p \tag{3.1b}$$

$$\kappa\ell_{p,i} - \sum_{j\in L'_{p,i}} z_j \le y_{p,i} \qquad p = 1,\dots,P, \quad i = 1,\dots,k_p \tag{3.1c}$$

$$\sum_{j=1}^{n'} z_j = \kappa n \tag{3.1d}$$

$$z_j \in \{0,1\} \qquad j = 1,\dots,n'. \tag{3.1e}$$

For each pair p, i, $p = 1, \ldots, P$ and $i = 1, \ldots, k_p$, constraint (3.1b) and (3.1c) ensure that $y_{p,i}$ assumes the absolute value of the difference between the number of selected level i control samples and $\kappa\ell_{p,i}$ at an optimal solution. These constraints also ensure that any feasible $y_{p,i}$ is non-negative and therefore a non-negativity constraint is not required for variable $y_{p,i}$. The constraint (3.1d) specifies that the size of selected subset equals the size of the treatment group multiplied by κ. This IP formulation does not have the network matrix structure in the constraints.

We next present an IP formulation with network flow constraints for the 2-covariate min κ-imbalance problem. We then show how to solve the problem efficiently with a minimum cost network flow algorithm.

A network flow IP formulation for the 2-covariate min κ-imbalance problem

It was noted, in Theorem 3.1, that there is no differentiation between the individual samples selected in each level intersection, only the number of those selected counts. We thus define the level-intersections variables as follows:
x_{i_1,i_2}: the number of control samples selected from the (i_1, i_2) level intersection $L'_{1,i_1} \cap L'_{2,i_2}$, for $i_1 = 1, ..., k_1, i_2 = 1, ..., k_2$; Let $u_{i_1,i_2} = |L'_{1,i_1} \cap L'_{2,i_2}|$ for $i_1 = 1, ..., k_1$, $i_2 = 1, ..., k_2$. Clearly, x_{i_1,i_2} must be an integer between 0 and u_{i_1,i_2}.

In our formulation, instead of using variables $y_{p,i}$, we use variables for excess and deficit. As discussed in Section 3.2, $||S' \cap L'_{p,i}| - \kappa\ell_{p,i}| = e_{p,i}(S') + d_{p,i}(S')$ for each p and i. We let the variable for excess for p and i be $e_{p,i}$ and the variable for deficit be $d_{p,i}$. Note that $y_{p,i} = e_{p,i} + d_{p,i}$ where both $e_{p,i}$ and $d_{p,i}$ are non-negative variables.

With these decision variables we get the following network flow formulation:

$$
\begin{aligned}
\min \quad & \sum_{p=1}^{2}\sum_{i=1}^{k_p}(e_{p,i}+d_{p,i}) && && (3.2a)\\
\text{s.t.} \quad & \sum_{i_2=1}^{k_2} x_{i_1,i_2}+d_{1,i_1}-e_{1,i_1}=\kappa\ell_{1,i_1} && i_1=1,...,k_1 && (3.2b)\\
& -\sum_{i_1=1}^{k_1} x_{i_1,i_2}-d_{2,i_2}+e_{2,i_2}=-\kappa\ell_{2,i_2} && i_2=1,...,k_2 && (3.2c)\\
& -\sum_{i_1=1}^{k_1} d_{1,i_1}+\sum_{i_1=1}^{k_1} e_{1,i_1}=0 && && (3.2d)\\
& \sum_{i_2=1}^{k_2} d_{2,i_2}-\sum_{i_2=1}^{k_2} e_{2,i_2}=0 && && (3.2e)\\
& e_{p,i},d_{p,i}\ge 0 && p=1,2, \quad k=1,...,k_p && (3.2f)\\
& 0\le x_{i_1,i_2}\le u_{i_1,i_2} && i_1=1,...,k_1, \quad i_2=1,...,k_2 && (3.2g)
\end{aligned}
$$

.

For each p and i, $|S'\cap L'_{p,i}|-\kappa\ell_{p,i}=e_{p,i}-d_{p,i} \Leftrightarrow |S'\cap L'_{p,i}|+d_{p,i}-e_{p,i}=\kappa\ell_{p,i}$. So the constraints (3.2b) and (3.2c), together with the non-negativity constraints (3.2f), ensure that $e_{p,i}$ and $d_{p,i}$ assume the values of respective excess and deficit. Constraints (3.2b) and (3.2c) for the two covariates are separated to facilitate the identification of the network matrix structure. Since $L'_{1,1},...,L'_{1,k_1}$ is a partition of the control group, $\sum_{i=1}^{k_1}|S'\cap L'_{1,i}|=|S'|$. Also, because $\ell_{1,1},...,\ell_{1,k_1}$ are the sizes of the levels partition of the treatment group for the first covariate, it follows that $\sum_{i=1}^{k_1}\ell_{1,i}=n$. Therefore, $\sum_{i=1}^{k_1}(e_{1,i}-d_{1,i})=\sum_{i=1}^{k_1}\left(|S'\cap L'_{1,i}|-\kappa\ell_{1,i}\right)=|S|-\kappa n$. So specifying $|S|=\kappa n$ is equivalent to constraint (3.2d). With similar reason, constraint (3.2e) is also equivalent to setting $|S'|=\kappa n$. Though constraint (3.2e) is redundant, we include it to show the network matrix structure.

In the constraint coefficient matrix of our integer programming formulation, each column has at exactly one 1 and exactly one -1:

(1) Both $\{L'_{1,1},...,L'_{1,k_1}\}$ and $\{L'_{2,1},...,L'_{2,k_2}\}$ are partitions of the control group, so $L'_{1,1},...,L'_{1,k_1}$ are mutually disjoint, and $L'_{2,1},...,L'_{2,k_2}$ are mutually disjoint. The column of each x_{i_1,i_2} has exactly one 1 in rows corresponding to (3.2b), and one -1 in rows corresponding to (3.2c).

(2) For each i, the column of $d_{1,i}$ has exactly one 1 in rows corresponding to (3.2b) and exactly one -1 in rows corresponding to (3.2d); the column of $e_{1,i}$ has exactly one -1 in rows corresponding to (3.2b)) and exactly one 1 in rows corresponding to (3.2d).

(3) For each i, the column of $d_{2,i}$ has exactly one -1 in rows corresponding to (3.2c) and exactly one 1 in rows corresponding to (3.2e); the column of $e_{2,i}$ has exactly one 1 in rows corresponding to (3.2c)) and exactly one -1 in rows corresponding to (3.2e).

The formulation (3.2) is a minimum cost network flow problem. The corresponding network is shown in Figure 3.5, where all capacity lower bounds are 0, and each arc has a cost per unit flow and upper bound associated with it. Nodes of type $(1, i_1)$ each has supply of ℓ_{1,i_1} and nodes of type $(2, i_2)$ each has demand of ℓ_{2,i_2} In Figure 3.5, for each i_1 and i_2, the flow on the arc between node $(1, i_1)$ and node $(2, i_2)$ represents variable x_{i_1,i_2}; arc from node 1 to node $(1, i_1)$ represents the excess e_{1,i_1}; arc to node 1 from any node $(1, i_1)$ represents the deficit d_{1,i_1}; arc from node 2 to any node $(2, i_2)$ represents the deficit d_{2,i_2}; arc to node 2 from any node $(2, i_2)$ represents the excess e_{2,i_2}. So the per unit arc cost should be 1 for arcs between node 1 or 2 and any node in $\{(1, 1), (1, 2), ..., (1, k_1)\} \cup \{(2, 1), (2, 2), ..., (2, k_2)\}$. All other arcs have cost 0. It is easy to verify that constraints (3.2b) are corresponding to the flow balance at nodes $(1, i_1)$ for all i_1, constraints (3.2c) are corresponding to the flow balance at nodes $(2, i_2)$ for all i_2. Constraint (3.2d) is corresponding to the flow balance at node 1, and constraint (3.2e) is corresponding to the flow balance at node 2.

Theorem 3.3. *The 2-covariate min κ-imbalance problem is solved as a minimum cost network flow problem in $O(\kappa n \cdot (\min\{n', k_1k_2\} + (k_1 + k_2) \log(k_1 + k_2)))$ time.*

Proof. We choose the algorithm of *successive shortest paths* that is particularly efficient for a MCNF with "small" total supply to solve the network flow problem of the 2-covariate min κ-imbalance problem.

The successive shortest path algorithm iteratively selects a node s with excess supply (supply not yet sent to some demand node) and a node t with unfulfilled demand and sends flow from s to t along a shortest path in the residual network [33, 32, 6]. The algorithm terminates when the flow satisfies all the flow balance constraints. Since at each iteration, the number of remaining units of supply to be sent is reduced by at least one unit, the number of iterations is bounded by the total amount of supply. For our problem the total supply is $O(\kappa n)$.

At each iteration, the shortest path can be solved with Dijkstra's algorithm of complexity $O(|A| + |V| \log |V|)$, where $|V|$ is number nodes and $|A|$ is number of arcs [56, 12]. In our formulation, $|V|$ is $O(k_1 + k_2)$. Since the number of nonempty sets $L'_{1,i_1} \cap L'_{2,i_2}$ is at most $\min\{n', k_1k_2\}$, the number of arcs $|A|$ is $O(\min\{n', k_1k_2\})$.

Hence, the total running time of applying the successive shortest path algorithm with node potentials on our formulation is $O(\kappa n \cdot (\min\{n', k_1k_2\} + (k_1 + k_2) \log(k_1 + k_2)))$. □

Maximum flow formulation

Here we show a maximum flow (max-flow) formulation for the min κ-imbalance problem with 2 covariates. Unlike the previous formulations, the maximum flow solution requires further manipulation in order to derive an optimal solution to the min κ-imbalance problem

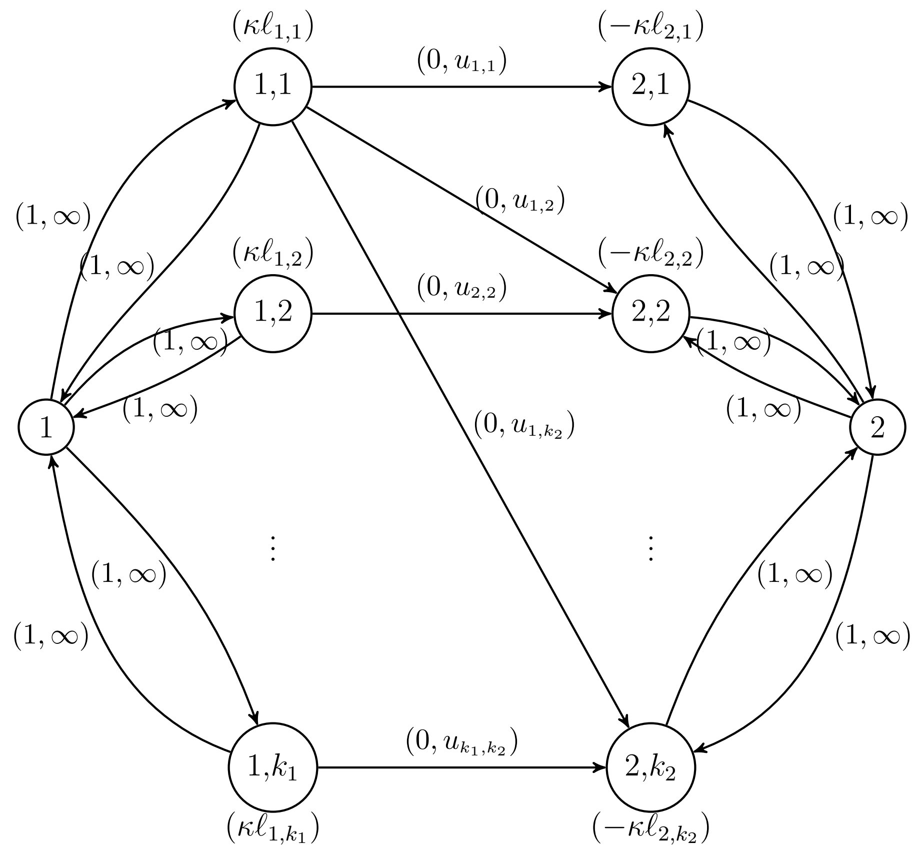


Figure 3.1: Min-cost network flow graph corresponding to formulation (3.2).

arc legend: (cost, upperbound)

with 2 covariate. That max-flow graph is illustrated in Figure (3.3). The source node s can send at most $\kappa\ell_{1,i_1}$ units of flow to node $(1,i_1)$ for each $i_1 = 1,...,k_1$, the sink node can get at most $\kappa\ell_{2,i_2}$ units of flow from node $(2,i_2)$ for each $i_2 = 1,...,k_2$, and there can be a flow from node $(1,i_1)$ to node $(2,i_2)$ with amount bounded by u_{i_1,i_2}, for $i_1 = 1,...,k_1, i_2 = 1,...,k_2$.

Let the maximum flow value for the max-flow problem presented in Figure (3.3) be denoted by f^*, and let $\mathbf{x}^*$ be the optimal flow vector, with $x^*_{i_1,i_2}$ denoting the flow amount between node $(1,i_1)$ and node $(2,i_2)$. It is obvious that $\sum_{i_1=1}^{k_1}\sum_{i_2=1}^{k_2} x^*_{i_1,i_2} = f^* \leq \sum_{i_1=1}^{k_1}\kappa\ell_{1,i_1} = \kappa n$. That means an initial selection S' generated by selecting the prescribed number of control samples as in the optimal max-flow solution, i.e., selecting $x^*_{i_1,i_2}$ control samples from $L'_{1,i_1}\cap L'_{2,i_2}$ is of size f^*. In order to get a feasible solution for the min κ-imbalance problem it is required to select $\kappa n - f^*$ additional control samples. The selection S' has no positive excess, only non-negative deficits with respect to the levels of both covariates. This is because $\sum_{i_2=1}^{k_2} x^*_{i_1,i_2} \leq \kappa\ell_{1,i_1}$ due to the upper bound of arc from s to $(1,i_1)$ for each i_1, and $\sum_{i_1=1}^{k_1} x^*_{i_1,i_2} \leq \kappa\ell_{2,i_2}$ due to the upper bound of arc from $(2,i_2)$ to t for each i_2. To recover an optimal solution for the min κ-imbalance problem from the initial set S', we add up to

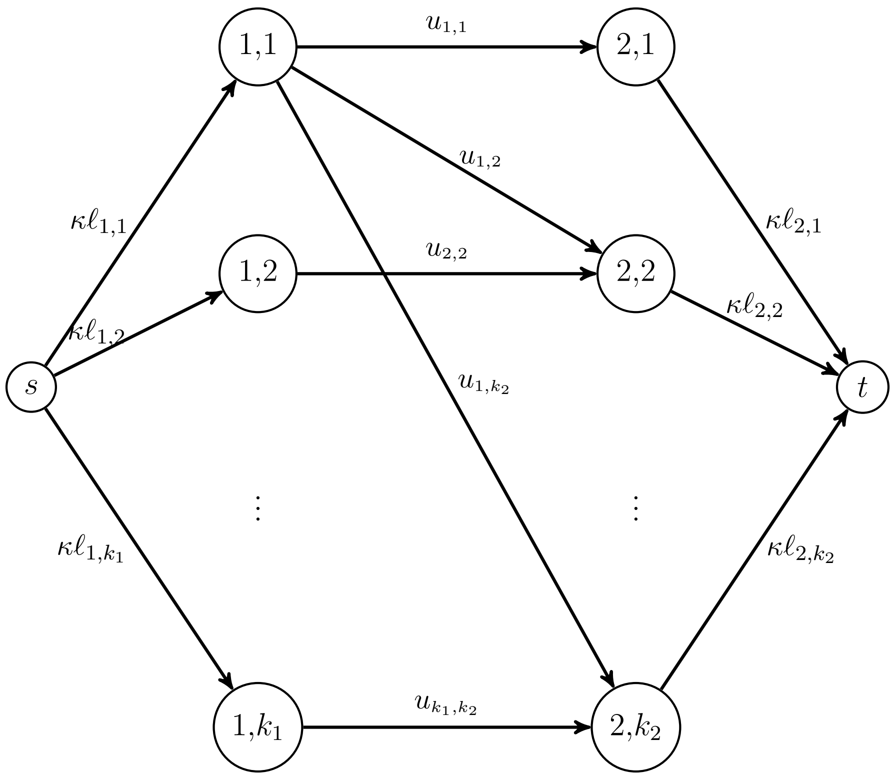


Figure 3.3: Maximum flow graph

arc legend: upperbound

$\kappa n - f^*$ unselected control samples, one at a time, each corresponding to a level with positive deficit under either covariate 1 or 2. This process is repeated until either $\kappa n - f^*$ such control samples are found, or until no such control sample exists. In the latter case, to complete the size of the selection, any randomly selected control samples are added. Algorithm 1 is a formal statement of this process of recovering an optimal solution of the min κ-imbalance problem from the initial selection S'.

Algorithm 1

```
**Initialization step:** Select $x^*_{i_1,i_2}$ number of control samples from $L'_{1,i_1} \cap L'_{2,i_2}$ in set $S'$.
while $|S'| < \kappa n$ do
    if there exists an unselected control sample $j \notin S'$ whose covariate 1 level is $i_1$ and covariate 2 level is $i_2$, such that $|S' \cap L'_{1,i_1}| < \kappa\ell_{1,i_1}$ or $|S' \cap L'_{2,i_2}| < \kappa\ell_{2,i_2}$ then,
        $S' \leftarrow S' \cup \{j\}$.
    else
        Let $S'' = S'$ and let $S^+$ be any $\kappa n - |S'|$ control samples $\notin S'$. Set $S' \leftarrow S' \cup S^+$.
    end if
end while
Output $S'$.
```

To show that Algorithm 1 provides an optimal solution to the min κ-imbalance problem, we distinguish two cases of Algorithm 1: (1) $S^+ = \emptyset$ and (2) $|S^+| \geq 1$. In the first case, there is, at each iteration, at least one control sample that belongs to some level with positive deficit. In Theorem 3.4 we prove that the output S' of Algorithm 1 is an optimal solution in this case.

Theorem 3.4. *If $S^+ = \emptyset$ then the output selection S' of Algorithm 1 is optimal for the min κ-imbalance problem, with an optimal objective value of $2(\kappa n - f^*)$.*

Proof. First, we show that the total imbalance of the selection S' is $IM(S') = 2(\kappa n - f^*)$. At the initialization step the selection S' has only deficits for all levels, with total deficit for covariate 1, $\sum_{i_1=1}^{k_1}(\kappa\ell_{1,i_1} - \sum_{i_2=1}^{k_2} x_{i_1,i_2}) = \kappa n - f^*$, and total deficit for covariate 2, $\sum_{i_2=1}^{k_2}(\kappa\ell_{2,i_2} - \sum_{i_1=1}^{k_1} x_{i_1,i_2}) = \kappa n - f^*$. At each iteration, there is an added control sample, say in $L'_{1,i_1} \cap L'_{2,i_2}$, such that either L'_{1,i_1} or L'_{2,i_2} has a positive deficit with respect to S'. It is however impossible for both L'_{1,i_1} and L'_{2,i_2} to have a positive deficit with respect to S' since otherwise, there is an s,t-augmenting path, from s to node $(1, i_1)$, to node $(2, i_2)$, to t, along which the flow can be increased by at least one unit. This is in contradiction to the optimality of the max-flow solution $\mathbf{x}^*$. As a result, at each iteration where a control sample is added, the total deficit is reduced by one unit, and the total excess is *increased* by one unit. Thus, at each iteration of the **if** step, the sum of total deficit and excess remains the same, namely $2(\kappa n - f^*)$.

Suppose, by contradiction, that there exists a selection S'^* for which the total imbalance is lower, $IM(S^*) < 2(\kappa n - f^*)$. We repeat the following iterative procedure of removing samples from S^* until there is no positive excess remaining: while there is a level of either covariate with positive excess with respect to S'^*, we remove one sample of S'^* that belongs to this level. Each such iteration results in the total excess reducing by at least 1 unit and the total deficit increasing by at most 1 unit, and therefore the sum of total deficit and excess does not increase. So when this iterative procedure ends, the total excess is zero and the total deficit is at most $IM(S'^*)$. Let x_{i_1,i_2} be the number of samples remaining in $S'^* \cap L'_{1,i_1} \cap L'_{2,i_2}$ after this excess removing procedure. Since there is no positive excess, $\mathbf{x}$ is a feasible solution for the max-flow problem with the flow between node $(1, i_1)$ and node $(2, i_2)$ equal to x_{i_1,i_2}. The sum of deficits associated with this remaining set is $\kappa n - \sum_{i_1=1}^{k_1} \sum_{i_2=1}^{k_2} x_{i_1,i_2}$ for covariate 1 and $\kappa n - \sum_{i_1=1}^{k_1} \sum_{i_2=1}^{k_2} x_{i_1,i_2}$ for covariate 2, for a total of $2(\kappa n - \sum_{i_1=1}^{k_1} \sum_{i_2=1}^{k_2} x_{i_1,i_2})$, which is at most $IM(S'^*)$. Therefore, the total flow value, $\sum_{i_1=1}^{k_1} \sum_{i_2=1}^{k_2} x_{i_1,i_2}$, satisfies that it is at least $\kappa n - \frac{IM(S'^*)}{2}$. Since $\kappa n - \frac{IM(S'^*)}{2} > \kappa n - (\kappa n - f^*) = f^*$, it follows that the value of the feasible flow induced by the set S'^* is greater than the maximum flow value f^*, which contradicts the optimality of f^*. $\square$

We now address the second case where $|S^+| \geq 1$ and $|S''| < \kappa n$. In this case, the total imbalance of S'' is, from the arguments in the proof of Theorem 3.4, $IM(S'') = 2(\kappa n - f^*)$. Each one of the S^+ samples selected adds 1 unit of excess to each covariate, resulting in the addition of 2 units of excess to the imbalance. Therefore, the total imbalance of the output solution is $2(\kappa n - f^*) + 2|S^+|$. We next show what the value of $|S^+|$ is, and then demonstrate that any feasible selection to the min κ-imbalance problem has total imbalance of at least $2(\kappa n - f^*) + 2|S^+|$. This will prove that the output of Algorithm 1, S', is an optimal solution to the min κ-imbalance problem.

It will be useful to consider an equivalent form of Algorithm 1. For each level i of covariate p that has $|S' \cap L'_{p,i}| < \kappa \ell_{p,i}$, we add the largest number possible of available control samples in $L'_{p,i}$ so long as the total does not exceed κn. This number is $\min\{\kappa \ell_{p,i} - |S' \cap L'_{p,i}|, \ell'_{p,i} - |S' \cap L'_{p,i}|\}$. Let $\bar{\ell}_{p,i} = \min\{\kappa \ell_{p,i}, \ell'_{p,i}\}$ then for each p, i that has $|S' \cap L'_{p,i}| < \kappa \ell_{p,i}$ we add $\bar{\ell}_{p,i} - |S' \cap L'_{p,i}|$ previously unselected control samples to S'. The outcome of this equivalent procedure is exactly the same as that of Algorithm 1. In the case that $|S^+| \geq 1$ there is an insufficient number of control samples to add to S' after for all levels the largest number possible has been added. Therefore, at the end of this process, the **if** step returns that another unselected control sample does not exist, and the total number of control samples of S'', for each level i of covariate p, is $\bar{\ell}_{p,i}$.

Lemma 3.1. *If $|S^+| \geq 1$ (and $|S''| = n - |S^+| < n$) then $|S^+| = n - (\bar{\ell}_1 + \bar{\ell}_2 - f^*)$ where $\bar{\ell}_1 = \sum_{i_1=1}^{k_1} \bar{\ell}_{1,i_1}$ and $\bar{\ell}_2 = \sum_{i_2=1}^{k_2} \bar{\ell}_{2,i_2}$.*

Proof. At the initialization step of Algorithm 1 $|S'| = f^*$ and the total deficit is $2(\kappa n - f^*)$. Each time a control sample is added to S' in the **if** step, the total deficit is decreased exactly by 1 unit. So we can derive the value of $|S''|$ when the algorithm terminates if we know the total deficit when the algorithm terminates. Note that the total excess may change, but we only consider here the deficit part of the imbalance.

From the discussion above, the total number of control samples of S'', for each level i of covariate p, is $\bar{\ell}_{p,i}$. We denote $\bar{\ell}_1 = \sum_{i_1=1}^{k_1} \bar{\ell}_{1,i_1}$, and $\bar{\ell}_2 = \sum_{i_2=1}^{k_2} \bar{\ell}_{2,i_2}$. Since the sum $\sum_{i_1=1}^{k_1} \ell_{1,i_1} = n$ and $\sum_{i_2=1}^{k_2} \ell_{2,i_2} = n$, the sum of deficits of set S'' under covariate 1 is $\sum_{i_1=1}^{k_1} \kappa \ell_{1,i_1} - \bar{\ell}_{1,i_1} = \kappa n - \bar{\ell}_1$, and the sum of deficits under covariate 2 equals $\sum_{i_2=1}^{k_2} \kappa \ell_{2,i_2} - \bar{\ell}_{2,i_2} = \kappa n - \bar{\ell}_2$. It follows that the sum of deficits of S'' is $2\kappa n - \bar{\ell}_1 - \bar{\ell}_2$.

Since the initial set S' that has total deficit of $2(\kappa n - f^*)$ has its deficit reduced through Algorithm 1 to $2\kappa n - \bar{\ell}_1 - \bar{\ell}_2$ in the set S'', the additional number of control sample in S'' that were added to the initial f^* control samples is $2(\kappa n - f^*) - (2\kappa n - \bar{\ell}_1 - \bar{\ell}_2) = \bar{\ell}_1 + \bar{\ell}_2 - 2f^*$. Therefore, the size of S'' is $f^* + (\bar{\ell}_1 + \bar{\ell}_2 - 2f^*) = \bar{\ell}_1 + \bar{\ell}_2 - f^*$. This number is less than n and the size of S^+ then satisfies, $|S^+| = \kappa n - (\bar{\ell}_1 + \bar{\ell}_2 - f^*)$.

□

Corollary 3.1. *If $|S^+| \geq 1$ when Algorithm 1 terminates, the total imbalance of the output solution S' is $IM(S') = 4\kappa n - 2\bar{\ell}_1 - 2\bar{\ell}_2$.*

Proof. The imbalance of S'', as in the proof of Theorem 3.4, is equal to $2(\kappa n - f^*)$. Since each sample in S^+ adds two units to the imbalance, the total imbalance of the output solution S' is $IM(S') = 2(\kappa n - f^*) + 2(\kappa n - (\bar{\ell}_1 + \bar{\ell}_2 - f^*))$, which is equal to $4\kappa n - 2\bar{\ell}_1 - 2\bar{\ell}_2$, as stated. □

Next, we prove that this is the minimum total imbalance achievable.

Theorem 3.5. *For any selection of size n, the total imbalance must be greater or equal to $4\kappa n - 2\bar{\ell}_1 - 2\bar{\ell}_2$.*

Proof. For the optimal selection S'^* of size κn, let $IM(S'^*)$ be the total imbalance of S'^*. We first classify the samples in S^* into three types, S_1', S_2' and S_3' that forms a partition of S'^*, using the 3-type Classification Procedure.

In the procedure we use variable $dis(p,i)$ to denote the value of $dis(S,p,i)$, discrepancy for control group selection S and level i under covariate p. With this notation the excess of the corresponding level is $e(S,p,i) = \max\{0, dis(p,i)\}$, and the deficit is $d(S,p,i) = \max\{0, -dis(p,i)\}$.

```
    3-type Classification Procedure
Initialize
    $S \leftarrow S'^*, S_1 \leftarrow \emptyset, S_2 \leftarrow \emptyset, S_3 \leftarrow \emptyset$;
    Let $dis(p,i) \leftarrow |S \cap L'_{p,i}| - \kappa\ell_{p,i}$ for $p = 1,2,\ i = 1,...,k_p$;
$S_1$ selection:
    While there exists a sample $j$ in $S$ whose covariate 1 level is $i_1$, covariate 2 level is $i_2$,
such that $dis(1,i_1) > 0$ and $dis(2,i_2) > 0$, pick $j$;
        $S_1 \leftarrow S_1 \cup \{j\}, S \leftarrow S - \{j\}, dis(1,i_1) \leftarrow dis(1,i_1) - 1,\ dis(2,i_2) \leftarrow dis(2,i_2) - 1$;
    end while;
$S_2$ selection:
    While there exists a sample $j$ in $S$ whose covariate 1 level is $i_1$, covariate 2 level is $i_2$,
such that $dis(1,i_1) > 0$ or $dis(2,i_2) > 0$, pick $j$;
        $S_2 \leftarrow S_2 \cup \{j\}, S \leftarrow S - \{j\}, dis(1,i_1) \leftarrow dis(1,i_1) - 1,\ dis(2,i_2) \leftarrow dis(2,i_2) - 1$;
    end while;
$S_3$ selection:
    $S_3 \leftarrow S$;
end.
```

The output S_1, S_2, S_3 of the procedure is not unique, since it depends on the order in which samples are picked. However, the statements of the theorem hold for any output of the procedure. Note that in the procedure, whenever a sample is picked, any $dis(p, i)$ can only go down. For that reason, once S_1 selection ends, there will not be another sample in S for which the discrepancy values of the corresponding levels under the two covariates are both positive. Furthermore, once the S_1 and S_2 selections are done, $dis(p, i) \leq 0$ for each p, i. That means, $|S_3 \cap L'_{p,i}| \leq \kappa\ell_{p,i}$ for all p, i.

Let the sizes of the three subsets be denoted by $s_1 = |S_1|, s_2 = |S_2|, s_3 = |S_3|$.

We claim that the total imbalance of samples in S_3 is $IM(S'^*) - 2s_1$. For the S_1 selection part of the procedure, each samples picked in S_1 reduces the total excess by 2. For each sample selected in the S_2 selection part of the procedure, the excess is reduced by 1, and the deficit is increased by 1. So for each sample selected in the S_2 selection step, the total imbalance does not change. Therefore, the total imbalance of the samples in S_3 is $IM(S'^*) - 2s_1$.

On the other hand, the total imbalance of S_3, which equals the sum of deficits of both covariates (all excesses equal zero as $|S_3 \cap L'_{p,i}| \leq \kappa\ell_{p,i}$), is $2(\kappa n - s_3)$. That says,

$$IM(S'^*) - 2s_1 = 2(\kappa n - s_3).$$

Hence,

$$IM(S'^*) = 2(\kappa n - s_3) + 2s_1 = 2(\kappa n - s_3) + 2(\kappa n - s_2 - s_3) = 4\kappa n - 2s_2 - 4s_3.$$

Here, the second equality comes from the fact that $s_1 + s_2 + s_3 = \kappa n$.

Next, we show that $s_2 \leq (\bar{\ell}_1 - s_3) + (\bar{\ell}_2 - s_3)$. Let the samples in S_2 be ordered according to the order they were picked, $j_1, j_2, ..., j_{s_2}$. We now add these samples to S_3, in the reverse order $j_{s_2}, ..., j_1$. When each sample j_q is added to S_3, the deficit is reduced by exactly 1 unit. Once all the S_2 samples are added to S_3 the deficit at each level of $S_2 \cup S_3$ is zero, or alternatively, $dis(S_2 \cup S_3, p, i) = |(S_2 \cup S_3) \cap L'_{p,i}| - \kappa\ell_{p,i} \geq 0$ for each p, i.

We now consider the total deficit of S_3: By the definition of $\bar{\ell}_1$ and $\bar{\ell}_2$, the positive deficit of S_3 under covariate 1 is at most $\bar{\ell}_1 - s_3$ and that the positive deficit of S_3 under covariate 2 is at most $\bar{\ell}_2 - s_3$. That means the size of S_2 is bounded by the amount of this deficit, $s_2 \leq (\bar{\ell}_1 - s_3) + (\bar{\ell}_2 - s_3)$.

Now we have,

$$\begin{aligned} s_2 \leq (\bar{\ell}_1 - s_3) + (\bar{\ell}_2 - s_3) \quad &\Leftrightarrow \quad s_2 + 2s_3 \leq \bar{\ell}_1 + \bar{\ell}_2 \\ &\Leftrightarrow \quad IM(S'^*) = 4\kappa n - 2s_2 - 4s_3 \geq 4\kappa n - 2\bar{\ell}_1 - 2\bar{\ell}_2. \end{aligned}$$

We conclude that the total imbalance $IM(S^*)$ is at least $4\kappa n - 2\bar{\ell}_1 - 2\bar{\ell}_2$. That implies that the selection output of Algorithm 1, S', which has a total imbalance of $4\kappa n - 2\bar{\ell}_1 - 2\bar{\ell}_2$, is optimal.

□

The conclusion from Corollary 3.1 and Theorem 3.5, is that for $|S^+| \geq 1$ when Algorithm 1 terminates, the output solution S' is an optimal selection to the min κ-imbalance problem.

Together with Theorem 3.4, we have that Algorithm 1 outputs an optimal selection for the min κ-imbalance problem using the max-flow solution to the flow problem in Figure 3.3 as input.

Theorem 3.6. *The maximum flow formulation of the 2-covariate min κ-imbalance problem is solved the in $O(n' \cdot \min\{n^{\frac{2}{3}}, n'^{\frac{1}{2}}\} \cdot \log n \cdot \log \kappa n)$ time.*

Proof. We choose here the *binary blocking flow* algorithm of Goldberg and Rao [16] for solving the max-flow problem shown in Figure 3.3 because this algorithm depends on the maximum arc capacity which is a small quantity in our formulation.

The complexity of the binary blocking flow algorithm for a graph $G = (V, A)$ is $O(|A| \cdot \min\{|V|^{\frac{2}{3}}, |A|^{\frac{1}{2}}\} \cdot \log \frac{|V|^2}{|A|} \log U)$ where $|V|$ is number of nodes, $|A|$ is number of arcs, and U is maximum arc capacity. As argued earlier for the minimum cost network flow formulation, the number of nodes in the network $|V|$ is $O\left(k_1 + k_2\right)$, which is no more than $O(n)$; and the number of arcs is bounded by $\min\{n', k_1k_2\}$. Although u_{i_1,i_2} could be as large as n', a feasible flow to our maximum flow formulation can not have more than $\kappa\ell_{1,i_1}$ units of flow on the arc from node $(1, i_1)$ to node $(2, i_2)$. Thus the maximum arc capacity U is effectively $O(\kappa n)$. The ratio $\frac{|V|^2}{|A|} \le \frac{(k_1+k_2)^2}{k_1+k_2} \le n$. Hence, the running time of applying the binary blocking flow algorithm to our max-flow problem is $O(n' \cdot \min\{n^{\frac{2}{3}}, n'^{\frac{1}{2}}\} \cdot \log^2 n)$.

The complexity of Algorithm 1 is $O(n)$ as the number of iterations is bounded by n, and each iteration takes $O(1)$ steps.

Therefore, the running time of solving the min κ-imbalance problem as a max-flow problem is $O(n' \cdot \min\{n^{\frac{2}{3}}, n'^{\frac{1}{2}}\} \cdot \log n \cdot \log \kappa n)$. □

3.4 The maximum κ-fine-balance selection (κ-FBS) problem

In this section we show the complexity and algorithmic results for the κ-FBS problems. We present the results separately first for the 1-covariate problem, next for three or more covariates, then the 2-covariate FBS problem, and finally the 2-covariate κ-FBS problem for $\kappa \ge 3$.

The solution to the 1-covariate κ-FBS problem is straightforward, as discussed in Section 3.2: select $\bar{\ell}_i = \min\{\ell_{1,i}, \lfloor \ell'_{1,i}/\kappa \rfloor\}$ number of level i treatment samples and $\kappa \cdot \bar{\ell}_i$ number of level i control samples. The union of the selections at each level is an optimal solution for the 1-covariate κ-FBS problem.The value of the objective function corresponding to this solution is $\sum_{i=1}^{k_1} \bar{\ell}_i = \sum_{i=1}^{k_1} \min\{\ell_{1,i}, \lfloor \ell'_{1,i}/\kappa \rfloor\}$.

NP-hardness for the κ-FBS problem for any constant κ with $P \ge 3$

We show here that even for κ being constant, the κ-FBS problem with three or more covariates is NP-hard by reducing the *3-Dimensional Matching* problem, one of Karp's 21 NP-hard

problems [35][14].
3-Dimensional Matching: Given a finite set X and a set of triplets $U \subset X \times X \times X$. Is there a subset $M \subseteq U$ such that $|M| = |X|$ and that no two elements of M agree in any coordinate?

Theorem 3.7. *The κ-FBS problem is NP-hard when $P = 3$ even for constant κ.*

Proof. Given an instance of 3-dimensional matching problem with a finite set X and a set of triplets $U \subset X \times X \times X$, we construct an instance of κ-FBS problem with $P = 3$ for any constant κ. Without loss of generality, we assume $X = \{1, ..., |X|\}$.

First we define the levels of the three covariates. For $p = 1, 2, 3$, the set of levels of covariate p is $\{1, ..., |X|\} \cup \{0, 0'\}$. So all the three covariates have $|X| + 2$ levels.

Next, we construct the samples. For each sample, we represent it by the ordered triplet (a, b, c) where a is the level of the first covariate, b is the level of the second covariate, and c is the level of the third covariate. The treatment group contains a sample (i, i, i) for $i = 1, ..., |X|$ as well as $|X|$ copies of $(0, 0, 0)$ and $|X|$ copies of $(0', 0', 0')$. For each triplet $u \in U$, whose elements are denoted by $[u_1, u_2, u_3]$, we create one control sample (u_1, u_2, u_3). In addition, for each element $i \in X$ we have $(\kappa - 1)$ copies for each of the three control samples $(i, 0', 0), (0', 0, i), (0, i, 0')$. We also create $|X|$ copies of $(0, 0, 0)$ and $|X|$ copies of $(0', 0', 0')$ for the control group. That is, the control group is the union of the following three sets (we represent the different copies by a superscript as shown below.):

$$\begin{aligned} C_1 &= \{(u_1, u_2, u_3) : \forall u = [u_1, u_2, u_3] \in U\}, \\ C_2 &= \{(i, 0', 0)^{(w)}, (0', 0, i)^{(w)}, (0, i, 0')^{(w)} : \forall i = 1, ..., |X|, \forall w = 1, ..., \kappa - 1\}, \\ C_3 &= \{(0, 0, 0)^{(w)}, (0', 0', 0')^{(w)} : \forall w = 1, ..., |X|\}. \end{aligned}$$

The treatment group constructed is of size $3|X|$ and the control group constructed is of size $|U| + (3\kappa - 1)|X|$. The two sizes are both polynomially bounded in the size of the 3-dimensional matching instance so the reduction can be computed in polynomial time.

Finally, we claim that the optimal value of the constructed 3-covariate κ-FBS problem is $3|X|$ if and only if there exist a subset $M \subseteq U$ such that $|M| = |X|$ and that no two elements of M agree in any coordinate for the 3-dimensional matching instance.

Let $M \subseteq U$ be the solution for the 3-dimensional matching instance, we derive a solution for the constructed problem as follows. We select all the treatment samples. For each triplet $u \in M$, we choose the control sample in C_1 whose covariates levels are corresponding to the elements in u. Additionally, we also choose all control samples in C_2 and C_3. To check the feasibility of this solution, first consider the appearances of level i for each $i = 1, ..., |X|$ under each covariate $p = 1, 2, 3$: level i appears once in the treatment group for each covariate p; it appears once among the selected samples in C_1 as i appears once in each coordinate in M; it appears for $\kappa - 1$ times in C_2; it does not appear in C_3. That is, there is exactly one selected treatment sample of level i under covariate p and κ selected control samples of level i under covariate p, for each i and p. Next, consider appearances of level 0 and $0'$ under each covariate $p = 1, 2, 3$: they each appear $|X|$ times in the treatment group; they do not appear in C_1; they each appear $(\kappa - 1)|X|$ times in C_2 and $|X|$ times in C_3. That is, 0 and $0'$ each appear $\kappa|X|$ times in the selected control samples under each covariate. Therefore, this selection is feasible and the objective value, the number of selected treatment samples, is $3|X|$.

On the other hand, if the constructed 3-covariate κ-FBS problem has an optimal solution S, S' of objective value $3|X|$, we say that $u = [u_1, u_2, u_3] \in U$ is selected for M if the control sample (u_1, u_2, u_3) is selected in S' for the constructed problem. We will show that M is a feasible solution of the 3-dimensional matching instance. Since the size of the treatment group is $3|X|$, all treatment samples must be selected in the optimal solution, and that $3\kappa|X|$ number of control samples must be selected. For each covariate $1, 2, 3$, levels 0 and $0'$ each appears $|X|$ times in the treatment group, so the number of appearance of each of these two levels must be $\kappa|X|$ in the selection S' of the control samples. So all samples in C_2 and C_3 must be selected, otherwise there is no enough level 0 or level $0'$ samples in S'. Therefore, $M = 3\kappa|X| - |C_2| - |C_3| = |X|$. Furthermore, for each covariate p and for $i = 1, ..., |X|$, level i appears exactly once in the treatment group so there are κ number of selected control samples in level i. For each i under each covariate p, since there are $(\kappa - 1)$ number of samples in level i in $C_2 \cup C_3$, only one sample in C_1 in that same level is selected. So there is no overlap in each coordinate for any two triplets in M.

With the above arguments, any 3-dimensional matching problem can be reduced to a 3-covariate κ-FBS problem for any constant integer κ, and hence, the κ-FBS problem is NP-hard for any such κ when $P = 3$. □

Corollary 3.2. *The κ-FBS problem is NP-hard for any integer $P \geq 3$, even for constant κ.*

Proof. For any constant integer κ, any 3-covariate κ-FBS problem, and any $P > 3$, we can construct an equivalent P-covariate κ-FBS problem as follows: for each sample of the given 3-covariate κ-FBS problem, we create a sample for the constructed κ-FBS problem such that they have the same level value for covariate $p = 1, 2, 3$. For $p = 4, ..., P$, set covariate p to have only one level so all samples in the constructed κ-FBS problem have the same value.

Therefore, the NP-hardness of 3-covariate κ-FBS problem implies that the P-covariate κ-FBS problem is NP-hard for every value of P when $P \geq 3$. □

Since the κ-FBS problem is NP-hard for $P \geq 3$, there is no polynomial time algorithm unless $P = NP$.

In the following subsections, we will discuss the remaining case of the 2-covariate problems.

The network flow algorithm for FBS with $P = 2$

In this subsection, we present an integer programming formulation with network flow constraints for the 2-covariate FBS problem. We then show how to solve the problem efficiently with a network flow algorithm.

It was noted, in Theorem 3.2, that there is no differentiation between the individual samples selected in each level intersection, only the number of those selected counts. We thus define the decision variables as follows:

x_{i_1,i_2}: the number of treatment samples selected from the (i_1,i_2) level intersection $L_{1,i_1}\cap L_{2,i_2}$, for $i_1=1,...,k_1$ and $i_2=1,...,k_2$;

x'_{i_1,i_2}: the number of control samples selected from the (i_1,i_2) level intersection $L'_{1,i_1}\cap L'_{2,i_2}$, for $i_1=1,...,k_1$ and $i_2=1,...,k_2$.

Let $u_{i_1,i_2}=|L_{1,i_1}\cap L_{2,i_2}|$ and $u'_{i_1,i_2}=|L'_{1,i_1}\cap L'_{2,i_2}|$ for $i_1=1,...,k_1$, $i_2=1,...,k_2$. Clearly, x_{i_1,i_2} must be an integer between 0 and u_{i_1,i_2}, and x'_{i_1,i_2} must be an integer between 0 and u'_{i_1,i_2}. With these decision variables the following is an integer programming formulation for the 2-covariate FBS problem:

$$\text{(IP-FBS)}\quad \max \quad \sum_{i_1=1}^{k_1}\sum_{i_2=1}^{k_2} x_{i_1,i_2} \tag{3.3a}$$

$$\text{s.t.}\quad \sum_{i_2=1}^{k_2} x_{i_1,i_2}-\sum_{i_2=1}^{k_2} x'_{i_1,i_2}=0 \qquad i_1=1,...,k_1 \tag{3.3b}$$

$$\sum_{i_1=1}^{k_1} x_{i_1,i_2}-\sum_{i_1=1}^{k_1} x'_{i_1,i_2}=0 \qquad i_2=1,...,k_2 \tag{3.3c}$$

$$0\le x_{i_1,i_2}\le u_{i_1,i_2} \qquad i_1=1,...,k_1,\quad i_2=1,...,k_2 \tag{3.3d}$$

$$0\le x'_{i_1,i_2}\le u'_{i_1,i_2} \qquad i_1=1,...,k_1,\quad i_2=1,...,k_2 \tag{3.3e}$$

$$x_{i_1,i_2},x'_{i_1,i_2}\ \text{integers} \qquad i_1=1,...,k_1,\quad i_2=1,...,k_2 \tag{3.3f}$$

.

The objective (3.3a) is the total number of selected treatment samples. Constraints (3.3b) are the fine balance requirement under covariate 1, as $\sum_{i_2=1}^{k_2} x_{i_1,i_2}$ equals the number of selected treatment samples in level i_1 under covariate 1 and $\sum_{i_2=1}^{k_2} x'_{i_1,i_2}$ equals the number of selected control samples in the same level. Similarly, constraints (3.3c) are the fine balance requirement under covariate 2.

Formulation (IP-FBS) is in fact also a network flow formulation. In a minimum cost network flow formulation, each column of the constraint matrix corresponding to a variable that is a flow along an arc, has exactly one 1 and one -1. The corresponding MCNF network is shown in Figure 3.5, where all capacity lower bounds are 0, and each arc has a cost per unit flow and upper bound associated with it. The flow on the arc from node $(1,i_1)$ to node $(2,i_2)$ represents variable x_{i_1,i_2}, which is bounded between 0 and u_{i_1,i_2} as stated in constraints (3.3d); arc from node $(2,i_2)$ to node $(1,i_1)$ represents variable x'_{i_1,i_2}, which is bounded between 0 and u'_{i_1,i_2} as stated in constraints (3.3e). To get a "minimize" type objective, we take the negative value of $|S|=\sum_{i_1=1}^{k_1}\sum_{i_2=1}^{k_2} x_{i_1,i_2}$ as the objective, so the per unit arc cost should be -1 for arcs from any node in $\{(1,1),(1,2),...,(1,k_1)\}$ to any node in $\{(2,1),(2,2),...,(2,k_2)\}$. All other arcs have cost 0. It is easy to verify that constraints

(3.3b) are corresponding to the flow balance at nodes $(1, i_1)$ for all i_1, constraints (3.3c) are corresponding to the flow balance at nodes $(2, i_2)$ for all i_2.

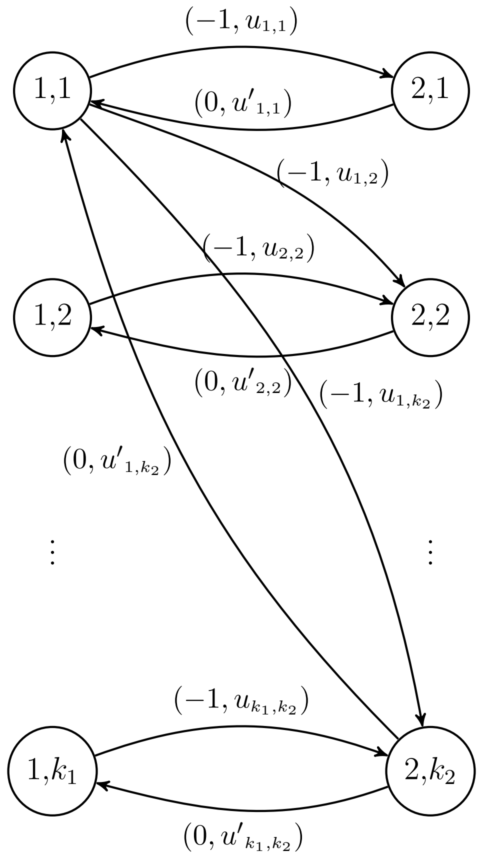


Figure 3.5: Min-cost network flow graph corresponding to formulation (IP-FBS) .

arc legend: (cost, upperbound)

Theorem 3.8. *The 2-covariate FBS problem is solved as a minimum cost network flow problem in* $O(n \cdot (\min\{n + n', k_1k_2\} + (k_1 + k_2) \log(k_1 + k_2))$ *time.*

Proof. To solve the minimum cost network flow problem of the 2-covariate FBS problem, we choose the algorithm of *successive shortest paths* that is particularly efficient for a MCNF with “small” total arc capacity (see [1] Section 9.7). The successive shortest paths algorithm starts with a network graph with no negative cycles, so we first modify the network shown in **??** using a well-known arc reversal transformation in [1] Section 2.4. The resulting network graph is shown in Figure 3.7.

The successive shortest path algorithm iteratively selects a node s with excess supply (supply not yet sent to some demand node) and a node t with unfulfilled demand and sends flow from s to t along a shortest path in the residual network [33, 32, 6]. The algorithm

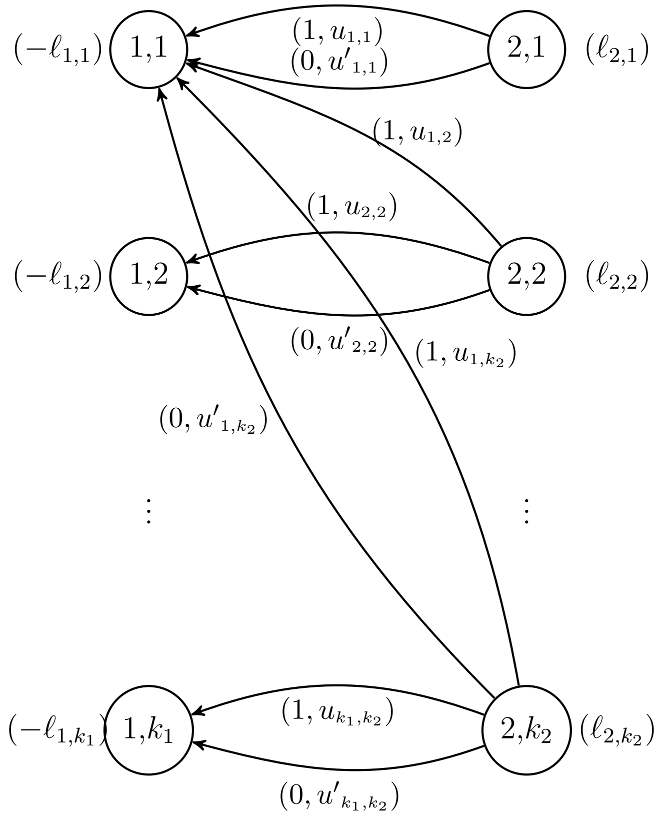


Figure 3.7: Min-cost network flow graph after arc reversal.

arc legend: (cost, upperbound); node legend: (supply)

terminates when the flow satisfies all the flow balance constraints. Since at each iteration, the number of remaining units of supply to be sent is reduced by at least one unit, the number of iterations is bounded by the total amount of supply. For the network in Figure 3.7 the total supply is n.

At each iteration, the shortest path can be solved with Dijkstra's algorithm of complexity $O(|A| + |V| \log |V|)$, where $|V|$ is number nodes and $|A|$ is number of arcs [56, 12]. In our formulation, $|V|$ is $O(k_1 + k_2)$, which is at most $O(n)$. Since the number of nonempty sets $L_{1,i_1} \cap L_{2,i_2}$ is at most $\min\{n, k_1k_2\}$, the number of unit-cost arcs is $O(\min\{n, k_1k_2\})$. Since the number of nonempty sets $L'_{1,i_1} \cap L'_{2,i_2}$ is at most $\min\{n', k_1k_2\}$, the number of zero-cost arcs is $O(\min\{n', k_1k_2\})$. So the total number of arcs $|A|$ is $O(\min\{n + n', k_1k_2\})$.

Hence, the total running time of applying the successive shortest path algorithm on our formulation is $O(n \cdot (\min\{n + n', k_1k_2\} + (k_1 + k_2) \log(k_1 + k_2))$. $\square$

In contrast to the 2-covariate FBS problem which is polynomial time solvable, we show next that the 2-covariate κ-FBS problem is NP-hard when $\kappa \geq 3$.

NP-hardness for the 2-covariate κ-FBS problem with $\kappa \geq 3$

We prove here that the 2-covariate κ-FBS problem is NP-hard for all constant values of κ such that $\kappa \geq 3$. The proof reduces the *exact 3-cover* problem, which is NP-hard [35, 14]. *Exact 3-cover*: Given a collection C of 3-element subsets (triplets) of a ground set E with $|E| = 3q$ for some integer q, is there a subcollection $C' \subseteq C$ where each element $e \in E$ appears in exactly one triplet of C'?

Theorem 3.9. *The 2-covariate κ-FBS problem is NP-hard for any constant $\kappa \geq 3$.*

Proof. Given an instance of Exact 3-cover problem with ground set E of size $3q$ and a collection of triplets C, we construct an instance of 2-covariate κ-FBS problem for any constant integer $\kappa \geq 3$.

First we define the levels of the two covariates. For covariate 1, we have a level T for each triplet $T \in C$, and a level e' for each element $e \in E$. So there are $|C| + |E|$ levels of covariate 1. For covariate 2, we have a level e'' for each element $e \in E$, and one additional level dentoed as X. So there are $|E| + 1$ levels of covariate 2.

Next, we construct the samples. For each sample, we represent it by the ordered pair (a, b) where a is the level of the first covariate and b is the level of the second covariate. The treatment group contains a sample (T, X) for each triplet $T \in C$ and a sample (e', e'') for each element $e \in E$. Moreover, for each triplet $T \in C$, whose elements are denoted by $e_{t_1}, e_{t_2}, e_{t_3}$, we create three control samples $(T, e''_{t_1}), (T, e''_{t_2}), (T, e''_{t_3})$, as well as $(\kappa - 3)$ copies of control sample (T, X). In addition, for each element $e \in E$ we have one control sample (e', X) and $(\kappa - 1)$ copies of control sample (e', e''). The treatment group constructed is of size $|C| + |E|$ and the control group constructed is of size $\kappa|C| + \kappa|E|$. The two sizes are both polynomially bounded in the size of the Exact 3-cover instance so the reduction can be computed in polynomial time.

Finally, we claim that the constructed 2-covariate κ-FBS problem has a feasible solution with objective value of at least $4q$ if and only if the Exact 3-cover instance has a subcollection $C' \subseteq C$ such that each element $e \in E$ appears in exactly one triplet of C'.

Let C' be a subcollection such that each element $e \in E$ appears in exactly one triplet of C', we derive a solution for the constructed problem as follows. In the treatment group, we choose (T, X) for all $T \in C'$, and (e', e'') for all $e \in E$. In the control group, for each $T \in C'$, whose elements are denoted by $e_{t_1}, e_{t_2}, e_{t_3}$, we choose $(T, e''_{t_1}), (T, e''_{t_2}), (T, e''_{t_3})$ and the $(\kappa - 3)$ copies of (T, X). Additionally, we also choose from the control group sample (e', X) and $(\kappa - 1)$ copies of (e', e'') for each $e \in E$. That is, the selection of treatment samples is

$$S = \{(T, X) : \forall T \in C'\} \cup \{(e', e'') : \forall e \in E\}$$

and the selection of control samples is

$$\begin{aligned} S' = &\{(T, e''_{t_1}), (T, e''_{t_2}), (T, e''_{t_3}) : \forall T = (e_{t_1}, e_{t_2}, e_{t_3}) \in C'\} \cup \{(T, X)^{(w)}) : \forall T \in C', w = 1, ..., \kappa - 3\} \cup \\ &\{(e', X) : \forall e \in E\} \cup \{(e', e'')^{(w)} : \forall e \in E, w = 1, ..., \kappa - 1\}. \end{aligned}$$

To check the feasibility of this solution, first consider the levels of the first covariate. For each $T \in C$, if $T \in C'$ then there is exactly one treatment sample with level T in S and we choose exactly κ such control samples for S'; and if $T \notin C'$, then we have not chosen any sample of this level for neither the treatment nor the control group. Furthermore, for each $e \in E$, we choose exactly one sample from the treatment group with covariate 1 level e' and exactly κ control samples with covariate 1 level e'. Next, consider the levels of the second covariate. We choose $|C'|$ number of level X treatment samples and $(\kappa-3)|C'|+|E|$ number of level X control samples. Note that the size of subcollection C' must be q as each element in E appears in exactly one triplet of C', so $(\kappa-3)|C'|+|E| = \kappa q = \kappa|C'|$. For every $\bar{e} \in E$, we choose exactly one treatment sample with level $\bar{e}''$. There are $\kappa - 1$ control samples of level $\bar{e}''$ under covariate 2 in the set $\{(e', \bar{e}'')^{(w)} : \forall e \in E, w = 1, ..., \kappa-1\}$. Since each element $\bar{e}$ appears in exactly one triple of C', $\bar{e}''$ appears exactly once in the set $\{(T, e''_{t_1}), (T, e''_{t_2}), (T, e''_{t_3}) : \forall T = (e_{t_1}, e_{t_2}, e_{t_3}) \in C'\}$. So there are κ control samples of level $\bar{e}''$ under covariate 2 in the selection S'. Therefore, this solution is feasible and the objective value of this solution is $|S| = |C'| + |E| = 4q$.

On the other hand, if the constructed 2-covariate κ-FBS problem has a feasible solution S, S' of objective value at least $4q$, we say that $T \in C$ is selected for subcollection C' if the treatment sample (T, X) is selected in S for the constructed problem. We will show that the subcollection of selected triplets is a feasible solution of the Exact 3-cover problem. Since there are only $3q$ samples in the treatment group that do not correspond to a triplets, the size of C' must be at lease q. So the subcollection is feasible if we can establish that every pair of selected triplets is disjoint. Assume by contradiction that there exist two selected subsets $T, T' \in C'$ that have a common element e. Due to (S, S')-κ-fine-balance, we know the number of control samples in each level under any covariate must be an integer multiple of κ. Since there is only one sample of level $\bar{e}''$ in the treatment group, in selection S' there can be either 0 or κ samples of level $\bar{e}''$ under covariate 2. Since both (T, X) and (T', X) are in the selection S, control samples $(T, \bar{e}'')$ and $(T', \bar{e}'')$ must be chosen in S', otherwise the number of selected samples in covariate 1 levels T and T' will not satisfy (S, S')-κ-fine-balance. Thus, the number of samples in S' with the second covariate being $\bar{e}''$ is at least 2. So the number $|S' \cap L'_{2,\bar{e}''}|$ must be κ, the number $|S \cap L_{2,\bar{e}''}|$ must be 1, and the number of sample $(\bar{e}', \bar{e}'')$ in selection S' must be less than or equal to $\kappa - 2$. This implies that the number of control samples of level $\bar{e}'$ under covariate 1 can not be more than $\kappa - 1$, which means, it must be zero. So we can derive that treatment sample $(\bar{e}', \bar{e}'')$ is not selected, which means the number of treatment samples in level e'' under covariate 2 is 0, contradicts with $|S \cap L_{2,e''}| = 1$.

□

3.5 The κ-balanced-matching (κ-BM) problem with $P \geq 2$

The 1-covariate κ-BM problem is solvable in polynomial time [46]. Sauppe et al. [50] showed that the 2-covariate κ-BM problem is NP-hard. The NP-hard statement can be extended for more than 2 covariates.

Corollary 3.3. *The P-covariate κ-BM problem is NP-hard for every value of P when $P \geq 2$.*

Proof. For any 2-covariate κ-BM problem, and any $P > 3$, we can construct an equivalent P-covariate κ-BM problem by adding $P - 2$ covariates for each sample in the 2-covariate κ-BM problem instance and set the value of the pth covariate to be the same for all samples, for each $p = 3, ..., P$.

Therefore, the NP-hardness of 2-covariate κ-BM problem implies that the P-covariate κ-BM problem is NP-hard as long as $P \geq 2$. □

We will show next that the 2-covariate BM problem and the 2-covariate κ-BM problem when the second covariate has a constant number of levels can be solved efficiently if and only if the exact matching problem on bipartite graphs can be solved efficiently. Note that if both covariates have constant numbers of levels, then the results of Section 3.7 give a polynomial time algorithms for the two problems. Here we consider an intermediate case where only one of the covariates has a constant number of levels.

The special case of 2-covariate BM and 2-covariate κ-BM problems where the second covariate has a constant number of levels

Let BM' be the special case of 2-covariate BM problem where the second covariate has a constant number of levels while the first covariate has super-constant and perhaps linear number of levels. In Section 3.7 we will establish that if both covariates have constant number of levels then the 2-covariate BM problem is polynomial time solvable. Here, we show that relaxing this constraint for one of the covariates is connected to the study of the *exact matching problem*. In order to present this connection we assume that the distance matrix is integral and all distances are given in unary, that is, there is a polynomial π of the input encoding length where $\delta_{ij} \leq \pi$ for all i, j.

The exact matching in bipartite graph problem is defined as follows. The input is an integer number k together with a bipartite graph $G = (V, E)$ with bipartition of the node set V into $V_1 \cup V_2$, and its edge set E is partitioned into $E_b \cup E_r$ where E_b is the set of blue edges and E_r is the set of red edges. The exact matching problem is to find a perfect matching that has exactly k blue edges (and all other edges are red). The complexity status of the exact matching problem is as follows. While [39] showed that there is a randomized polynomial time algorithm for the problem, the existence of a deterministic polynomial time algorithm is still an important open problem. We show the following connection between the exact matching problem and problem BM'.

Theorem 3.10. *There is a deterministic (or randomized) polynomial time algorithm for BM' if and only if there is a deterministic (or randomized, respectively) polynomial time algorithm for exact matching in bipartite graphs.*

Proof. Assume that there is a polynomial time algorithm ALG for BM' and we will establish the existence of a polynomial time algorithm for the exact matching problem. Given an input to the exact matching problem with $2q$ nodes (q nodes on each side of the bipartite graph), we denote the bipartition of the graph by $V_1 \cup V_2$ and the partition of the edge set into $E_b \cup E_r$, and we let k be the number of the required blue edges in the matching. We define the following input for BM'. We will associate samples with nodes so the control group consists of nodes and the treatment group also consists of nodes. For every node $v \in V_1$, we have two nodes in the control corresponding to v: a red node v_r and a blue node v_b. All blue edges in E_b that were incident to v in the input to the exact matching problem are now incident to v_b, and all red edges that were incident to v are now incident to v_r. These edges corresponding to original edges in the input for the exact matching instance have zero distance, while all other distances are set to 1. The nodes in V_2 (of the original input graph to the exact matching) are the treatment nodes, so the distances we defined represent the distances between an treatment node and a control node.

The levels of the first covariate are defined such that every pair $[v_r, v_b]$ for $v \in V_1$ defines one level of the first covariate, and we have one treatment node in each such level. Observe that the number of levels of the first covariate is q, and we have q nodes in the treatment group, so this assignment of levels of the first covariate is feasible. Next, consider the second covariate. We will have two levels of the second covariate corresponding to blue and red. The red level of the control is the set of all red nodes, and the blue level of the control is the set of all blue nodes. The second level of the treatment are defined so that there will be exactly k nodes of the treatment in the blue level (and the remaining $q - k$ nodes in the treatment are in the red level). Now, we would like to apply algorithm ALG on the BM' instance and check if the output cost is zero or strictly positive. In any feasible solution of the BM' defined, exactly one of two control nodes $[v_r, v_b]$ is matched for each $v \in V_1$, as there is one treatment node in each level of the first covariate. And since we need to match k red control nodes and $q - k$ blue control nodes, a zero distance matching of the BM' represents a set of edges in the exact matching instance that is a perfect matching consisting of k blue edges and $q - k$ red edges.

Observe that this construction is a deterministic polynomial time algorithm. Thus the algorithm that constructs the input to BM' and apply ALG on that input is a deterministic (randomized) polynomial time algorithm for the exact matching problem if ALG is a deterministic (randomized, respectively) polynomial time algorithm for BM'.

We next prove the other direction. Assume that there is a polynomial time algorithm for the exact matching problem in bipartite graph, we will establish the existence of a polynomial time algorithm for BM'. Here we are going to use the fact that the maximum distance is at most π and without loss of generality, we assume that $q \leq \pi$. We set $\epsilon = 1/(q\pi + 1)$, and we consider the following multi-objective optimization problem. The goal is to find the

$(1+\epsilon)$-approximated Pareto set of matchings for the following multi-criteria objective.

The first objective is the total distance. The total distance is a non-negative integer of at most $q\pi$. Therefore, if we approximate this objective with an approximation ratio of $1+\epsilon$ then we get the optimal value of this objective (given the values of the other objectives). The other objectives of this multi-criteria problem are defined so that we have one objective per level of the second covariate and this objective is to minimize the number of selected control samples with this level of the second covariate. Similarly to the first objective, every possible value of this objective is an integer between 0 and q, and therefore if we approximate this objective with an approximation ratio of $1+\epsilon$ then we get the optimal value of this objective (given the values of the other objectives). Note that the sum of the last set of objectives (all objectives except to the first one) is always equal to q, and therefore if we have one of those objectives as super-optimal then another one is sub-optimal. Thus the $(1+\epsilon)$-approximated Pareto set need to include one solution per each possible vector of the number of elements in the different levels of the second covariate. So there are constant number of objectives and for our choice of ϵ the approximated Pareto set is in fact the Pareto set.

Now, we use the result of Papadimitriou and Yannakakis [43] to conclude that the algorithm for exact matching gives the required algorithm for approximating the Pareto set. In this Pareto set, we find the point corresponding to the level counters of the treatment, i.e., the point in the Pareto set corresponding to a feasible point for BM'.

Furthermore, if we use the existing algorithm of [39] to solve the exact matching then the resulting algorithm will be randomized polynomial time algorithm whereas if we will use it with (future) deterministic polynomial time algorithm then the resulting algorithm for BM' is also deterministic polynomial time algorithm. □

Next we consider the κ-BM' that is the special case of 2-covariate κ-BM where the second covariate has a constant number of levels, and once again we assume that the distance matrix is integral and the maximum distance is upper bounded by a polynomial π of the input encoding length. We show that κ-BM' has the same complexity status as BM' (for all $\kappa \geq 2$). That is, we establish the following result.

Theorem 3.11. *There is a polynomial time algorithm for BM' if and only if there is a polynomial time algorithm for κ-BM'.*

Proof. Assume that there is a polynomial time algorithm ALG for BM'. Consider an instance of the κ-BM' problem, and replace every treatment sample by κ copies of it with the same pair of levels as the original element of the treatment group. We define the distance matrix as follows. The distance between an treatment sample x that is a copy of the original treatment sample x' (of the instance for the κ-BM') and a control sample y, is now defined as the distance between x' and y. The resulting treatment group and control group is the instance for BM', and we apply ALG on that instance. A feasible solution for this BM' instance gives a feasible solution for the κ-BM' instance (simply by matching an treatment sample to a control sample if one of the copies of the treatment sample was matched to that control sample) and of the same total distance. Similarly, a feasible solution to the original

κ-BM' instance gives a feasible solution for the BM' instance of the same cost by matching the set of κ matched control sample to the unique treatment sample to the κ copies of this treatment sample in the BM' instance (with one control sample matched to each copy). Thus, we get a polynomial time algorithm for κ-BM' problem.

Consider the other direction. Assume that we are given a polynomial time algorithm ALG for κ-BM' and we establish the existence of a polynomial time algorithm for BM'. Consider an instance of the BM' problem. First, we add $(q+1)\cdot\pi$ for every component of the distance matrix. This modification of the distance matrix ensures that every feasible solution for BM' has a cost that is at least $(q^2+q)\pi$ and at most $(q^2+2q)\cdot\pi$. Next, for every treatment sample x we add another $\kappa-1$ control samples with the same levels as the ones of x whose distance from x is 0 and from other treatment samples the distance is $2q^3\cdot\pi$. These $\kappa-1$ control samples for each treatment sample are called dummy control samples. This is the instance of κ-BM' on which we apply ALG. The output has cost $B \le (q^2+2q)\cdot\pi$ if and only if the solution we obtain by deleting all the dummy control sample is a feasible solution for BM' of cost B. To prove the last claim note that the solution for the κ-BM' instance cannot match dummy control samples to treatment samples if their distance is not zero. Furthermore, the $\kappa-1$ zero distances for each treatment sample must be in the selected control samples as otherwise, the total distance would be at least $(q+1)^2\cdot\pi > (q^2+2q)\cdot\pi$. Thus, by deleting the dummy control sample we get a feasible solution for the BM' problem (after the modification of the distance matrix) of the same cost. Similarly, if we take an optimal solution for the BM' problem (before or after the modification of the distances) and add to it the dummy control samples that are matched using the zero-distances then we get an optimal solution for the κ-BM'. Therefore, by applying ALG on the last κ-BM' problem we get a polynomial time algorithm for solving the original BM' instance. □

3.6 The maximum selection κ-fine-balance matching (κ-MSBM) problem

Since any κ-BM problem can be solved as a κ-MSBM problem with the same κ and the same number of covariates, Corollary 3.3 implies that the P-covariate κ-MSBM problem is also NP-hard for $P \ge 2$ and any constant κ. And in Section 3.2 we show that the 1-covariate MSBM problem can be solved as an MCNF problem in polynomial time. We are going to show next that the 1-covariate κ-MSBM problem is NP-hard even for any constant $\kappa \ge 3$ by reduction from the Exact-3-cover problem (see Section 3.4) .

Theorem 3.12. *For any constant value of κ such that $\kappa \ge 3$, the 1-covariate κ-MSBM problem is NP-hard even with only one level.*

Proof. Given an instance of Exact-3-cover, namely a collection C of 3-element subsets (triplets) of a ground set E with $|E| = 3q$ for some integer q. We will define an instance of 1-covariate κ-MSBM with only one level for any constant κ such that $\kappa \ge 3$. In the constructed instance, we have one treatment sample for every triplet in C, and we have one control sample for every element $e \in E$. In addition we have $(\kappa - 3) \cdot q$ *dummy* control samples. For each triplet $T \in C$, the distance of the corresponding treatment sample to a control sample is defined as follows: it is zero if the control sample is one of the dummy samples or if the control sample is an element of the triplet T. All other distances (that are still undefined) are set to one. Observe that a feasible solution for the κ-MSBM instance selects all control group and selects exactly q treatment samples. We claim that in this instance, an optimal solution for κ-MSBM has total distance of zero, if and only if the Exact-3-cover instance is a YES instance.

To see the last claim assume first that there is a subcollection of triplets $C' \subseteq C$ such that every element of E appears in exactly one triplet in C'. Then we construct a zero-distance solution for the κ-MSBM instance as follows. We select the samples C' of the treatment group and each such selected sample $T \in C'$ is matched to the samples of the control consisting of the three elements samples in T together with $\kappa - 3$ of the dummy control samples. Since C' has q triplets, we have sufficient number of additional control samples. Furthermore, since every element in E appears only once in triplets of C', we conclude that its control sample is matched to exactly one selected treatment sample.

On the other hand, assume that there is a zero-distance solution for the κ-MSBM instance. Then, this solution selects exactly q treatment samples corresponding to q triplets. Denote by $C' \subseteq C$ this subcollection of q triplets. Note that if there is a selected treatment sample that is matched to at least $\kappa - 2$ dummy control samples, then there exists a selected treatment sample that is matched to at most $\kappa - 4$ dummy control samples, and thus it is matched to at least four control samples that correspond to elements of E, then at least one of those matches has distance one. This contradicts the assumption that the cost of the κ-MSBM solution is zero. Therefore, every selected treatment sample is matched to exactly $\kappa - 3$ additional control samples and three control samples that are corresponding to its elements. Since every control sample corresponding to an element is matched exactly once, we conclude that every element of E appears in exactly one triplet of C', so the Exact-3-cover instance is a YES instance. □

For the 1-covariate κ-MSBM problem where $\kappa = 2$, the complexity status remains open.

3.7 Fixed-parameter tractable algorithms

In this section, we consider the special cases of the min κ-imbalance, the κ-FBS, κ-BM, and κ-MSBM problems where all covariates have a small number of levels.

Let $K = \prod_{i=1}^{P} k_i$ be the number of level-intersections. Observe that if the number of covariates is constant and all covariates have constant number of levels, then K is a constant. We note that the problems κ-FBS, κ-BM, and MSBM can be solved in fixed-parameter tractable (FPT) time with parameter K. In order to state these results, we say that a problem is fixed-parameterized complexity with parameter K and denote it by

$FPT(K)$ if it has an algorithm whose time complexity is upper bounded by a function of the form $f(K)\cdot$ poly where $f(K)$ is some computable function of the parameter K, and poly is some polynomial of the input binary encoding length. We also say that an algorithm runs in $FPT(K)$ time and mean that its time complexity can be upper bounded by a function of the form $f(K)\cdot$poly where $f(K)$ is some computable function of the parameter K, and poly is some polynomial of the input binary encoding length. Here we show that these problems, namely κ-FBS, κ-BM problems for all κ, and MSBM problem are $FPT(K)$. Furthermore, we also show that similar results for κ-MSBM where $\kappa \geq 3$ cannot be obtained unless $P = NP$ as those special cases of κ-MSBM are NP-hard even if there is only one covariate that has only one level (i.e., for the case $K = 1$). The complexity status of the 2-MSBM problem with constant K is open.

Our proof for the $FPT(K)$ results uses the existence of fast algorithms for solving integer programming in fixed dimension and for solving mixed-integer linear programs if the number of integral variables is fixed. Lenstra [37] (see also [34] for an improved time complexity of these algorithms) showed that the integer linear programming problem with a fixed number of variables is polynomially solvable, and he also showed that a mixed-integer linear program with a fixed number of integer variables can be solved in polynomial time. In fact, these algorithms runs in FPT time with parameter being the number of integral variables. Therefore, to prove our results we show either an integer programming (IP) formulation with number of decision variables $O(K)$ or a mixed-integer linear program (MILP) with $O(K)$ integer variables such that solving this MILP to optimality ensures that the resulting solution is integral and solves the corresponding problem.

The min κ-imbalance problem

First consider the min κ-imbalance problem. For this problem we change the IP formulation (3.1) of Bennett et al. by replacing the decision variables by the level-intersections representation.

Let $u'_{i_1,i_2,...,i_P} = |L'_{1,i_1} \cap L'_{2,i_2} \cap ... \cap L'_{P,i_p}|$ for $i_p = 1,...,k_p,\ p = 1,...,P$. The level-intersections decision variables are:

$x'_{i_1,i_2,...,i_P}$: the number of control samples selected from the $(i_1, i_2, \ldots, i_P)$ level intersection $L'_{1,i_1} \cap L'_{2,i_2} \cap ... \cap L'_{P,i_p}$, for $i_p = 1,...,k_p,\ p = 1,...,P$.
The integer programming formulation is:

$$\min \quad \sum_{p=1}^{P}\sum_{i=1}^{k_p} y_{p,i} \tag{3.4a}$$

$$\text{s.t.} \quad \sum_{i_1=1}^{k_1}\cdots\sum_{i_{p-1}=1}^{k_{p-1}}\sum_{i_{p+1}=1}^{k_{p+1}}\cdots\sum_{i_P=1}^{k_P} x'_{i_1,i_2,\ldots,i_P} - \kappa\ell_{p,i} \le y_{p,i} \quad p=1,\ldots,P \;\; i_p=1,\ldots,k_p \tag{3.4b}$$

$$\kappa\ell_{p,i} - \sum_{i_1=1}^{k_1}\cdots\sum_{i_{p-1}=1}^{k_{p-1}}\sum_{i_{p+1}=1}^{k_{p+1}}\cdots\sum_{i_P=1}^{k_P} x'_{i_1,i_2,\ldots,i_P} \le y_{p,i} \quad p=1,\ldots,P \;\; i_p=1,\ldots,k_p \tag{3.4c}$$

$$\sum_{i_1=1}^{k_1}\cdots\sum_{i_P=1}^{k_P} x'_{i_1,i_2,\ldots,i_P} = \kappa n \tag{3.4d}$$

$$0 \le x'_{i_1,i_2,\ldots,i_P} \le u'_{i_1,i_2,\ldots,i_P} \quad p=1,\ldots,P, \;\; i_p=1,\ldots,k_p \tag{3.4e}$$

$$x'_{i_1,i_2,\ldots,i_P} \text{ integers} \quad p=1,\ldots,P, \;\; i_p=1,\ldots,k_p \tag{3.4f}$$

This integer programming formulation has $2K$ decision variables and $O(K)$ constraints, and thus the algorithm that constructs it and solves it to optimality runs in $FPT(K)$ time. The optimal solution for this integer program encodes the optimal solution for min κ-imbalance problem as shown in theorem 3.1.

The κ-FBS problem

For the κ-FBS problem we use an integer program with dimension $O(K)$ that is based on (IP-FBS) . Let $u_{i_1,i_2,\ldots,i_P} = |L_{1,i_1} \cap L_{2,i_2} \cap \ldots \cap L_{P,i_p}|$ and $u'_{i_1,i_2,\ldots,i_P} = |L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_p}|$ for $i_p = 1,\ldots,k_p,\; p = 1,\ldots,P$. The decision variables are:

$x_{i_1,i_2,\ldots,i_P}$: the number of treatment samples selected from the $(i_1, i_2, \ldots, i_P)$ level intersection $L_{1,i_1} \cap L_{2,i_2} \cap \ldots \cap L_{P,i_p}$, for $i_p = 1,\ldots,k_p,\; p = 1,\ldots,P$;

$x'_{i_1,i_2,\ldots,i_P}$: the number of control samples selected from the $(i_1, i_2, \ldots, i_P)$ level intersection $L'_{1,i_1} \cap L'_{2,i_2} \cap \ldots \cap L'_{P,i_p}$, for $i_p = 1,\ldots,k_p,\; p = 1,\ldots,P$.

The integer programming formulation is:

$$
\begin{aligned}
\max \quad & \sum_{i_1=1}^{k_1}\sum_{i_2=1}^{k_2}\cdots\sum_{i_P=1}^{k_p} x_{i_1,i_2,\dots,i_P} && (3.5a)\\
\text{s.t.} \quad & \kappa\cdot\sum_{i_1=1}^{k_1}\cdots\sum_{i_{p-1}=1}^{k_{p-1}}\sum_{i_{p+1}=1}^{k_{p+1}}\cdots\sum_{i_P=1}^{k_P} x_{i_1,i_2,\dots,i_P} = \sum_{i_1=1}^{k_1}\cdots\sum_{i_{p-1}=1}^{k_{p-1}}\sum_{i_{p+1}=1}^{k_{p+1}}\cdots\sum_{i_P=1}^{k_P} x'_{i_1,i_2,\dots,i_P} && \\
& \qquad p=1,\dots,P \quad i_p=1,\dots,k_p && (3.5b)\\
& 0\le x_{i_1,i_2,\dots,i_P}\le u_c \quad p=1,\dots,P,\quad i_p=1,\dots,k_p && (3.5c)\\
& 0\le x'_{i_1,i_2,\dots,i_P}\le u'_{i_1,i_2,\dots,i_P} \quad p=1,\dots,P,\quad i_p=1,\dots,k_p && (3.5d)\\
& x_{i_1,i_2,\dots,i_P},\, x'_{i_1,i_2,\dots,i_P} \text{ integers} \quad p=1,\dots,P,\quad i_p=1,\dots,k_p && (3.5e)
\end{aligned}
$$

.

Note that this integer programming formulation has $2K$ decision variables and $O(K)$ constraints, and thus the algorithm that constructs it and solves it to optimality runs in $FPT(K)$ time. The optimal solution for this integer program encodes the optimal solution for κ-FBS as shown in theorem 3.2.

The κ-BM problem

Next consider the κ-BM problem. In Section 3.2, we describe an MCNF formulation when the level intersection sizes $s'_{i_1,i_2,\dots,i_P}$ for $p=1,\dots,P$ and $i_p=1,\dots,k_p$ are given. Observe that if we treat the sizes $s'_{i_1,i_2,\dots,i_P}$ for all p and i_p as decision variables, then by enforcing the integrality of these K variables and adding the constraints saying that

$$\sum_{i_1=1}^{k_1}\cdots\sum_{i_{p-1}=1}^{k_{p-1}}\sum_{i_{p+1}=1}^{k_{p+1}}\cdots\sum_{i_P=1}^{k_P} s'_{i_1,i_2,\dots,i_P} = \kappa\cdot\ell_{p,i_p}, \quad i_p=1,\dots,k_p \quad p=1,\dots,P$$

forcing the κ-fine balance constraints to the MCNF formulation, we get a MILP formulation of κ-BM with K integral variables. In fact if we restrict ourselves to a common integral values of these K variables, then the other decision variables are integral as we argue next. By considering the values of these K integral variables as constants, the resulting linear programming formulation is in fact an MCNF LP formulation whose supply/demand vector depends on the values of these K integral variables. Thus, the optimal solution for the MILP is without loss of generality integral, and even if it does not satisfy this integral requirement it can be transformed to another optimal solution that is integral in polynomial time.

Since the number of variables of the resulting mixed-integer program is at most $n\cdot n'+K$, the number of integer variables is K, and the number of constraints is $O(n\cdot n')$, we conclude that the algorithm that formulates this MILP and solves it to optimality guaranteeing that the optimal solution is integral, runs in $FPT(K)$ time.

The MSBM and κ-MSBM problems

We know from Theorem 3.12 that the 1-covariate κ-MSBM problem for $\kappa \geq 3$ is NP-hard already if the unique covariate has only one level.

We consider next the MSBM problem. In Section 3.2, we describe an MCNF formulation if all the level intersection sizes $s_{i_1,i_2,\ldots,i_P}$ and $s'_{i_1,i_2,\ldots,i_P}$ are given. Observe that if we treat $s_{i_1,i_2,\ldots,i_P}$ and $s'_{i_1,i_2,\ldots,i_P}$ as decision variables, then by enforcing the integrality of these $O(K)$ variables and adding the constraints saying that

$$\sum_{i_1=1}^{k_1} \cdots \sum_{i_{p-1}=1}^{k_{p-1}} \sum_{i_{p+1}=1}^{k_{p+1}} \cdots \sum_{i_P=1}^{k_P} s_{i_1,i_2,\ldots,i_P} = \sum_{i_1=1}^{k_1} \cdots \sum_{i_{p-1}=1}^{k_{p-1}} \sum_{i_{p+1}=1}^{k_{p+1}} \cdots \sum_{i_P=1}^{k_P} s'_{i_1,i_2,\ldots,i_P}, \quad i_p = 1, \ldots, k_p \;\; p = 1, \ldots, P$$

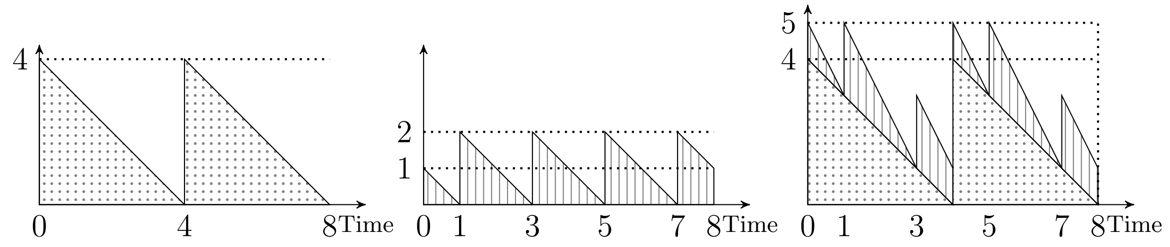
that is the fine balance constraints, in addition to the constraint saying that the sum over all $s_{i_1,i_2,\ldots,i_P}$ equals the objective function value of κ-FBS, to the MCNF formulation, we get a MILP formulation of κ-MSBM with $2K$ integral variables. In fact if we restrict ourselves to a common integral values of these $2K$ variables, then the other decision variables are without loss of generality integral as well as we argue next. By considering the values of these $2K$ integral variables as constants, the resulting linear programming formulation is in fact a MCNF LP formulation whose supply/demand vector depends on the values of these $2K$ integral variables. Thus, the optimal solution for the MILP is without loss of generality integral, and even if it does not satisfy this integral requirement it can be transformed to another optimal solution that is integral in polynomial time.

Since the number of variables of the resulting mixed-integer program is at most $n \cdot n' + 2K$, the number of integer variables is $2K$, and the number of constraints is $O(n \cdot n')$, we conclude that the algorithm that formulates this MILP and solves it to optimality guaranteeing that the optimal solution is integral, runs in $FPT(K)$ time.

Chapter 4

Integer Programming Formulation Inspired Approximation Schemes for the Replenishment Storage Problem

The Replenishment Storage problem (RSP) arises in planning a periodic replenishment schedule of multiple items so as to minimize the storage capacity required. The input to the RSP consists of a multi-item inventory system where each item has deterministic demand, a given reorder size and its own cycle length determined by its Economic Order Quantity. Here the reorders can only take place at an integer time unit within the cycle. The problem is to determine the timing of the first replenishment of each item within its cycle so that the maximum inventory level of all items over time is minimized.



(a) Inventory level of item 1 (b) Inventory level of item 2 (c) Total inventory level

Figure 4.1: A geometric view of RSP

Geometrically, RSP can be viewed as a problem of shifting periodic triangle functions and then packing them on top of each other so as to minimize the peak value required. To illustrate that, Figure 4.1a is the triangles function for the inventory level of item 1, for which the order quantity is 4 units and the cycle length is 4 periods. These correspond to the height and the width of each triangle respectively. The replenishment timing of item 1 is at time 0. A second item, item 2, is shown in Figure 4.1b. It has order quantity of 2 units and cycle length 2 and its replenishment timing is 1. With this selection of the replenishment timings the sum of the inventory levels of the two items is maximized at time 0 with a peak storage level is 5, illustrated in Figure 4.1c. This peak storage level is also the smallest possible for this RSP problem. We note that the peak storage level always coincides with the reorder timing of one of the items.

Chapter Overview

The different variants of the RSP is introduced in Section 4.1, together with a review on the related literature, and a summary of our result. Section 4.2 introduces notation, a review of the known integer programming formulation of RSP and the derivation of our new integer programming formulation. In Section 4.3 we prove the strong NP-hardness of RSP for non-constant value of k (joint cycle length) and the weak NP-hardness of RSP for constant value of k. These proofs apply for the complexity of both single-cycle RSP and multi-cycle RSP. The weak NP-hardness of RSP for constant k (joint cycle length) is proved here via a pseudo-polynomial time dynamic programming algorithm that solves the problem optimally. This establishes that both single-cycle RSP and multi-cycle RSP with constant joint cycle are weakly NP-hard. Then we first describe in Section 4.4 the new FPTAS for the multi-cycle RSP for constant k, and next in Section 4.5 we describe the fixed-parameter tractable FPTAS for the single-cycle RSP for constant k. Section 4.6 includes the new PTAS for the single-cycle RSP with non-constant k. That section also provides a description of the $(1+\frac{2}{k})$-approximation algorithm for single-cycle RSP of Hall [18]. We conclude with several remarks in Section 4.7.

4.1 Variants of the RSP

An instance of RSP consists of n items. Each item i is associated with an integer individual cycle length k_i, and an integer reorder size s_i . Here s_i is expressed in terms of the storage amount required for the reorder quantity. The *joint cycle length* of the n items is the least common multiple (lcm) of the lengths k_i, $i = 1, \dots, n$. We denote $k = \mathrm{lcm}(k_1, \dots, k_n)$. By the cyclical nature of the problem, the total inventory levels repeat periodically every k units of time for any reorder schedule. If all items have the same cycle time, k, the problem is said to be *single-cycle*, otherwise it is said to be *multi-cycle*.

The continuous-time variant of RSP, referred to as *continuous-RSP* here, allows the reorders (replenishment timing) to take place at any time within the cycle of each item. The continuous-RSP has been studied extensively. We clarify here the complexity status of the RSP and delineate the complexity gaps between it and the continuous variant of the problem.

The single-cycle RSP was shown by Hall [18] to be NP-hard, even for constant k. Since single-cycle RSP is a special case of multi-cycle RSP, it implies that multi-cycle RSP is NP-hard, even for constant joint cycle length k. Here we prove that for non-constant k, both single-cycle and multi-cycle RSP are strongly NP-hard. This matches the strongly NP-hardness of the continuous-RSP for the multi-cycle RSP with non-constant k, [13]. Prior to our results here it was conceivable that the RSP for multi-cycle and non-constant k could be easier than the continuous problem, if it were to be shown to be weakly NP-hard. With the strong NP-hardness result here both discrete and continuous problems are of equal complexity for this case.

For constant k, we provide here, for the first time, a pseudo-polynomial time algorithm that solves RSP optimally, proving that both single-cycle RSP and multi-cycle RSP for constant k are weakly NP-hard. Weakly NP-hard problems can have a Fully Polynomial Time Approximation Scheme (FPTAS). Indeed, we devise for the single-cycle RSP a FPTAS. That approximation scheme utilizes a newly introduced integer programming formulation of RSP. This new formulation is of independent interest and is shown to be tighter that the known integer program that has been used for RSP to date. For the strongly NP-hard single-cycle RSP with non-constant k we devise a Polynomial Time Approximation Scheme (PTAS), which is the best approximation possible for strongly NP-hard problems. Furthermore, for a fixed parameter ϵ, that PTAS delivers an ϵ-approximate solution in linear time.

Our results narrow the complexity gap between the RSP and the continuous variant for multi-cycle (with either constant or non-constant cycle length) and single-cycle with constant cycle length. The single-cycle continuous-RSP is polynomial time solvable [30]. Hence, our results widen the gap between single-cycle RSP and the single-cycle continuous variant with non-constant cycle length. For the multi-cycle case, and constant joint cycle length, the complexity status of continuous-RSP is open, whereas it is proved here that the RSP is weakly NP-hard. A tabular summary of the complexity results here is provided in Table 4.1 and 4.2.

Related literature

Single-cycle RSP was shown to be NP-hard for $k = 2$ [18], which implies that both single-cycle RSP and multi-cycle RSP are at least weakly NP-hard. Such proof however, does not exclude the possibility that the problems are strongly NP-hard, and therefore the complexity status of these two RSPs, as strongly NP-hard versus weakly NP-hard, has been unresolved until now. Hall [18] provided an approximation algorithm for the single-cycle RSP, with an approximation factor of $\left(1 + \frac{2}{k}\right)$. (Another algorithm provided by Hall in [18] delivers a 2-approximation, which is observed here to be satisfied by any feasible solution.) The $(1 + \frac{2}{k})$-approximation algorithm is of particular interest as it is a part of the PTAS we derive for the single-cycle RSP with non-constant k. That algorithm links the continuous and discrete problems by rounding (down) the fractional values of the closed form solution to the respective continuous problem. A complete description of this $(1 + \frac{2}{k})$-approximation algorithm is given in Section 4.6.

For multi-cycle RSP with only two items, Murthy, Benton, and Rubin [40] provided an optimal closed-form replenishment solution, meaning that it is solved in constant time.

Studies of multi-cycle RSP with more than two items have been focused on the development of heuristics. These include genetic algorithms [38, 60], a smoothing procedure utilizing a Boltzmann function [61], local-search procedures [10], a simulated-annealing algorithm [4] and a hybrid heuristic [4, 49]. None of these heuristics were shown to deliver a guaranteed approximation bound.

The continuous single-cycle RSP was first explicitly studied by Homer [30], who derived an optimal closed form solution. Later, Page and Paul [42], and Zoller [62] independently rediscovered Homer's result. Hartley and Thomas [20] considered the continuous-time multi-cycle RSP with only two items and devised an optimal closed-form solution. For multi-item multi-cycle continuous-RSP, the problem was proved to be strongly NP-complete for non-constant cycle lengths [13]. Anily [2], Hariga and Jackson [19] obtained lower bounds on the minimum peak storage required and proposed heuristics for multi-cycle continuous-RSP. Teo, Ou, and Tan [54] addressed multi-cycle continuous-RSP when for all pairs of individual cycle lengths, (k_i, k_j), the larger value is an integer multiple of the smaller value. For the problem with this integer-ratio cycles' lengths assumption, they devised a heuristic which is a $\frac{15}{8}$-approximation algorithm.

Continuous-RSP was used as a subproblem for the problem of identifying optimal cycle lengths and replenishment schedule for multi-items so as to minimize the order and inventory holding costs without exceeding a given storage capacity. In this latter problem each item is associated with a unit holding cost and an order cost. Under the assumption of identical cycles, an optimal closed form solution was derived from the closed form solution to the replenishment schedule for single-cycle continuous-RSP [30, 42, 62]. For this problem with non-identical cycles and only two items, a similar approach resulted in a closed form solution generated from the continuous-RSP closed form solution [20, 55].

A summary of the relevant known results for continuous and discrete RSP is given in Table 4.1 for non-constant k, and Table 4.2 for constant k. These tables include also our contributions here.

	Continuous-RSP	RSP (previous)	RSP (here)
2 items	Closed-form solution [20]	Closed-form [40]	
single-cycle	Closed-form solution [30, 42, 62]	$(1 + 2/k)$-approximation [18]	Strongly NP-hard & PTAS
multi-cycle	Strongly NP-hard [13]	NP-hard [18]	Strongly NP-hard

Table 4.1: Summary of complexity results for the RSP and a continuous-time variant with **non-constant** joint cycle length.

Summary of Contributions

Our main contributions on the RSP are:

1. Resolving the complexity status of RSP: We prove that for non-constant joint cycle length k the RSP is strongly NP-hard, whether single-cycle or multi-cycle. In contrast, for constant joint cycle length k we show that the problem is weakly NP-hard and provide a pseudo-polynomial time optimization algorithm with complexity that is a

		Continuous-RSP	RSP (previous)	RSP (here)
single-cycle	Complexity	Polynomial	NP-hard [18]	Weakly NP-hard
	Algorithm	Closed-form solution [30, 42, 62]	$(1+2/k)$-approximation [18]	Pseudo-poly optimization algorithm & FPTAS (fixed-parameter tractable)
multi-cycle	Complexity	Open	NP-hard [18]	Weakly NP-hard
	Algorithm			Pseudo-poly optimization algorithm & FPTAS

Table 4.2: Summary of complexity results for the RSP and a continuous-time variant with **constant** joint cycle length.

polynomial function of the sum of the order sizes s_i and is exponential in the joint cycle length k. These results demonstrate that for non-constant k both the RSP and the continuous-RSP are strongly NP-hard, and in that closes the gap between these two problems. This provides extra evidence to support the conjecture that the RSP can only be harder than the continuous variant. For constant k the continuous-RSP is polynomial for the single-cycle case and of unknown complexity for the multi-cycle case. From our results the problem is weakly NP-hard for both the single and multi-cycle cases which, if our conjecture is true, implies that the continuous-RSP for the multi-cycle case cannot be strongly NP-hard.

2. Identifying a new innovative IP formulation for RSP. This formulation enables the derivation of the three approximation schemes listed below for the RSP.

3. Devising the first known FPTAS for, the weakly NP-hard, multi-cycle RSP with constant k.

4. Devising the first known fixed-parameter tractable FPTAS for, the weakly NP-hard, single-cycle RSP in terms of parameter k.

5. Devising the first known polynomial time approximation scheme (PTAS) for, the strongly NP-hard, single-cycle RSP with non-constant k.

A summary of our results for RSP as compared to the best previously known results is given in the two tables: Table 4.1 for RSP with non-constant joint cycle length k, and Table 4.2 for RSP with constant joint cycle length k.

4.2 Preliminaries and Integer Programming Formulations

Given an instance of RSP, the demand rates and inventory levels are given in terms of the respective reorder size: for item i, the demand per unit of time is $\frac{s_i}{k_i}$, and its inventory levels at each replenishment cycle of k_i time units starting at time T, $(T+0, T+1, \ldots, T+k_i-1)$, are $(s_i, \frac{k_i-1}{k_i}s_i, \frac{k_i-2}{k_i}s_i, \dots, \frac{1}{k_i}s_i)$. The inventory level of item i on time ℓ, given that it is re-ordered on time j, is denoted by $V_{ij\ell}$. Thus $V_{ij\ell} = s_i \cdot \left(1 - \frac{(\ell-j) \bmod k_i}{k_i}\right)$.

Recall that since $k = \text{lcm}(k_1, \dots, k_n)$, the inventory levels are periodic within a cycle of k time units (repeat every k time units). It is therefore sufficient to determine the peak storage requirement by examining a time interval of length k. In this interval, we only need to look at discrete time values from 1 to k for that peak storage always coincides with the reorder timing of an item. Note that inventory level at time 0 is the same as inventory level at time k.

The decision variables in the integer programming formulations are the assignments of time periods within the k-unit time frame to the orders of all items. This assignment of timing is given as an $n \times k$ binary matrix $\mathbf{x}$ where

$$x_{ij} = \begin{cases} 1 & \text{if item } i \text{ is ordered at time j,} \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4.1. *A $n \times k$ binary matrix $\mathbf{x}$ is said to be a* valid assignment *for a given instance if and only if each item i is replenished exactly once every k_i time units. That is,*

$$\textstyle\sum_{j=1}^{k_i} x_{ij} = 1 \qquad i = 1, \dots, n, \quad \textit{and} \quad x_{ij} = x_{i,(j-k_i)} \quad i = 1, \dots, n, \quad j = k_i + 1, \dots, k.$$

The following listed notations denote demand rates, inventory levels, the total sum of reorder sizes at an integer time and the optimal peak storage:
$d_i = \frac{s_i}{k_i}$: demand rate of item i for $i = 1, ..., n$.
$D = \sum_{i=1}^n d_i = \sum_{i=1}^n \frac{s_i}{k_i}$: total demand (aggregate stock depletion) per unit of time.
$V_{ij\ell} = s_i \cdot \left(1 - \frac{(\ell-j) \bmod k_i}{k_i}\right)$: the inventory level of item i at time ℓ given that the reorder time is j.
$V_\ell(\mathbf{x}) = \sum_{i=1}^n \sum_{j=1}^{k_i} V_{ij\ell} x_{ij}$: the inventory level at time ℓ for $\ell = 1, ..., k$.
$V(\mathbf{x}) = \max_{\ell \in \{1,..,k\}} V_\ell(\mathbf{x})$: the maximum inventory level (peak storage) of a cycle.
$Q_j(\mathbf{x}) = \sum_{i=1}^n s_i x_{ij}$: the total sum of reorder sizes at time j for $j = 1, ..., k$.
$V^* = \min_{\mathbf{x} \text{ valid}} V(\mathbf{x})$: the optimal peak inventory level.
Let the following quantity, which is a constant, be denoted by C: $C = \sum_{i=1}^n \frac{1}{2}(1 + \frac{1}{k_i}) k s_i + \frac{(1+k)k}{2} D$. This quantity is used in deriving the new formulation and the FPTAS.

It is observed next that the peak inventory level for any valid assignment $\mathbf{x}$ is within twice the optimum. Hence, any replenishment schedule is a 2-approximate solution:

Lemma 4.1. *Any valid assignment $\mathbf{x}$ is a 2-approximate solution.*

Proof. At time ℓ, for $\ell = 1, \dots, k$, the inventory level of all items is $V_\ell(\mathbf{x}) = \sum_{i=1}^n \sum_{j=1}^{k_i} V_{ij\ell} x_{ij}$. Since $V_{ij\ell} = s_i \cdot \left(1 - \frac{(\ell-j) \bmod k_i}{k_i}\right) \le s_i$,

$$V_\ell(\mathbf{x}) \le \sum_{i=1}^n \sum_{j=1}^{k_i} s_i x_{ij} = \sum_{i=1}^n \left(s_i \sum_{j=1}^{k_i} x_{ij} \right) = \sum_{i=1}^n s_i.$$

Hence the storage capacity needed is less than or equal to $\sum_{i=1}^{n} s_i$.

On the other hand, the average inventory level of item i per time unit is $\frac{1}{2}(1+\frac{1}{k_i})s_i$ for any i. So the aggregated average inventory level of all items per time unit is $\frac{1}{k}\sum_{\ell=1}^{k} V_\ell(\mathbf{x}) = \sum_{i=1}^{n}\frac{1}{2}(1+\frac{1}{k_i})s_i > \sum_{i=1}^{n}\frac{1}{2}s_i$. The peak inventory level is at least as much as this aggregated average inventory level, thus, $V^* \geq \frac{1}{2}\sum_{i=1}^{n} s_i$.

Consequently, $V(\mathbf{x}) \leq \sum_{i=1}^{n} s_i \leq 2V^*$, which means that $\mathbf{x}$ is a 2-approximate solution.

□

The standard integer programming formulation:

We present next a straightforward formulation of RSP was used in previous studies [40, 4, 49]. Let the binary variables y_{ij} be,

$$y_{ij} = \begin{cases} 1 & \text{if item } i \text{ is ordered at time j} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, ..., n, \quad j = 1, ..., k_i.$$

Note that variables y_{ij} coincide with variables x_{ij}, that are defined for k time periods, within the first k_i time periods of a single cycle. In that sense the Y_i variables are a projection of the x_{ij} variables on the first k_i periods.

Let $V_{ij\ell}$ be parameters defined as above, $V_{ij\ell} = s_i\left(1 - \frac{(\ell-j) \bmod k_i}{k_i}\right)$. The standard integer programming formulation (RSP$_0$) is as follows:

$$\begin{array}{lll} (\text{RSP}_0) & \min & V \\ \text{subject to} & & \sum_{i=1}^{n}\sum_{j=1}^{k_i} V_{ij\ell}y_{ij} \leq V \quad \ell = 1, \ldots, k \\ & & \sum_{j=1}^{k_i} y_{ij} = 1 \quad\quad i = 1, \ldots, n \\ & & y_{ij} \text{ binary for } i = 1, ..., n, \quad j = 1, .., k_i. \end{array}$$

A New Integer Programming Formulation

We observe that formulation (RSP$_0$) has multiple solutions of the same value. Specifically, for any valid assignment that attains the peak storage at day ℓ there are $k-1$ other valid assignments of the same objective value that attain the maximum storage on any day other than ℓ. For example, to attain the peak storage at day $\ell+1$ we shift the reorder by one day forward. Our new formulation requires that any feasible assignment attains its peak storage at time k (or equivalently, at time 0). This is proved to be possible with "shift"-permutations in Lemma 4.2. Another aspect of the new formulation is that instead of dealing with inventory levels directly, as in (RSP$_0$), the new formulation uses, as the main variables, the total reorder size at time j, $Q_j(\mathbf{x}) = \sum_{i=1}^{n} s_i x_{ij}$ for $j = 1, ..., k$. Lemma 4.3 and Lemma 4.4 establish the relationship between inventory levels and reorder sizes.

Lemma 4.2. *For any valid assignment* $\mathbf{x}$ *there is a shift-permutation of* $1, \ldots, k$, *denoted by* $\pi(1), \ldots, \pi(k)$, *such that the valid assignment* $\mathbf{x}'$ *with* $x'_{ij} = x_{i\pi(j)}$, *attains peak inventory level at time k, and this new peak inventory level equals the peak inventory level of assignment* $\mathbf{x}$. *That is,* $V_k(\mathbf{x}') = V(\mathbf{x}') = V(\mathbf{x})$.

Proof. Suppose the peak storage for assignment $\mathbf{x}$ is attained at time h, $V_h(\mathbf{x}) = V(\mathbf{x})$.

For the following shift-permutation, $\pi(j) = \begin{cases} (j+h), & \text{if } j+h \le k \\ (j+h)-k, & \text{if } j+h > k \end{cases}$, the assignment $\mathbf{x}'$ with $x'_{ij} = x_{i\pi(j)}$ is a new valid assignment which is h time units shifted back in time as compared to $\mathbf{x}$. In other words, the inventory levels induced by the new assignment $\mathbf{x}'$ form a shift permutation of the original sequence of inventory levels, $V_j(\mathbf{x}') = V_{\pi(j)}(\mathbf{x})$. Therefore the peak inventory levels of the two assignments are the same, and $V_k(\mathbf{x}') = V_{\pi(k)}(\mathbf{x}) = V_h(\mathbf{x}) = V(\mathbf{x})$.

□

With Lemma 4.2, we can restrict valid assignments to those attaining peak inventory level at time k without changing the optimal solution of RSP. Hence, RSP can also be formulated as minimizing the inventory level at time k such that the schedule is a valid assignment that attains peak inventory level at time k, which can be written as $V_\ell(\mathbf{x}) \le V_k(\mathbf{x})$ for $\ell = 1, ..., k$.

Next we show, in Lemma 4.3, that for any valid assignment $\mathbf{x}$, the inventory level of time ℓ can be determined based only on $Q_j(\mathbf{x})$, the total amount ordered at time j, the inventory level at time k, $V_k(\mathbf{x})$, and on D, the total sum of demand rates of all items:

Lemma 4.3. *For any valid assignment* $\mathbf{x}$*,*

$$V_\ell(\mathbf{x}) = V_k(\mathbf{x}) - \ell D + \sum_{j=1}^{\ell} Q_j(\mathbf{x}), \quad \ell = 1, .., k \tag{4.1}$$

Proof. Each unit of time, item i's inventory is reduced by its demand rate d_i, and the total inventory level is reduced by $D = \sum_{i=1}^n d_i$. Since $Q_j(\mathbf{x}) = \sum_{i=1}^n s_i x_{ij}$ is the reorder size at time j, the inventory level at time j is $V_j(\mathbf{x}) = V_{(j-1) \bmod k}(\mathbf{x}) - D + Q_j(\mathbf{x})$. Note that inventory level at time 0 is the same as inventory level at time k by the cyclical nature of RSP. For any integer time ℓ, we apply this equation recursively with $j = \ell, \ell-1, ..., 1$ to derive that, $V_\ell(\mathbf{x}) = V_k(\mathbf{x}) - \ell D + \sum_{j=1}^{\ell} Q_j(\mathbf{x})$.

□

Lemma 4.2 shows that we can formulate RSP as minimizing the inventory level at time k such that the schedule is a valid assignment. A consequence of this lemma is that inequalities $V_\ell(\mathbf{x}) \le V_k(\mathbf{x})$ for $\ell = 1, ..., k$ can be rewritten using equations (4.1) in Lemma 4.3 as:

$$V_k(\mathbf{x}) \ge V_k(\mathbf{x}) - \ell D + \textstyle\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \text{ for } \ell = 1, \ldots, k$$

or equivalently,

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D \text{ for } \ell = 1, .., k \tag{4.2}$$

We refer to this set of inequalities (4.2) as the *cascading constraints.* These constraints enforce the peak storage at time k.

Next we address the objective function. Let $z(\mathbf{x})$ be the following function of a valid assignment $\mathbf{x}$:

$$z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} s_i x_{ij}.$$

In the next lemma we prove that minimizing the inventory level of time k, $V_k(\mathbf{x})$, is equivalent to maximizing $z(\mathbf{x})$. This is proved by showing that the sum of $kV_k(\mathbf{x})$ and $z(\mathbf{x})$ is a constant, the constant C defined earlier.

Lemma 4.4. *$kV_k(\mathbf{x}) + z(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{2}(1+\frac{1}{k_i})ks_i + \frac{(1+k)k}{2}D$.*

Proof. For item i, the sum of storage space it takes up during its k_i time units of reorder cycle is $\sum_{j=1}^{k_i} \frac{k_i+1-j}{k_i}s_i = \frac{1}{2}(1+\frac{1}{k_i})k_i s_i$. There are $\frac{k}{k_i}$ reorder cycles for item i during the time frame of length k so item i takes a total of $\frac{1}{2}(1+\frac{1}{k_i})ks_i$ units storage space. The sum of all items' inventory levels over the k integer times is hence $\sum_{\ell=1}^{k} V_\ell(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{2}(1+\frac{1}{k_i})ks_i$. Using formula (4.1) in the statement of Lemma 4.3, $\sum_{\ell=1}^{k} V_\ell(\mathbf{x})$ can be re-written as:

$$\begin{aligned}\sum_{\ell=1}^{k} V_\ell(\mathbf{x}) &= \sum_{\ell=1}^{k}\left(V_k(\mathbf{x}) - \ell D + \sum_{j=1}^{\ell} Q_j(\mathbf{x})\right)\\ &= kV_k(\mathbf{x}) - \frac{(1+k)k}{2}D + \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x})\\ &= kV_k(\mathbf{x}) - \frac{(1+k)k}{2}D + z(\mathbf{x}).\end{aligned}$$

It follows that,

$$kV_k(\mathbf{x}) + z(\mathbf{x}) = \sum_{\ell=1}^{k} V_\ell(\mathbf{x}) + \frac{(1+k)k}{2}D = \sum_{i=1}^{n} \tfrac{1}{2}(1+\frac{1}{k_i})ks_i + \frac{(1+k)k}{2}D,$$

and the right hand side is a constant for every problem instance.

□

From Lemma 4.4 it follows that minimizing $V(\mathbf{x})$ can be replaced by maximizing $z(\mathbf{x})$. This, with the cascading constraints lead to the new integer programming formulation (RSP). For presentation simplicity we use $Q_j(\mathbf{x}) = \sum_{i=1}^{n} s_i x_{ij}$:

$$\begin{array}{lll}\text{(RSP)} & \max & z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x})\\ \text{subject to} & & \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D \quad \ell = 1,..,k\\ & & \sum_{j=1}^{k_i} x_{ij} = 1 \qquad i = 1,\dots,n\\ & & x_{ij} = x_{i,(j-k_i)} \quad i = 1,\dots,n, \quad j = k_i+1,\dots,k\\ & & x_{ij} \text{ binary for } i = 1,...,n, \quad j = 1,..,k_i.\end{array}$$

A Graphical Visualization of (RSP)

To provide the intuition behind the cascading constraints and objective function $z(\mathbf{x})$ of (RSP) we present a graphical illustration in Fig 4.2. In the example illustrated we let the cycle length be $k = 4$. In Figure 4.2a, there are $k = 4$ columns with heights ℓD for time $\ell = 1, ..., k$, which represent the right hand sides of the cascading constraints.

In Figure 4.2b, there is a rectangle with height $Q_1(\mathbf{x})$ that extends from day 1 to day k. A second rectangle with height $Q_2(\mathbf{x})$ is stacked on the first rectangle, extending from day 2 to day k. The third rectangle of height $Q_3(\mathbf{x})$ extends from day 3 to day k, and the last, fourth, rectangle of height $Q_4(\mathbf{x})$ extends only over time period (day) $k = 4$. With this construction, the sum of heights of all rectangles within the column corresponding to time ℓ is exactly the left hand side of the ℓth cascading constraint, $\sum_{j=1}^{\ell} Q_j(\mathbf{x})$.

The area of the first rectangle is $kQ_1(\mathbf{x})$, the second is $(k-1)Q_2(\mathbf{x})$, and in general, the jth rectangle covers an area of $(k+1-j)Q_j(\mathbf{x})$. Therefore, the sum of areas of all rectangles is $z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x})$, which is the objective function of (RSP).

Therefore, geometrically, (RSP) seeks to find the valid assignment $\mathbf{x}$ that maximizes the area of the rectangles representing $Q_j(\mathbf{x})$'s, subject to the constraints that the total height of the rectangles in column ℓ does not exceed the column height ℓD.

We call the area *not* covered in each column, the remainder. The remainders are the spaces between the column heights in Figure 4.2 and the packed rectangles of area $z(\mathbf{x})$. Obviously minimizing those is identical to maximizing $z(\mathbf{x})$, since the total area of the remainders plus the area covered by the rectangles $z(\mathbf{x})$ is a constant, $\sum_{\ell=1}^{k} \ell D = \frac{(k+1)kD}{2}$. It is important to observe that the area of the remainders is not fully available for additional items. That is because if an item j is added to the reorder in day $p(j)$, its size s_j will diminish not only the remainder on day $p(j)$ but also on days $p(j)+1, \ldots, k$. Instead, the critical form of remainders are the *adjusted remainders* defined in Section 4.5, which are added to the objective function $z(\mathbf{x})$ for the derivation of the FPTAS for the single-cycle RSP.

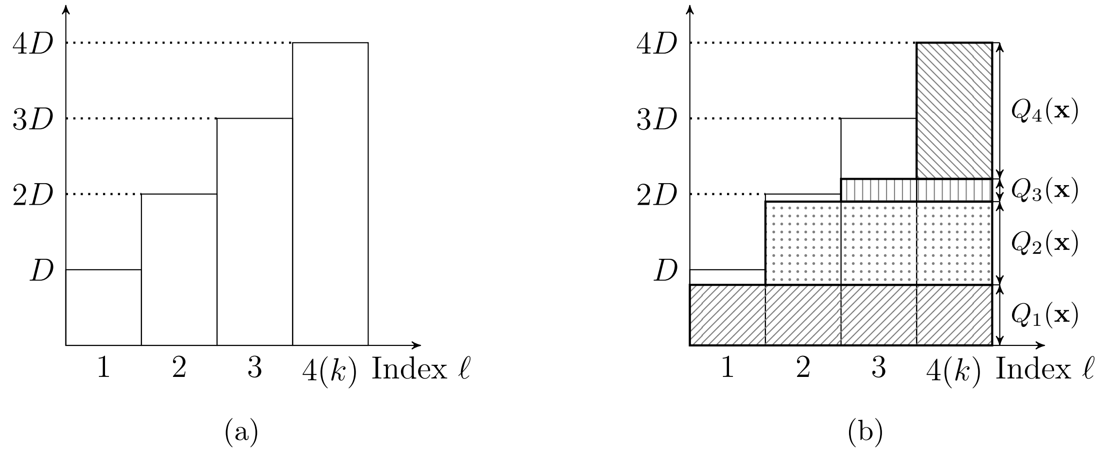


Figure 4.2: A graphical visualization of formulation (RSP).

Comparing the New and Standard Formulations, (RSP) and (RSP$_0$)

As noted before, (RSP) removes multiple optima that differ in the ordering present in (RSP$_0$), and always selects an optimal solution with peak storage attained at time k. In that sense any solution to (RSP) and its LP-relaxation is contained in the set of feasible solutions to (RSP$_0$) or its LP-relaxation. Yet there are other features on which the two problems differ. The new presentation of the problem as a maximization problem opens up the option, when there are multiple solutions to (RSP), of selecting the solution that minimizes the "adjusted remainders", a concept described in Section 4.5, which is crucial in the derivation of the FPTAS and the PTAS for the single-cycle RSP. A key result shown in Section 4.5 is that the optimal solution to the new integer programming formulation when the sum of the adjusted remainders is added to the objective, is "close" to the optimal solution.

Another difference between the two formulations is the form of the constraints' matrix coefficients. The coefficients of binary variables in the constraints of (RSP$_0$) vary for each i, j, ℓ, as the coefficient $V_{ij\ell}$ follows the triangular line. By contrast, in the cascading constraints of (RSP) the coefficients of binary variable x_{ij} can only be 0 or s_i for each $i, j\ell$ as they represent the packing of rectangles. For single-cycle RSP, the coefficients of x_{ij} in (RSP) are all 0 in the first $j-1$ cascading constraint and all s_i for the remaining ones. We believe that as a result, the LP-relaxation of (RSP$_0$) is likely to have more fractional variables in an optimal solution than the LP-relaxation of (RSP)

4.3 The Complexity of RSP for Constant and Non-constant Joint Cycle Length

RSP has been known to be NP-hard for both the single-cycle and multi-cycle versions and for either constant or non-constant joint cycle length k [18]. We show in this section that for constant k both the single-cycle RSP and multi-cycle RSP are actually weakly NP-hard. We derive for both problems a pseudo-polynomial time algorithm that solves these problems optimally. For non-constant k we show here that both the single-cycle RSP and multi-cycle RSP are strongly NP-hard (and hence there cannot be a pseudo-polynomial time algorithm for non-constant k unless NP=P). We first prove the strong NP-hardness for non-constant k.

Non-constant Joint Cycle RSP is Strongly NP-hard

The single-cycle RSP is a special case of multi-cycle RSP. It is therefore sufficient to prove that the non-constant joint cycle length, single-cycle RSP is strongly NP-hard, as that would imply that the multi-cycle RSP is also strongly NP-hard for k non-constant. The reduction is from the *3-Partition* problem, a well-known strongly NP-hard problem [14].
3-Partition: Given integers $a_1, a_2, \ldots, a_{3m}$, and integer b such that $\frac{1}{4}b < a_i < \frac{1}{2}b$ for each i

and $\sum_{i=1}^{3m} a_i = mb$. Can $\{1, 2, \ldots, 3m\}$ be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for $1 \leq j \leq m$, $\sum_{i \in A_j} a_i = b$?

Theorem 4.1. *The single-cycle RSP with non-constant joint cycle length is strongly NP-hard.*

Proof. Given an instance of *3-Partition*, we define an instance of single-cycle RSP with $n = 3m$, $k = m$, and $s_i = a_i$ for $1 \leq i \leq 3m$. The decision problem for RSP is stated as: Is there a valid assignment $\mathbf{x}$ with peak inventory level that is less than or equal to $V = \frac{1+m}{2m} \sum_{i=1}^{3m} a_i$?

We observe that the sum of inventory levels at integer times over a cycle equals $\sum_{i=1}^{3m}(1 + \frac{m-1}{m} + \ldots + \frac{1}{m})a_i = \sum_{i=1}^{3m} \frac{m+1}{2} a_i = mV$. Having $\sum_{\ell=1}^{k} V_\ell(\mathbf{x}) = mV$, the peak inventory level $\max_\ell V_\ell(\mathbf{x}) \leq V$ is equivalent to $V_\ell(\mathbf{x}) = V$ for all ℓ. This means that the inventory levels at all integer time units are the same. From Lemma 4.3 it follows that $V_1(\mathbf{x}) = V_2(\mathbf{x}) = \ldots = V_m(\mathbf{x})$ if and only if $Q_1(\mathbf{x}) = Q_2(\mathbf{x}) = \ldots = Q_m(\mathbf{x}) = D$, where $D = \sum_{i=1}^{3m} \frac{a_i}{m} = b$. Let $A_j = \{i | x_{ij} = 1\}, j = 1, \ldots, m$, then $Q_j(\mathbf{x}) = \sum_{i=1}^{n} a_i x_{ij} = \sum_{i \in A_j} a_i$. Therefore the decision problem corresponding to single-cycle RSP has a 'yes' answer if and only if there is a partition $A_j, j = 1, \ldots, m$, such that $\sum_{i \in A_j} a_i = b$ for $1 \leq j \leq m$.

□

Constant Joint Cycle Length RSP is Weakly NP-hard

We prove the weak NP-hardness of the RSP with constant k by devising a dynamic programming algorithm solving RSP optimally in pseudo-polynomial time. The complexity of this dynamic programming algorithm depends on the value of the order sizes (rather than the length/logarithm of these sizes) and is exponential in k. Such complexity is considered pseudo-polynomial for constant k. Since both the single-cycle RSP and the multi-cycle RSP with constant k are NP-hard [18], the existence of such an algorithm implies that the RSP with constant joint cycle length k is weakly NP-hard for both single-cycle and multi-cycle.

The dynamic programming algorithm presented in this paper is associated with the integer programming formulation (RSP). It is also possible to devise a dynamic programming algorithm of the same complexity based on the standard formulation (RSP$_0$). We comment however that in order to generate the FPTAS for the single-cycle RSP, presented later, it is essential to use the dynamic programming procedure that is based on (RSP).

For h an integer such that $0 \leq h \leq n$, let $\mathbf{x}^h$ denote the assignment of reorders for the first h items. Let the function $f_h(q_1, q_2, ..., q_k)$ be the maximum of $z(\mathbf{x}^h)$ with the cumulative reorder sizes at time ℓ being restricted to less than or equal to q_ℓ for $\ell = 1, ..., k$. Here, $(q_1, \ldots, q_k)$ is an integer array with $q_\ell \in [0, \ell D]$. Formally,

$$\begin{aligned} f_h(q_1,q_2,...,q_k) = \quad & \max & & \textstyle\sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^h) \\ & \text{subject to} & & \textstyle\sum_{j=1}^{\ell} Q_j(\mathbf{x}^h) \le q_\ell \quad \ell = 1,..,k \\ & & & \textstyle\sum_{j=1}^{k_i} x_{ij} = 1 \qquad i = 1,\dots,h \\ & & & x_{ij} = x_{i(j-k_i)} \quad i = 1,\dots,h, \quad j = k_i+1,\dots,k \\ & & & x_{ij} \text{ binary for } i = 1,...,h, \quad j = 1,..,k_i, \end{aligned}$$

where $Q_j(\mathbf{x}^h) = \sum_{i=1}^{h} s_i x_{ij}$. We set $f_h(q_1,q_2,...,q_k) = -\infty$ if the above integer programming problem is infeasible. The optimal solution being sought is $f_n(D,2D,...,kD)$.

The values of the function $f_h(q_1,q_2,...,q_k)$ are evaluated for every $h = 1,...,n$ and any integer array $(q_1,\dots,q_k)$, where $q_j \in [0,jD]$, with a dynamic programming recursion. The boundary conditions are $f_0(q_1,q_2,\dots,q_k) = 0$ for any $(q_1,q_2,\dots,q_k)$. The recursive derivation of $f_h(q_1,q_2,\dots,q_k)$ from $f_{h-1}(\cdot)$ requires to determine the timing to replenish item h within the first k_h time units so as to maximize the objective $\sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^h)$.

To see how the recursion works we split the objective function into the terms involving item h and the terms involving items $1,\dots,h-1$:

$$\sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^h) = \sum_{j=1}^{k}(k-j+1)s_h x_{hj} + \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^{(h-1)}) \tag{4.3}$$

If time τ, for $1 \le \tau \le k_h$, is selected to be the first reorder timing of item h, the following reorder timings of item h are $\tau + k_h, \tau + 2k_h, ..., \tau + (\frac{k}{k_h} - 1)k_h$. The first part of (4.3) is then

$$\sum_{j=1}^{k}(k-j+1)s_h x_{hj} = \sum_{t=0}^{k/k_h-1}(k+1-\tau-tk_h)s_h = \left(\frac{k+k_h}{2} + 1 - \tau\right)\frac{k}{k_h}s_h.$$

Let $q'_\ell(\tau) = q_\ell - \sum_{j=1}^{\ell} s_h x_{hj} = q_\ell - \lfloor\frac{\ell-\tau+k_h}{k_h}\rfloor s_h$, the second term in (4.3), $\sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^{(h-1)})$, has a maximum of $f_{h-1}(q'_1(\tau),...,q'_k(\tau))$ if $q'_\ell(\tau) \ge 0$ for all ℓ, and minus infinity otherwise.

The recursive equation, using the notation $q'_\ell(\tau) = q_\ell - \lfloor\frac{\ell-\tau+k_h}{k_h}\rfloor s_h$, is:

$$f_h(q_1,q_2,...,q_k) = \begin{cases} \max_{\tau=1,...,k_h}\{\left(\frac{k+k_h}{2} + 1 - \tau\right)\frac{k}{k_h}s_h + f_{h-1}(q'_1(\tau),...,q'_k(\tau))\}, & \text{if } q'_\ell(\tau) \ge 0 \text{ for all } \ell \\ -\infty & \text{otherwise.} \end{cases}$$

All function values are evaluated recursively for $h = 1,...,n$ and for all integer values of $(q_1,\dots,q_k)$, where each $q_j \in [0,jD]$ and q_j integer. Each function evaluation is associated with a choice of $\tau(h)$, which is the timing of the replenishment of item h within the k_h cycle. The optimal objective value is then $f_n(D,2D,...,kD)$.

To recover the optimal valid assignment we record the choices of the replenishment timings within the k cycle, for each function value evaluation.

Since there are $O(n)$ possible values of h and $O(\ell D)$ possible values of q_ℓ for each $1 \leq \ell \leq k$, the total number of function evaluations is $O(n \cdot 1D \cdot 2D \cdot \ldots \cdot kD) = O(n \cdot k!D^k)$. Each evaluation for item h enumerates the $O(k_h)$ choices of τ, and for each choice τ, it takes $O(k)$ to compute $q'_1(\tau), ..., q'_k(\tau)$. Therefore, the run time of this dynamic programming procedure is $O(k^2 \cdot k!D^k \cdot n)$, which is $O(nD^k)$ for constant k. This complexity is pseudo-polynomial and hence RSP is weakly NP-hard for constant k. This weak NP-hardness applies for both the single-cycle RSP and the multi-cycle RSP problems.

4.4 A fully polynomial-time approximation scheme for the multi-cycle RSP with constant joint cycle length

In this section, we establish the first known FPTAS for the multi-cycle RSP for constant joint cycle length. We derive a family of $(1+\epsilon')$-approximation algorithms for the multi-cycle RSP for every $\epsilon' > 0$. The $(1+\epsilon')$-approximation algorithm works by applying the dynamic programming algorithm in Section 4.3 with scaled reorder sizes with scaling factor T. We show in this section that the output of the dynamic programming algorithm using the scaled sizes is within a factor of $1+\epsilon'$ of the optimal solution. The run time of this approximation algorithm is polynomial in n and $\frac{1}{\epsilon'}$, and hence this family of algorithms is a fully polynomial approximation scheme.

The $(1+\epsilon')$-approximation solves a modified RSP, (scaled-RSP), in which the order sizes are scaled by a factor T. The scaled problem is solvable using the dynamic programming procedure of Section 4.3 and the solution of it is a valid assignment that has objective function value close to the optimal value of (RSP).

The scaling of (RSP), (scaled-RSP)

For any $\epsilon' > 0$, we let $\epsilon = \epsilon'/2$ and we scale the reorder sizes by the factor $T = \frac{\epsilon D}{kn}$ as follows. Let $s'_i = \lfloor \frac{s_i}{T} \rfloor$ be the scaled sizes of items $i = 1, ..., n$ and $D' = \frac{D}{T}$ be the scaled demand. Let $Q'_j(\mathbf{x})$ and $z'(\mathbf{x})$ denote the "scaled" replenishment sizes at time j and the objective function for the scaled sizes s'_i: $Q'_j(\mathbf{x}) = \sum_{i=1}^n s'_i x_{ij}, \quad j = 1, .., k$; $z'(\mathbf{x}) = \sum_{j=1}^k (k-j+1) Q'_j(\mathbf{x})$.

The scaled problem (scaled-RSP) is formulated as follows:

$$\begin{array}{lll}
\text{(scaled-RSP)} & \max & z'(\mathbf{x}) = \sum_{j=1}^k (k-j+1) Q'_j(\mathbf{x}) \\
\text{subject to} & & \sum_{j=1}^\ell Q'_j(\mathbf{x}) \leq \ell D' \quad \ell = 1, .., k \\
& & \sum_{j=1}^{k_i} x_{ij} = 1 \qquad i = 1, \ldots, n \\
& & x_{ij} = x_{i,(j-k_i)} \quad i = 1, \ldots, n, \quad j = k_i + 1, \ldots, k \\
& & x_{ij} \text{ binary for } i = 1, ..., n, \quad j = 1, .., k_i.
\end{array}$$

The optimal solution for (scaled-RSP) is found by applying the dynamic programming procedure in Section 4.3 with scaled sizes D' and $s'_1, \ldots, s'_n$.

The running time of finding the optimal solution for (scaled-RSP) with the dynamic programming procedure, is $O(nD'^k) = O(\frac{n^{k+1}}{\epsilon^k})$.

Next we define the (ϵ-relaxed RSP) and then prove that any feasible solution for (scaled-RSP), including $\hat{\mathbf{x}}$, is feasible for (ϵ-relaxed RSP).

The ϵ-relaxed RSP

The (ϵ-relaxed RSP) formulation allows the cascading constraints to be violated by up to ϵD as follows:

$$\begin{array}{lll} (\epsilon\text{-relaxed RSP}) & \max & z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\ \text{subject to} & & \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \leq \ell D + \epsilon D \quad \ell = 1, .., k \\ & & \sum_{j=1}^{k_i} x_{ij} = 1 \qquad i = 1, \ldots, n \\ & & x_{ij} = x_{i,(j-k_i)} \quad i = 1, \ldots, n, \quad j = k_i + 1, \ldots, k \\ & & x_{ij} \text{ binary for } i = 1, ..., n, \quad j = 1, .., k_i. \end{array}$$

We refer to the constraints $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \leq \ell D + \epsilon D$ as the *ϵ-relaxed cascading constraints.* We next show that the effect of the ϵ-relaxed cascading constraints on the optimal solution is at most ϵD.

Lemma 4.5. *The peak inventory level of any feasible solution* $\mathbf{x}$ *to (ϵ-relaxed RSP) is at most* $V_k(\mathbf{x}) + \epsilon D$.

Proof. Any feasible solution $\mathbf{x}$ for (ϵ-relaxed RSP) is a valid assignment, so Lemma 4.3 applies. That is, $V_\ell(\mathbf{x}) = V_k(\mathbf{x}) + \left(\sum_{j=1}^{\ell} Q_j(\mathbf{x}) - \ell D\right)$ for $\ell = 1, ..., k$. The ϵ-relaxed cascading constraints state that $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) - \ell D \leq \epsilon D$ for all ℓ. So when $\mathbf{x}$ is a feasible solution of (ϵ-relaxed RSP), $V_\ell(\mathbf{x}) \leq V_k(\mathbf{x}) + \epsilon D$ for all ℓ, and hence, $V(\mathbf{x}) = \max_\ell V_\ell(\mathbf{x}) \leq V_k(\mathbf{x}) + \epsilon D$. □

The next lemma proves that any feasible solution for (scaled-RSP), including $\hat{\mathbf{x}}$, is feasible for (ϵ-relaxed RSP).

Lemma 4.6. *Any assignment* $\mathbf{x}$ *that is feasible for (scaled-RSP) is feasible for (ϵ-relaxed RSP).*

Proof. In both problems $\mathbf{x}$ is required to be a valid assignment. It remains to show that $\mathbf{x}$ satisfies the ϵ-relaxed cascading constraints, that is, $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \leq \ell D + \epsilon D$ for $\ell = 1, .., k$.

By definition, $s'_i = \lfloor \frac{s_i}{T} \rfloor$. So $s_i < T(s'_i + 1)$ and thus,

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}) = \sum_{j=1}^{\ell}\sum_{i=1}^{n} s_i \mathbf{x}_{ij} \leq \sum_{j=1}^{\ell}\sum_{i=1}^{n} T(s'_i + 1)\mathbf{x}_{ij} = T\left(\sum_{j=1}^{\ell}\sum_{i=1}^{n} s'_i \mathbf{x}_{ij} + \sum_{j=1}^{\ell}\sum_{i=1}^{n} \mathbf{x}_{ij}\right). \quad (4.4)$$

Since $\mathbf{x}$ is feasible for (scaled-RSP) and $D' = \frac{D}{T}$,

$$\sum_{j=1}^{\ell}\sum_{i=1}^{n} s'_i \mathbf{x}_{ij} = \sum_{j=1}^{\ell} Q'_j(\mathbf{x}) \le \ell D' = \frac{\ell D}{T}. \tag{4.5}$$

For $\ell = 1, ..., k$,

$$\sum_{j=1}^{\ell}\sum_{i=1}^{n} \mathbf{x}_{ij} \le \sum_{j=1}^{k}\sum_{i=1}^{n} \mathbf{x}_{ij} \le kn \tag{4.6}$$

Hence from inequalities (4.10), (4.11) and (4.12),

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le T\left(\frac{\ell D}{T} + nk\right) = \ell D + Tkn = \ell D + \epsilon D.$$

□

Using the relationship between reorder sizes s_i and the scaled sizes s'_i, we show that for any feasible solution of (scaled-RSP), $\mathbf{x}$, the objective with original sizes $z(\mathbf{x}) = \sum_{j=1}^{k}(k - j + 1)Q_j(\mathbf{x})$ is closely approximated by the objective with scaled sizes $z'(\mathbf{x}) = \sum_{j=1}^{k}(k - j + 1)Q'_j(\mathbf{x})$, corrected for the scaling factor T:

The approximation property of the solution to (scaled-RSP)

Lemma 4.7. *For any assignment of items* $\mathbf{x}$ *feasible for (scaled-RSP), the values of the objective function with original and scaled sizes,* $z(\mathbf{x})$ *and* $z'(\mathbf{x})$ *respectively, satisfy,*

$$Tz'(\mathbf{x}) \;\le\; z(\mathbf{x}) \;\le\; Tz'(\mathbf{x}) + \epsilon k D.$$

Proof. Recall that $s'_i = \lfloor \frac{s_i}{T} \rfloor$, so $Ts'_i \le s_i < T(s'_i + 1)$. We derive the lower bound on $z(\mathbf{x})$ as follows:

$$\begin{aligned} z(\mathbf{x}) &= \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\ &= \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} s_i x_{ij} \\ &\ge T \cdot \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} s'_i x_{ij} \\ &= T \cdot \sum_{j=1}^{k}(k-j+1))Q'_j(\mathbf{x}) \\ &= Tz'(\mathbf{x}). \end{aligned} \tag{4.7}$$

The upper bound on $z(\mathbf{x})$ can be derived as follows:

$$
\begin{aligned}
z(\mathbf{x}) &= \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\
&= \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} s_i x_{ij} \\
&\le T \cdot \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n}(s_i'+1)x_{ij} \\
&= T \cdot \left[\sum_{j=1}^{k}(k-j+1)Q_j'(\mathbf{x}) + \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} x_{ij}\right] \\
&\le Tz'(\mathbf{x}) + Tk^2 n \\
&= Tz'(\mathbf{x}) + \epsilon k D. \qquad (4.8)
\end{aligned}
$$

□

Lemma 4.7 leads to the following lower bound on $z(\hat{\mathbf{x}})$ for $\hat{\mathbf{x}}$ being an optimal solution of (scaled-RSP):

Theorem 4.2. *For any feasible solution* $\mathbf{x}$ *of (RSP),* $z(\hat{\mathbf{x}}) \ge z(\mathbf{x}) - \epsilon k D$.

Proof. By Lemma 4.7, we know $z(\hat{\mathbf{x}}) \ge Tz'(\hat{\mathbf{x}})$. Since any feasible solution of (RSP), $\mathbf{x}$, is also feasible for (scaled-RSP), we use the upper bound of $z(\mathbf{x})$ from Lemma 4.7 to get:

$$Tz'(\mathbf{x}) \ge z(\mathbf{x}) - \epsilon k D.$$

Because $\hat{\mathbf{x}}$ is optimal for (scaled-RSP), it follows that $z'(\hat{\mathbf{x}}) \ge z'(\mathbf{x})$. Combining the three inequalities, we get

$$z(\hat{\mathbf{x}}) \ge Tz'(\hat{\mathbf{x}}) \ge Tz'(\mathbf{x}) \ge z(\mathbf{x}) - \epsilon k D.$$

□

Consequently, the optimal solution $\hat{\mathbf{x}}$ for (scaled-RSP) attains a objective value $z(\hat{\mathbf{x}})$ that is at least as much as the optimal objective of (RSP) minus $\epsilon k D$.

The $(1+\epsilon')$-approximation bound

From the discussion above, we know that the optimal solution for (scaled-RSP) $\hat{\mathbf{x}}$ is a valid assignment whose inventory levels at time k approximates that maximum inventory level, and the value $z(\mathbf{x})$ approximates the optimal objective value of (RSP). We will prove here that $\hat{\mathbf{x}}$ is an $(1+\epsilon')$-approximation solution for $\epsilon' = 2\epsilon$ and any $\epsilon > 0$.

Theorem 4.3. *The optimal solution $\hat{\mathbf{x}}$ for (scaled-RSP) is a $(1+\epsilon')$-approximation solution for the RSP.*

Proof. Assignment $\hat{\mathbf{x}}$ is valid as it is feasible for (scaled-RSP). So we just need to prove the approximation factor for the peak inventory level.

Let $\mathbf{x}^*$ be an optimal solution of (RSP), and V^* the corresponding peak inventory level.

As stated in Theorem 4.2, $z(\hat{\mathbf{x}}) \geq z(\mathbf{x}) - \epsilon kD$ for any $\mathbf{x}$ that is feasible of (RSP), including $\mathbf{x}^*$. From Lemma 4.4, the inventory levels at time k for $\hat{\mathbf{x}}$ and $\mathbf{x}^*$ are $V_k(\hat{\mathbf{x}}) = \frac{C}{k} - \frac{z(\hat{\mathbf{x}})}{k}$ and $V_k(\mathbf{x}^*) = \frac{C}{k} - \frac{z(\mathbf{x}^*)}{k}$ respectively. Therefore,

$$V_k(\hat{\mathbf{x}}) = \frac{C}{k} - \frac{z(\hat{\mathbf{x}})}{k} \leq \frac{C}{k} - \frac{z(\mathbf{x}^*)}{k} + \frac{\epsilon kD}{k} = V_k(\mathbf{x}^*) + \epsilon D.$$

From Lemma 4.5 it follows that the peak inventory level for $\hat{\mathbf{x}}$ satisfies $V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon D$. Since $\mathbf{x}^*$ is a solution of (RSP), the peak inventory level for $\mathbf{x}^*$ is $V^* = V_k(\mathbf{x}^*)$. Hence,

$$V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon D \leq V^* + 2\epsilon D.$$

That is, for the optimum peak storage of (RSP), V^*, and for the optimal solution of (scaled-RSP) $\hat{\mathbf{x}}$, the ratio $V(\hat{\mathbf{x}})/V^*$ is at most $1 + 2\epsilon D/V^*$. Observe that V^* must be at least the per unit time demand D, it follows that $2\epsilon D/V^* \leq 2\epsilon$.

Therefore, the ratio $V(\hat{\mathbf{x}})/V^*$ is at most $1 + 2\epsilon = 1 + \epsilon'$. Hence, $\hat{\mathbf{x}}$ is a $(1+\epsilon')$-approximate solution to the RSP. □

The complexity of this approximation procedure is $O(\frac{n^{(k+1)}}{\epsilon^k})$ for constant k. Noted that $\frac{1}{\epsilon} = O(\frac{1}{\epsilon'})$. Therefore the complexity of the RSP $(1+\epsilon')$-approximation algorithm is $O(\frac{n^{(k+1)}}{\epsilon'^k})$, which is polynomial in n and $\frac{1}{\epsilon'}$ for constant k. And a family of $(1+\epsilon')$-approximation algorithms with complexity that is polynomial in n and $\frac{1}{\epsilon'}$ is called a Fully Polynomial Time Approximation Scheme.

4.5 A Fully Polynomial-time Approximation Scheme for the Single-cycle RSP with Constant k

In this section, we establish a FPTAS for the single-cycle RSP for constant joint cycle length. This approximation scheme is a family of $(1+\epsilon')$-approximation algorithms for every $\epsilon' > 0$, with fixed-parameter tractable running time $O(\frac{n}{\epsilon^{2k}})$. For any given $\epsilon' > 0$, we let $\epsilon = \epsilon'/4 > 0$. The $(1+\epsilon')$-approximation algorithm for the single-cycle RSP works by first assigning reorder timings of "large" items, where "large" items are those with reorder size greater than ϵD. This large assignment is determined by solving a variant of (RSP) that involves "adjusted remainders". All items that are not large, which are considered "small", are assigned in a greedy fashion to any "adjusted" cascading constraint that still has a positive slack. It is shown that such a solution is within a factor of $1 + \epsilon'$ of the optimal solution. The run time of this approximation algorithm is polynomial in n and $\frac{1}{\epsilon}$. Since $O(\frac{1}{\epsilon}) = O(\frac{1}{\epsilon'})$, this run time is also polynomial in $\frac{1}{\epsilon'}$, and hence this family of algorithms is a fully polynomial approximation scheme which is fixed-parameter tractable in terms of the joint cycle length k.

Since we address here the single-cycle RSP, we use a formulation referred to as (k-RSP) which is the integer programming (RSP) for the single cycle of length k RSP:

$$\begin{array}{lll} \text{(k-RSP)} & \max & z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\ \text{subject to} & & \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \leq \ell D \quad \ell = 1,..,k \\ & & \sum_{j=1}^{k} x_{ij} = 1 \qquad i = 1,\ldots,n \\ & & x_{ij} \text{ binary for } i = 1,...,n, \quad j = 1,..,k. \end{array}$$

We next define the classification of the n items as large or small:
For a given value of $\epsilon > 0$, let the set of large items be $I_{\mathrm{L}} = \{i | s_i > \epsilon D\}$ and the set of small items be $I_{\mathrm{S}} = \{i | s_i \leq \epsilon D\}$. Let $n_{\mathrm{L}} = |I_{\mathrm{L}}|$ denote the number of large items and $n_{\mathrm{S}} = |I_{\mathrm{S}}|$ denote the number of small items. It is observed that since $kD = \sum_{i=1}^{n} s_i \geq \sum_{i \in I_{\mathrm{L}}} s_i > n_{\mathrm{L}} \cdot \epsilon D$, the number of large items n_{L} is bounded by $\frac{k}{\epsilon}$.

Let an $n \times k$ binary matrix $\mathbf{x}^{\mathrm{L}}$ determine the assignment of large items as:

$$x_{ij}^{\mathrm{L}} = \begin{cases} 1 & \text{if } i \in I_{\mathrm{L}}, \text{ and item } i \text{ is ordered on time j} \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, an $n \times k$ binary matrix $\mathbf{x}^{\mathrm{S}}$ is the assignment of small items where,

$$x_{ij}^{\mathrm{S}} = \begin{cases} 1 & \text{if } i \in I_{\mathrm{S}}, \text{ and item } i \text{ is ordered on time j} \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4.2. *An assignment of large (small) items* $\mathbf{x}^L$ ($\mathbf{x}^S$) *is a* valid large (small) assignment *for a given single-cycle RSP instance if and only if any large (small) item i is replenished exactly once in a joint cycle. That is,*

$$\sum_{j=1}^{k} x_{ij}^{L} = 1 \qquad i \in I_L \qquad (\sum_{j=1}^{k} x_{ij}^{S} = 1 \qquad i \in I_S).$$

By Definition 4.1 (of valid assignment) and Definition 4.2, it is clear that the sum of any valid large assignment $\mathbf{x}^{\mathrm{L}}$ and any valid small assignment $\mathbf{x}^{\mathrm{S}}$, $\mathbf{x} = \mathbf{x}^{\mathrm{L}} + \mathbf{x}^{\mathrm{S}}$, is a valid assignment of all items.

Definition 4.3. *For any assignment* $\mathbf{x}$ *and* $I \subseteq \{1,...,n\}$, *the* $n \times k$ *binary matrix* $P_I(\mathbf{x})$ *is said to be the* projection *of the* $\mathbf{x}$ *onto* I *where,*

$$P_I(\mathbf{x})_{ij} = \begin{cases} x_{ij} & \text{if } i \in I \\ 0 & \text{if } i \notin I. \end{cases}$$

It follows that for a valid assignment $\mathbf{x}$, its projection on the large (small) item set $\mathbf{x}^{\mathrm{L}} = P_{I_{\mathrm{L}}}(\mathbf{x})$ ($\mathbf{x}^{\mathrm{S}} = P_{I_{\mathrm{S}}}(\mathbf{x})$) is a valid large (small) assignment, and furthermore, $z(\mathbf{x}) = z(\mathbf{x}^{\mathrm{L}}) + z(\mathbf{x}^{\mathrm{S}})$; $Q_j(\mathbf{x}) = Q_j(\mathbf{x}^{\mathrm{L}}) + Q_j(\mathbf{x}^{\mathrm{S}})$.

The single-cycle FPTAS devised here consists of two stages. In the first stage we determine an assignment of the large items, $\hat{\mathbf{x}}^{\mathrm{L}}$, and in the second stage we assign the small items, $\hat{\mathbf{x}}^{S}$. For a given $\epsilon' > 0$ and $\epsilon = \epsilon'/4$, we show that the valid assignment $\hat{\mathbf{x}} = \hat{\mathbf{x}}^{\mathrm{L}} + \hat{\mathbf{x}}^{\mathrm{S}}$ is a $(1+\epsilon')$-approximate solution.

The Assignment of Large Items

The procedure for assigning the large items addresses a modified (k-RSP) in which the objective function is changed by adding terms dependent on a form of slacks in the cascading constraints, the *adjusted remainders*. We call this problem (modified k-RSP). The modified problem is then scaled in that the order sizes are scaled by $T = \frac{\epsilon^2 D}{k}$. The resulting scaled problem, (scaled-modified k-RSP), is then shown to be solvable using the dynamic programming procedure of Section 4.3 and generating an assignment of large items that has objective function value close to the optimal value of (k-RSP).

The adjusted remainders

The cascading constraints are equivalent to $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{S}}) \leq \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})$ for $\ell = 1, .., k$. Hence the remaining "space" for small items in the first ℓ integer times is no more than $R_\ell(\mathbf{x}^{\mathrm{L}}) = \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})$. We refer to $R_\ell(\mathbf{x}^{\mathrm{L}})$ as the *remainder* of time ℓ. The cascading constraints are then equivalently written as

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{S}}) \leq R_\ell(\mathbf{x}^{\mathrm{L}}) \text{ for } \ell = 1, .., k. \tag{4.9}$$

For any integer ℓ_1, ℓ_2 such that $1 \leq \ell_1 < \ell_2 \leq k$, the constraint of time ℓ_2 in the form of (4.9) implies

$$\sum_{j=1}^{\ell_1} Q_j(\mathbf{x}^{\mathrm{S}}) \leq R_{\ell_2}(\mathbf{x}^{\mathrm{L}}) - \sum_{j=\ell_1+1}^{\ell_2} Q_j(\mathbf{x}^{\mathrm{S}}) \leq R_{\ell_2}(\mathbf{x}^{\mathrm{L}}).$$

Hence these constraints (4.9) are equivalent to $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{S}}) \leq \min_{j \geq \ell} R_j(\mathbf{x}^{\mathrm{L}})$ for $\ell = 1, \ldots, k$. We refer to the right hand sides of these inequalities, $\bar{R}_\ell(\mathbf{x}^{\mathrm{L}}) = \min_{j \geq \ell} R_j(\mathbf{x}^{\mathrm{L}})$, as the *adjusted remainders*. Obviously the adjusted remainder for any inequality ℓ can only be smaller than the respective remainder, $\bar{R}_\ell(\mathbf{x}^{\mathrm{L}}) \leq R_\ell(\mathbf{x}^{\mathrm{L}})$.

To illustrate the concepts of remainders and adjusted remainders, we provide an instance of a single-cycle RSP with five items for $k = 4$ (see Table 4.3). In this instance $D = 24$, and we take $\epsilon = 0.6$. Then 3 items are determined as large and 2 are small:

Consider the assignment of large items: item 1 to time 1, item 2 to time 2, and item 3 to time 4. Figure 4.1 visualizes the four cascading constraints for this assignment of large

i	1	2	3	4	5
s_i	23	18	52	2	1
Large/Small	Large	Large	Large	Small	Small

Table 4.3: A problem instance of single-cycle RSP with $k = 4$.

items: the columns heights indicate the right hand side of the four cascading constraints; rectangles with patterns in each column represent the large items that are replenished before and on the indexed time. So the remainders are represented by the white space in each column. Figure 4.4 shows the corresponding remainders and adjusted remainders.

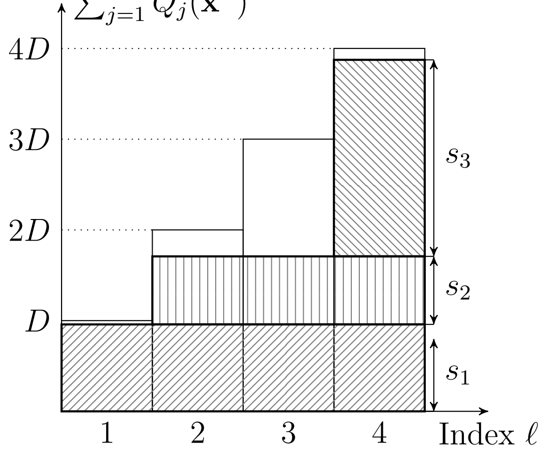


Figure 4.3: An assignment of large items for the example in Table 4.3.

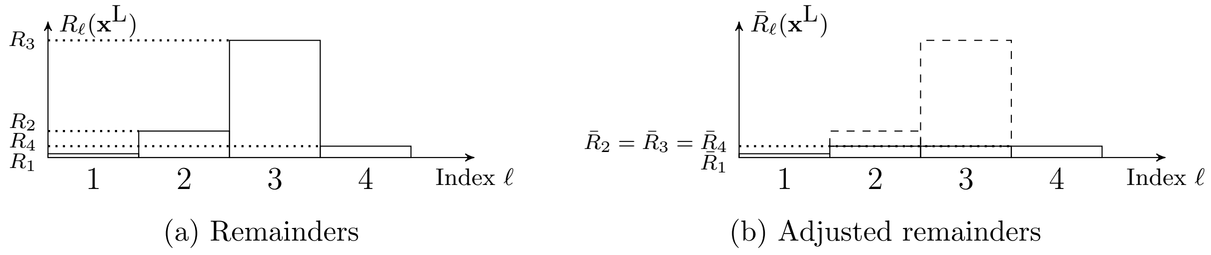


Figure 4.4: The remainders and the adjusted remainders corresponding to Fig 4.3.

The modified objective of (modified k-RSP)

We modified the objective function $z(\mathbf{x})$ by adding the adjusted remainders. Specifically, let the objective function $g(\mathbf{x}^{\mathrm{L}})$ be defined for any valid assignment of large items $\mathbf{x}^{\mathrm{L}}$:

$$g(\mathbf{x}^{\mathrm{L}}) = z(\mathbf{x}^{\mathrm{L}}) + \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^{\mathrm{L}}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^{\mathrm{L}}).$$

An important property of function $g(\cdot)$ is that for $\mathbf{x}$ feasible for (k-RSP) and $\mathbf{x}^{\mathrm{L}}$ its projection to the large items set, $g(\mathbf{x}^{\mathrm{L}})$ is an upper bound of $z(\mathbf{x})$, as proved next:

Lemma 4.8. *For any feasible solution* $\mathbf{x}$ *of (k-RSP) and* $\mathbf{x}^L = P_{I_L}(\mathbf{x})$, $g(\mathbf{x}^L) \geq z(\mathbf{x})$.

Proof. To show $g(\mathbf{x}^{\mathrm{L}}) = z(\mathbf{x}^{\mathrm{L}}) + \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^{\mathrm{L}}) \geq z(\mathbf{x})$ it suffices to prove that for $\mathbf{x}^{\mathrm{S}} = P_{I_{\mathrm{S}}}(\mathbf{x})$, $z(\mathbf{x}^{\mathrm{S}}) = z(\mathbf{x}) - z(\mathbf{x}^{\mathrm{L}}) \leq \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^{\mathrm{L}})$.

By the definition of adjusted remainders, any $\mathbf{x}$ feasible for (k-RSP) satisfies, $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{S}}) \leq \bar{R}_\ell(\mathbf{x}^{\mathrm{L}})$ for $\ell = 1, \ldots, k$. Therefore, $z(\mathbf{x}^{\mathrm{S}}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^{\mathrm{S}}) = \sum_{j=1}^{k}\sum_{\ell=j}^{k} Q_j(\mathbf{x}^{\mathrm{S}}) = \sum_{\ell=1}^{k}\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{S}}) \leq \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^{\mathrm{L}})$.

□

The scaling of (modified k-RSP), (scaled-modified k-RSP)

The data is scaled by the factor $T = \frac{\epsilon^2 D}{k}$ as follows: Let $s'_i = \lfloor \frac{s_i}{T} \rfloor$ be the scaled sizes of items $i = 1, ..., n$ and $D' = \frac{D}{T} = \frac{k}{\epsilon^2}$ be the scaled demand. Let $Q'_j(\mathbf{x}^{\mathrm{L}})$, $\bar{R}'_\ell(\mathbf{x}^{\mathrm{L}})$ and $g'(\mathbf{x}^{\mathrm{L}})$ denote the "scaled" replenishment sizes at time j, the adjusted remainder at time ℓ and the objective function for the scaled sizes s'_i:
$Q'_j(\mathbf{x}^{\mathrm{L}}) = \sum_{i \in I_{\mathrm{L}}} s'_i x^{\mathrm{L}}_{ij}, j = 1, .., k;$
$R'_\ell(\mathbf{x}^{\mathrm{L}}) = \ell D' - \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}}), \ell = 1, .., k;$
$\bar{R}'_\ell(\mathbf{x}^{\mathrm{L}}) = \min_{j \geq \ell} R'_j(\mathbf{x}^{\mathrm{L}}), \ell = 1, .., k;$
$g'(\mathbf{x}^{\mathrm{L}}) = \sum_{j=1}^{k}(k-j+1)Q'_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\ell=1}^{k} \bar{R}'_\ell(\mathbf{x}^{\mathrm{L}}).$

Let the large items be re-indexed from 1 to n_{L}. The scaled problem that is solved to determine the assignment of the large items is (scaled-modified k-RSP), formulated as follows:

$$\begin{array}{lll} \text{(scaled-modified k-RSP)} & \max & g'(\mathbf{x}^{\mathrm{L}}) \\ \text{subject to} & & \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}}) \leq \ell D' \quad \ell = 1, .., k \\ & & \sum_{j=1}^{k} x^{\mathrm{L}}_{ij} = 1 \quad\quad i = 1, ..., n_{\mathrm{L}} \\ & & x^{\mathrm{L}}_{ij} \text{ binary for } i = 1, ..., n_{\mathrm{L}}, \quad j = 1, .., k. \end{array}$$

The optimal solution for (scaled-modified k-RSP) is found by applying the dynamic programming procedure in Section 4.3 with scaled sizes:

Algorithm 2 Large-assignment$(D', s'_1, \ldots, s'_{n_{\mathrm{L}}})$

```
**Initialize**: $f_0(q_1, q_2, \ldots, q_k) \leftarrow 0$ for any $(q_1, q_2, \ldots, q_k)$ with $q_j \in [0, jD']$ and $q_j$ integer.
**for** $h = 1, ..., n_{\mathrm{L}}$ **do**
    **for** $j = 1, ..., k$ and for each $q_j \in \{0, 1, ... jD'\}$ **do**
        $f_h(q_1, q_2, ..., q_k) \leftarrow \begin{cases} \max_{\tau=1,\ldots,k}\{(k+1-\tau)\, s'_h + f_{h-1}(q'_1(\tau), ..., q'_k(\tau))\}, & \text{if } q'_\ell(\tau) \geq 0 \text{ for all } \ell \\ -\infty & \text{otherwise} \end{cases}$
        where $q'_\ell(\tau) = q_\ell - \lfloor \frac{\ell - \tau + k_h}{k_h} \rfloor s_h$.
    **end for**
**end for**
**Output** $\hat{\mathbf{x}}^L$, the solution attaining the value $\max_{(q_1,\ldots,q_k)} f_{n_{\mathrm{L}}}(q_1, ..., q_k) + \sum_{\ell=1}^{k} \bar{r}_j$, where $\bar{r}_\ell = \min_{j \geq \ell}(jD' - q_j)$.
```

The complexity of algorithm Large-assignment, with the scaled sizes, is $O(k^2 \cdot k! \cdot D'^k \cdot n_{\mathrm{L}}) = O(\frac{n}{\epsilon^{2k}})$ for constant k.

Next we define the (ϵ-relaxed k-RSP), the single-cycle version of (ϵ-relaxed RSP) introduced in Section 4.4. And then we prove that any feasible solution for (scaled-modified k-RSP), including $\hat{\mathbf{x}}^L$, is feasible for (ϵ-relaxed k-RSP).

The ϵ-relaxed k-RSP

The (ϵ-relaxed k-RSP) formulation allows the cascading constraints to be violated by up to ϵD as follows:

$$\begin{array}{lll} (\epsilon\text{-relaxed k-RSP}) & \max & z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\ \text{subject to} & & \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \leq \ell D + \epsilon D \quad \ell = 1, .., k \\ & & \sum_{j=1}^{k} x_{ij} = 1 \qquad i = 1, \ldots, n \\ & & x_{ij} \text{ binary for } i = 1, ..., n, \quad j = 1, .., k. \end{array}$$

Since the single-cycle is a special case of multi-cycle. The following lemma follows from Lemma 4.5.

Lemma 4.9. *The peak inventory level of any feasible solution* $\mathbf{x}$ *to (ϵ-relaxed k-RSP) is at most* $V_k(\mathbf{x}) + \epsilon D$.

The next lemma proves that any feasible solution for (scaled-modified k-RSP), including $\hat{\mathbf{x}}^L$, is feasible for (ϵ-relaxed k-RSP).

Lemma 4.10. *Any valid assignment of large items* $\mathbf{x}^L$ *that is feasible for (scaled-modified k-RSP) satisfies,* $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^L) \leq \ell D + \epsilon D$ *for* $\ell = 1, .., k$.

Proof. By definition, $s'_i = \lfloor \frac{s_i}{T} \rfloor$. So $s_i < T(s'_i + 1)$ and thus,

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}}) = \sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}} s_i \mathbf{x}_{ij}^{\mathrm{L}} \leq \sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}} T(s'_i + 1)\mathbf{x}_{ij}^{\mathrm{L}} = T\left(\sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}} s'_i \mathbf{x}_{ij}^{\mathrm{L}} + \sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}} \mathbf{x}_{ij}^{\mathrm{L}}\right). \quad (4.10)$$

Since $\mathbf{x}^{\mathrm{L}}$ is feasible for (Scaled Large Packing) and $D' = \frac{1}{\epsilon^2}$,

$$\sum_{j=1}^{\ell} \sum_{i=1}^{n_{\mathrm{L}}} s_i' \mathbf{x}_{ij}^{\mathrm{L}} = \sum_{j=1}^{\ell} Q_j'(\mathbf{x}^{\mathrm{L}}) \leq \ell D' = \frac{\ell D}{T}. \tag{4.11}$$

Since the reorder size for any large item is greater than ϵD, and the sum of reorder sizes of large items is bounded by the total reorder sizes kD, we can infer that the number of large items is bounded by $n_{\mathrm{L}} < \frac{k}{\epsilon}$. From feasibility of x^{L} for (Scaled Large Packing), we also know that $\sum_{j=1}^{k} x_{ij}^{\mathrm{L}} = 1$ for any large item i. Hence, for $\ell = 1, ..., k$,

$$\sum_{j=1}^{\ell} \sum_{i=1}^{n_{\mathrm{L}}} \mathbf{x}_{ij}^{\mathrm{L}} \leq \sum_{j=1}^{k} \sum_{i=1}^{n_{\mathrm{L}}} \mathbf{x}_{ij}^{\mathrm{L}} = \sum_{i=1}^{n_{\mathrm{L}}} \left(\sum_{j=1}^{k} \mathbf{x}_{ij}^{\mathrm{L}} \right) = n_{\mathrm{L}} < \frac{k}{\epsilon}. \tag{4.12}$$

Hence from inequalities (4.10), (4.11) and (4.12),

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}}) \leq T \left(\frac{\ell D}{T} + \frac{k}{\epsilon} \right) = \ell D + \epsilon D.$$

□

Using the relationship between reorder sizes s_i and the scaled sizes s_i', we show that for any feasible solution of (Scaled Large Packing), $\mathbf{x}^{\mathrm{L}}$, the objective with original sizes $g(\mathbf{x}^{\mathrm{L}})$ is closely approximated by the objective with scaled sizes $g'(\mathbf{x}^{\mathrm{L}}) = \sum_{j=1}^{k} (k - j + 1) Q_j'(\mathbf{x}^{\mathrm{L}}) + \sum_{\ell=1}^{k} \bar{R}_\ell'(\mathbf{x}^{\mathrm{L}})$, corrected for the scaling factor T:

The approximation property of the solution to (scaled-modified k-RSP)

Theorem 4.4. *For any assignment of large items $\mathbf{x}^L$ feasible for (scaled-modified-k-RSP$_1$), the values of the objective function with original and scaled sizes, $g(\mathbf{x}^L)$ and $g'(\mathbf{x}^L)$ respectively, satisfy,*

$$T \cdot g'(\mathbf{x}^L) - \epsilon k^2 D \;\leq\; g(\mathbf{x}^L) \;\leq\; T \cdot g'(\mathbf{x}^L) + \epsilon k^2 D.$$

Proof. Recall that $s_i' = \lfloor \frac{s_i}{T} \rfloor$, so $T s_i' \leq s_i < T(s_i' + 1)$. So for any integer time j,

$$Q_j(\mathbf{x}^{\mathrm{L}}) = \sum_{i=1}^{n_{\mathrm{L}}} s_i x_{ij}^{\mathrm{L}} < T \cdot \sum_{i=1}^{n_{\mathrm{L}}} (s_i' + 1) x_{ij}^{\mathrm{L}} = T \cdot \left(Q_j'(\mathbf{x}^{\mathrm{L}}) + \sum_{i=1}^{n_{\mathrm{L}}} x_{ij}^{\mathrm{L}} \right)$$

The second term in the parentheses, $\sum_{i=1}^{n_{\mathrm{L}}} x_{ij}^{\mathrm{L}}$, must be less tan or equal to the number of large items, which is bounded by $\frac{k}{\epsilon}$. Therefore we derive the following inequality

$$Q_j(\mathbf{x}^{\mathrm{L}}) < T \cdot \left(Q_j'(\mathbf{x}^{\mathrm{L}}) + \frac{k}{\epsilon} \right) = T \cdot Q_j'(\mathbf{x}^{\mathrm{L}}) + \epsilon D \quad \text{for } j = 1, ..., k. \tag{4.13}$$

Using $s_i \geq Ts'_i$ for any i, we get

$$Q_j(\mathbf{x}^{\mathrm{L}}) = \sum_{i=1}^{n_{\mathrm{L}}} s_i x_{ij}^{\mathrm{L}} \geq T \cdot \sum_{i=1}^{n_{\mathrm{L}}} s'_i x_{ij}^{\mathrm{L}} = T \cdot Q'_j(\mathbf{x}^{\mathrm{L}}) \quad \text{for } j = 1, ..., k. \tag{4.14}$$

Recall that the adjusted remainder of time τ is $\bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) = \min_{\ell \geq \tau} R_\ell = \min_{\ell \geq \tau} \left(\ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})\right)$, and that the scaled adjusted remainder of time τ is $\bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}) = \min_{\ell \geq \tau} \left(\ell D' - \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right)$. We derive from inequality (4.13) that for any time τ,

$$\begin{aligned}
\bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) &= \min_{\ell \geq \tau} \left(\ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})\right) \\
&\geq \min_{\ell \geq \tau} \left[\ell D - \sum_{j=1}^{\ell} \left(T \cdot Q'_j(\mathbf{x}^{\mathrm{L}}) + \epsilon D\right)\right] \\
&\geq \min_{\ell \geq \tau} \left[\ell D - T \cdot \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right] - \epsilon k D \\
&= T \cdot \bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}) - \epsilon k D.
\end{aligned} \tag{4.15}$$

And we derive from inequality (4.14) that for any time τ,

$$\bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) = \min_{\ell \geq \tau} \left(\ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})\right) \leq \min_{\ell \geq \tau} \left(\ell D - T \cdot \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right) = T \cdot \bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}) \tag{4.16}$$

Using the inequalities (4.13) and (4.16), we prove the upper bound on $g(\mathbf{x}^{\mathrm{L}})$ as follows:

$$\begin{aligned}
g(\mathbf{x}^{\mathrm{L}}) &= \sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\tau=1}^{k} \bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) \\
&< \sum_{j=1}^{k} (k-j+1) \left(T \cdot Q'_j(\mathbf{x}^{\mathrm{L}}) + \epsilon D\right) + \sum_{\tau=1}^{k} T \cdot \bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}) \\
&= T \cdot \left[\sum_{j=1}^{k} (k-j+1) Q'_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\tau=1}^{k} \bar{R}'_\tau(\mathbf{x}^{\mathrm{L}})\right] + \epsilon D \cdot \sum_{j=1}^{k} (k-j+1) \\
&\leq T \cdot g'(\mathbf{x}^{\mathrm{L}}) + \epsilon k^2 D.
\end{aligned}$$

The lower bound on $g(\mathbf{x}^{\mathrm{L}})$ follows from inequalities (4.14) and (4.15):

$$\begin{aligned}
g(\mathbf{x}^{\mathrm{L}}) &= \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\tau=1}^{k}\bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) \\
&\geq \sum_{j=1}^{k}(k-j+1)T\cdot Q'_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\tau=1}^{k}\Big(T\cdot\bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}) - \epsilon k D\Big) \\
&= T\cdot\left[\sum_{j=1}^{k}(k-j+1)Q'_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\tau=1}^{k}\bar{R}'_\tau(\mathbf{x}^{\mathrm{L}})\right] - \epsilon k^2 D \\
&= T\cdot g'(\mathbf{x}^{\mathrm{L}}) - \epsilon k^2 D.
\end{aligned}$$

This completes the proof of the statement of the theorem. □

Theorem 4.4 leads to the following lower bound on $g(\hat{\mathbf{x}}^{\mathrm{L}})$ for $\hat{\mathbf{x}}^{\mathrm{L}}$ being an optimal solution of (scaled-modified k-RSP):

Theorem 4.5. *For any feasible solution $\mathbf{x}$ of (k-RSP) and $\mathbf{x}^L = P_{I_L}(\mathbf{x})$, $g(\hat{\mathbf{x}}^L) \geq g(\mathbf{x}^L) - \delta(\epsilon)$ where $\delta(\epsilon) = 2\epsilon k^2 D$.*

Proof. By Theorem 4.4, we have the lower bound of $g(\hat{\mathbf{x}}^{\mathrm{L}})$:

$$g(\hat{\mathbf{x}}^{\mathrm{L}}) \geq T\cdot g'(\hat{\mathbf{x}}^{\mathrm{L}}) - \epsilon k^2 D.$$

Since for any feasible solution of (k-RSP), the projection to the large items set, $\mathbf{x}^{\mathrm{L}}$, is also feasible for (scaled-modified k-RSP), we use the upper bound of $g(\mathbf{x}^{\mathrm{L}})$ from Theorem 4.4 to get:

$$Tg'(\mathbf{x}^{\mathrm{L}}) \geq g(\mathbf{x}^{\mathrm{L}}) - \epsilon k^2 D$$

Because $\hat{\mathbf{x}}^{\mathrm{L}}$ is optimal for (scaled-modified k-RSP), it follows that $g'(\hat{\mathbf{x}}^{\mathrm{L}}) \geq g'(\mathbf{x}^{\mathrm{L}})$. Combining the three inequalities, we get

$$\begin{aligned}
g(\hat{\mathbf{x}}^{\mathrm{L}}) &\geq T\cdot g'(\hat{\mathbf{x}}^{\mathrm{L}}) - \epsilon k^2 D \\
&\geq T\cdot g'(\mathbf{x}^{\mathrm{L}}) - \epsilon k^2 D \\
&\geq g(\mathbf{x}^{\mathrm{L}}) - \epsilon k^2 D - \epsilon k^2 D \\
&= g(\mathbf{x}^{\mathrm{L}}) - 2\epsilon k^2 D.
\end{aligned}$$

□

For any feasible solution $\mathbf{x}$ of (k-RSP) and $\mathbf{x}^{\mathrm{L}} = P_{I_{\mathrm{L}}}(\mathbf{x})$, Lemma 4.8 states that $g(\mathbf{x}^{\mathrm{L}}) \geq z(\mathbf{x})$. Hence, a corollary of Theorem 4.5 and Lemma 4.8 is:

Corollary 4.1. *For any feasible solution $\mathbf{x}$ of (k-RSP), $g(\hat{\mathbf{x}}^L) \geq z(\mathbf{x}) - \delta(\epsilon)$ where $\delta(\epsilon) = 2\epsilon k^2 D$.*

Consequently, the algorithm Large-assignment yields an output $\hat{\mathbf{x}}^{\mathrm{L}}$ such that $g(\hat{\mathbf{x}}^{\mathrm{L}})$ is at least as large as the optimal objective of (k-RSP) minus $\delta(\epsilon)$.

The Assignment of Small Items

Given the assignment of large items $\hat{\mathbf{x}}^{\mathrm{L}}$ and the corresponding adjusted remainders $\bar{R}_\ell(\mathbf{x}^{\mathrm{L}})$, we derive a valid assignment of the small items so that the joint large and small items assignment is a feasible solution to (ϵ-relaxed RSP).

As stated in section 4.5, the cascading constraints are equivalent to,

$$\sum_{j=1}^{\ell_1} Q_j(\mathbf{x}^{\mathrm{S}}) \leq \bar{R}_\ell(\mathbf{x}^{\mathrm{L}}) \qquad \text{for } \ \ell = 1, ..., k. \tag{4.17}$$

The following greedy procedure assigns any yet unassigned small item to the lowest integer index time with positive adjusted remainder. For simplicity, the small items are re-indexed 1 to n_{S} in the procedure's description provided next:

Algorithm 3 Small-assignment$(\bar{R}_1(\hat{\mathbf{x}}^{\mathrm{L}}), \ldots, \bar{R}_k(\hat{\mathbf{x}}^{\mathrm{L}}), s_1, \ldots, s_{n_{\mathrm{S}}})$

```
$\tilde{R}_\ell(0) \leftarrow \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) \quad \ell = 1, ..., k.$
for $i = 1, ..., n_{\mathrm{S}}$ do
    $j(i) \leftarrow \min\{\ell | \tilde{R}_\ell(i-1) > 0\}.$
    Assign item $i$ to time $j(i)$.
    $\tilde{R}_\ell(i) \leftarrow \tilde{R}_\ell(i-1) \quad \ell = 1, ..., j(i)-1.$
    $\tilde{R}_\ell(i) \leftarrow \tilde{R}_\ell(i-1) - s_i \quad \ell = j(i), ..., k.$
end for
```

Let $\hat{\mathbf{x}}^{\mathrm{S}}$ denote the output assignment of the small items of Small-assignment.

Lemma 4.11. *Algorithm* Small-assignment *is correct, and its complexity is linear, $O(n_S)$.*

Proof. To prove correctness one needs to prove that the algorithm terminates only after all small items were assigned, meaning that there is always a positive adjusted remainder available for each small item.

The notation used in Small-assignment, $\tilde{R}_\ell(h)$, is the slack of cascading constraint (4.17) at time ℓ after iteration h. That is, $\tilde{R}_\ell(h) = \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) - \sum_{j=1}^{\ell} \sum_{i=1}^{h} s_i \hat{x}_{ij}^{\mathrm{S}}$. Also, $\tilde{R}_\ell(n_{\mathrm{S}}) = \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) - \sum_{j=1}^{\ell} \sum_{i=1}^{n_{\mathrm{S}}} s_i \hat{x}_{ij}^{\mathrm{S}} = \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) - \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{S}})$.

By the definition of remainders and adjusted remainders, it follows that the initial slack corresponding to day k is $\tilde{R}_k(0) = \bar{R}_k(\hat{\mathbf{x}}^{\mathrm{L}}) = R_k(\hat{\mathbf{x}}^{\mathrm{L}}) = kD - \sum_{i \in I_L} s_i = \sum_{i \in I_S} s_i$. In each iteration of Small-assignment, the slack corresponding to day k is reduced by the reorder size of the small item assigned at that iteration. So the slack of day k is positive until all small items have been assigned. That is, Small-assignment guarantees an assignment of all the small items.

The complexity of Small-assignment is linear since there are n_{S} iterations, each of which runs in $O(1)$ steps when k is a constant, as assumed here.

□

As an illustrative example consider the adjusted remainders in Figure 4.4 and the small items in Table 4.3. SMALL-ASSIGNMENT assigns item 4 to time 1 and item 5 to time 2. Figure 4.5 visualizes this assignment example of small items. Columns heights correspond to the value of the adjusted remainders. The rectangles with patterns in each column are the small items that are replenished on or prior to the indexed time. The combined assignment of all items is shown by Figure 4.6. In Figure 4.6a, the column heights are truncated by the same amount as the difference between remainder and adjusted remainders. Figure 4.6b combines the assignment of large items and small items in the truncated columns.

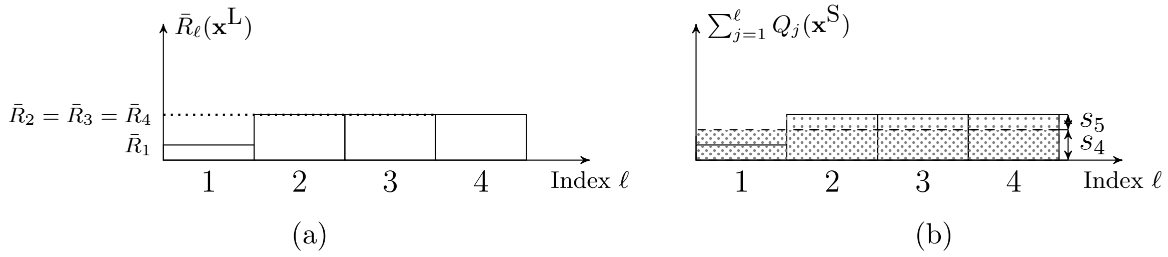


Figure 4.5: The adjusted remainders (a) and assignment of small items (b) for the example in Table 4.3 by SMALL-ASSIGNMENT.

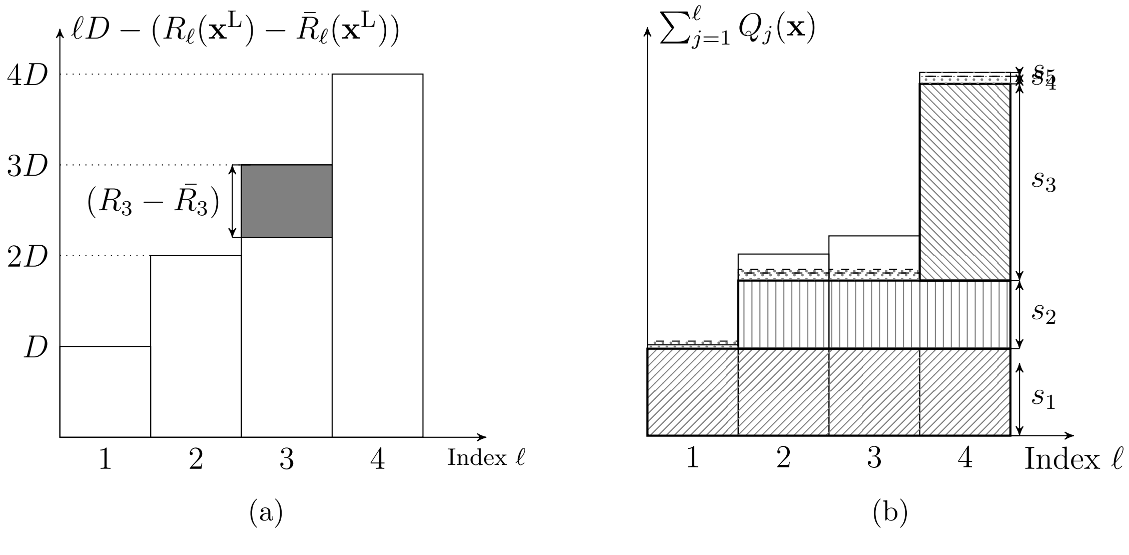


Figure 4.6: Assignment of all items in Table 4.3.

The update of the slacks $\tilde{R}_\ell(h)$ in the algorithm is such that the following properties hold:

Property 1. The slacks prior to the assigning timing $j(h)$ are unaffected.

Property 2. The slacks of time $j(h)+1, ..., k$ are reduced by the same amount as the slack of assigning timing $j(h)$.

We note that Property 1 is not satisfied by the (RSP$_0$) formulation, and that Property 2 does not hold for multi-cycle RSP.

From the two properties it follows that the positive slacks at each iteration are non-decreasing in the integer indices of time from 1 to k.

Lemma 4.12. $0 < \tilde{R}_{j(h)}(h-1) \leq \tilde{R}_{j(h)+1}(h-1) \leq ... \leq \tilde{R}_k(h-1)$ *for any* $1 \leq h \leq n_S$.

Proof. We prove this by induction on the iteration index.
Base: Since the adjusted remainders are defined as $\bar{R}_\ell(\hat{\mathbf{x}}^L) = \min_{j\geq \ell} R_\ell(\hat{\mathbf{x}}^L)$, we have $\bar{R}_1(\hat{\mathbf{x}}^L) \leq \bar{R}_2(\hat{\mathbf{x}}^L) \leq ... \leq \bar{R}_k(\hat{\mathbf{x}}^L)$. And as $\tilde{R}_\ell(0) = \bar{R}_\ell(\hat{\mathbf{x}}^L)$ for all ℓ, we also have $\tilde{R}_1(0) \leq \tilde{R}_2(0) \leq ... \leq \tilde{R}_k(0)$. The algorithm SMALL-ASSIGNMENT assign $j(1)$ to be the smallest integer index such that $\tilde{R}_{j(1)}(0) > 0$. Therefore, $0 < \tilde{R}_{j(1)}(0) \leq \tilde{R}_{j(1)+1}(0) \leq ... \leq \tilde{R}_k(0)$, the statement is true for $h = 1$.
Inductive Step: Assume that the statement holds for h, $1 \leq h \leq n_{\mathrm{S}} - 1$, now we prove for $h+1$: Since $j(h+1) \geq j(h)$, the assumption implies $\tilde{R}_{j(h+1)}(h-1) \leq \tilde{R}_{j(h+1)+1}(h-1) \leq ... \leq \tilde{R}_k(h-1)$. As $\tilde{R}_\ell(h) = \tilde{R}_\ell(h-1) - s_h$ for all $\ell \geq j(h)$, $\tilde{R}_{j(h+1)}(h) \leq \tilde{R}_{j(h+1)+1}(h) \leq ... \leq \tilde{R}_k(h)$. Moreover, by the algorithm SMALL-ASSIGNMENT, we have $\tilde{R}_{j(h+1)}(h) > 0$. Therefore, $0 < \tilde{R}_{j(h+1)}(h) \leq \tilde{R}_{j(h+1)+1}(h) \leq ... \leq \tilde{R}_k(h)$

□

Next we show that the joint assignment of large items with the output assignment of small items $\hat{\mathbf{x}}^{\mathrm{S}}$, yields a feasible solution for (ϵ-relaxed k-RSP).

Lemma 4.13. *The assignment* $\hat{\mathbf{x}} = \hat{\mathbf{x}}^L + \hat{\mathbf{x}}^S$ *is feasible for (ϵ-relaxed k-RSP).*

Proof. It is easy to see that $\hat{\mathbf{x}}$ is valid. It remains to show that $\hat{\mathbf{x}}$ satisfies the ϵ-relaxed cascading constraints.

By Lemma 4.10, we know that $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}}) \leq \ell D + \epsilon D$, so $R_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) = \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}}) \geq -\epsilon D$ for any ℓ. Thus $\bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) = \min_{j\geq\ell} R_j(\hat{\mathbf{x}}^{\mathrm{L}}) \geq -\epsilon D$ for any ℓ.

By Lemma 4.12, only the positive slacks could be reduced at each iteration. And as the reorder sizes of small items are less than or equal to ϵD, the reduction is at at most ϵD. Thus, $\tilde{R}_\ell(h) \geq -\epsilon D$ for any ℓ and h, meaning that $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{S}}) = \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) - \tilde{R}_\ell(n_{\mathrm{S}}) \leq \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) + \epsilon D$.

By definition of $\bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}})$, for any ℓ, $\bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) \leq R_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) = \ell D - \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{L}})$, so $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{S}}) + \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{L}}) \leq \ell D + \epsilon D$.

□

Lemma 4.14. $\tilde{R}_\ell(n_S) \leq 0$ *for* $\ell = 1, ..., k$.

Proof. By contradiction, suppose at the conclusion of the algorithm SMALL-ASSIGNMENT, there exists an index ℓ so that $\tilde{R}_\ell(n_{\mathrm{S}}) > 0$. This implies $j(h) \leq \ell$ for any h, meaning that all small items are assigned to the first ℓ integer times. In that case $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{S}}) = \sum_{h\in I_{\mathrm{S}}} s_h$.

Therefore, $\tilde{R}_\ell(n_{\mathrm{S}}) = \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) - \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{S}}) = \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) - \sum_{h\in I_{\mathrm{S}}} s_h \le \bar{R}_k(\hat{\mathbf{x}}^{\mathrm{L}}) - \sum_{i=1}^{n_{\mathrm{S}}} s_i = \left(kD - \sum_{h\in I_{\mathrm{L}}} s_h\right) - \sum_{h\in I_{\mathrm{S}}} s_h = 0$, contradicting the assumption.

□

Theorem 4.6. *Given the assignment of large items $\hat{\mathbf{x}}^L$ that is optimal for (scaled-modified k-RSP),* Small-assignment *will generate an output $\hat{\mathbf{x}}^S$ such that $z(\hat{\mathbf{x}}^S) \ge \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^L)$.*

Proof. From Lemma 4.14, $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{S}}) = \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) - \tilde{R}_\ell(n_{\mathrm{S}}) \ge \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}})$. Therefore,

$$z(\hat{\mathbf{x}}^{\mathrm{S}}) = \sum_{j=1}^{k}(k-j+1)Q(\hat{\mathbf{x}}^{\mathrm{S}}) = \sum_{\ell=1}^{k}\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^{\mathrm{S}}) \ge \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}).$$

□

Theorem 4.6, together with Corollary 4.1, implies the following theorem:

Theorem 4.7. *Let $\hat{\mathbf{x}} = \hat{\mathbf{x}}^L + \hat{\mathbf{x}}^S$, where $\hat{\mathbf{x}}^L$ and $\hat{\mathbf{x}}^S$ are the outputs of algorithms* Large-assignment *and* Small-assignment *respectively. Then for any $\mathbf{x}$ that is feasible for (k-RSP), $z(\hat{\mathbf{x}}) \ge z(\mathbf{x}) - \delta(\epsilon)$.*

Proof. By Theorem 4.6, $z(\hat{\mathbf{x}}^{\mathrm{S}}) \ge \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}})$, so $z(\hat{\mathbf{x}}) = z(\hat{\mathbf{x}}^{\mathrm{L}}) + z(\hat{\mathbf{x}}^{\mathrm{S}}) \ge z(\hat{\mathbf{x}}^{\mathrm{L}}) + \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) = g(\hat{\mathbf{x}}^{\mathrm{L}})$. And by Corollary 4.1, $g(\hat{\mathbf{x}}^{\mathrm{L}}) \ge z(\mathbf{x}) - \delta(\epsilon)$ for any $\mathbf{x}$ that is feasible of (k-RSP). Therefore, $z(\hat{\mathbf{x}}) \ge g(\hat{\mathbf{x}}^{\mathrm{L}}) \ge z(\mathbf{x}) - \delta(\epsilon)$ for any $\mathbf{x}$ that is feasible of (k-RSP).

□

Therefore, assignment $\hat{\mathbf{x}}$ described in Theorem 4.7 attains a objective value $z(\hat{\mathbf{x}})$ that is at least as much as the optimal objective of (k-RSP) minus $\delta(\epsilon)$.

The $(1+\epsilon')$-approximation Algorithm

Combining the theorems of the previous sections, the algorithms Large-assignment and Small-assignment deliver an FPTAS for the single-cycle RSP. That is, a $(1+\epsilon')$-approximation algorithm for any $\epsilon' > 0$. The $(1+\epsilon')$-approximation algorithm consists of following steps:

Algorithm 4 SINGLE-CYCLE APPROXIMATION

```
Step 1 (initialize) Let $\epsilon = \epsilon'/4$; $D = \frac{1}{n}\sum_{i=1}^{n} s_i$;
       (partition the items into large and small) Let $I_{\mathrm{L}} = \{i : s_i > \epsilon D\}$ and $I_{\mathrm{S}} = \{i : s_i \leq \epsilon D\}$;
       (scaling) Let $T = \frac{\epsilon^2 D}{k}$, $s'_i = \lfloor \frac{s_i}{T} \rfloor$ for $i \in I_{\mathrm{L}}$ and let $D' = \frac{D}{T}$;
Step 2 Call LARGE-ASSIGNMENT($D', s'_i$ for $i \in I_{\mathrm{L}}$) to get $\hat{\mathbf{x}}^{\mathrm{L}}$;
       Compute $R_\ell(\mathbf{x}^{\mathrm{L}}) = \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})$ for $\ell = 1, ..., k$;
       Compute $\bar{R}_\ell(\hat{\mathbf{x}}^{\mathrm{L}}) = \min_{j \geq \ell} R_j(\mathbf{x}^{\mathrm{L}})$ for $\ell = 1, ..., k$;
Step 3 Apply SMALL-ASSIGNMENT($\bar{R}_1(\hat{\mathbf{x}}^{\mathrm{L}}), \ldots, \bar{R}_k(\hat{\mathbf{x}}^{\mathrm{L}}), s_i$ for $i \in I_{\mathrm{S}}$) to get $\hat{\mathbf{x}}^{\mathrm{S}}$;
Step 4 Output $\hat{\mathbf{x}}^{(\epsilon)} = \hat{\mathbf{x}}^{\mathrm{L}} + \hat{\mathbf{x}}^{\mathrm{S}}$.
```

Theorem 4.8. SINGLE-CYCLE APPROXIMATION *is a* $(1+\epsilon')$*-approximation algorithm for single-cycle RSP.*

Proof. Let $\mathbf{x}^*$ be an optimal solution of (k-RSP), and V^* the corresponding peak inventory level.

As stated in Theorem 4.7, $z(\hat{\mathbf{x}}) \geq z(\mathbf{x}) - \delta(\epsilon)$ for any $\mathbf{x}$ that is feasible of (k-RSP), including $\mathbf{x}^*$. From Lemma 4.4, the inventory levels at time k for $\hat{\mathbf{x}}$ and $\mathbf{x}^*$ are $V_k(\hat{\mathbf{x}}) = \frac{C}{k} - \frac{z(\hat{\mathbf{x}})}{k}$ and $V_k(\mathbf{x}^*) = \frac{C}{k} - \frac{z(\mathbf{x}^*)}{k}$ respectively. Therefore,

$$V_k(\hat{\mathbf{x}}) = \frac{C}{k} - \frac{z(\hat{\mathbf{x}})}{k} \leq \frac{C}{k} - \frac{z(\mathbf{x}^*)}{k} + \frac{\delta(\epsilon)}{k} = V_k(\mathbf{x}^*) + \frac{\delta(\epsilon)}{k}.$$

From Lemma 4.9 it follows that the peak inventory level for $\hat{\mathbf{x}}$ satisfies $V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon k D$. Since $\mathbf{x}^*$ is a solution of (k-RSP), the peak inventory level for $\mathbf{x}^*$ is $V^* = V_k(\mathbf{x}^*)$. Hence,

$$V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon D \leq V^* + \frac{\delta(\epsilon)}{k} + \epsilon D.$$

That is, for the optimum peak storage of (k-RSP), V^*, and for the output of our SINGLE-CYCLE APPROXIMATION, $\hat{\mathbf{x}}$, the ratio $V(\hat{\mathbf{x}})/V^*$ is at most $1 + \left(\frac{\delta(\epsilon)}{k} + \epsilon D\right)/V^*$. Since $V^* \geq \frac{k+1}{2k}\sum_{i=1}^{n} s_i = \frac{k+1}{2}D$, it follows that

$$\left(\frac{\delta(\epsilon)}{k} + \epsilon D\right)/V^* \leq (2k+1)\,\epsilon D \cdot \frac{2}{(k+1)D} \leq 4\epsilon = \epsilon'.$$

Hence, $\hat{\mathbf{x}}$ is a $(1+\epsilon')$-approximate solution to RSP. □

The complexity of this SINGLE-CYCLE APPROXIMATION procedure is dominated by the complexity of LARGE-ASSIGNMENT, which is $O(\frac{n}{\epsilon^{2k}})$ for constant k. (SMALL-ASSIGNMENT takes linear time in n). Noted that $\frac{1}{\epsilon} = \frac{4}{\epsilon'}$. Therefore the complexity of RSP $(1+\epsilon')$-approximation algorithm is $O(\frac{n}{\epsilon'^{2k}})$, which is polynomial in n and $\frac{1}{\epsilon'}$ for constant k. Therefore, the family of SINGLE-CYCLE APPROXIMATION for any $\epsilon' > 0$ is a Fully Polynomial Time Approximation Scheme.

4.6 A Polynomial-time Approximation Scheme for Single-cycle RSP with Non-constant k

In this section, we combine the $(1+\frac{2}{k})$-approximation algorithm by Hall [18] with the FPTAS we just presented, to get a PTAS for the single-cycle RSP with non-constant k. Since this problem is strongly NP-hard, a PTAS is the best approximation possible.

Hall's $(1+\frac{2}{k})$-approximation algorithm has a complexity of $O(n)$. Recall that the continuous single-cycle RSP has closed form solutions ([30]; [42]; and [62]). Hall's algorithm rounds down the reorder timings of the optimal solution for the continuous problem. As the value of k increases, the resolution of the discrete problem becomes more and more refined, the rounded solution becomes very close to the continuous solution and close to the optimum. (This is because the continuous optimal solution is a lower bound on the value of the discrete optimum.) For the sake of completeness we include the pseudocode of the algorithm below.

Algorithm 5 $(1+\frac{2}{k})$-APPROXIMATION [18]

```
for $i = 1, ..., n$ do
    $j(i) \leftarrow \lfloor k \cdot \frac{\sum_{h=1}^{i} s_h}{\sum_{h=0}^{n} s_h} \rfloor$.
    Assign item $i$ to time $j(i)$.
end for
```

While this $(1+\frac{2}{k})$-approximation algorithm performs better as k increases, our single-cycle FPTAS works well for small values of k where the continuous solution is not close to the discrete solution and Hall's approximation algorithm does not work well.

We take the advantage of the complexity of Hall's algorithm and the approximation factor of our FPTAS into a PTAS which works as follows: when $k > \frac{2}{\epsilon}$, we run Hall's algorithm, which is a $(1+\epsilon)$-approximation algorithm; when $k \leq \frac{2}{\epsilon}$, we run the FPTAS in this paper, and the running time is $O(k^2 \cdot k! \cdot \frac{n}{\epsilon^{2k}}) = O((\frac{2}{\epsilon})! \cdot \frac{n}{\epsilon^{2k+2}})$ when k non-constant. The overall running time $O((\frac{2}{\epsilon})! \cdot \frac{n}{\epsilon^{2k+2}})$ is linear in n for fixed ϵ so it is a polynomial-time approximation scheme.

4.7 Concluding Remarks

In this chapter we resolve the complexity of RSP, the problem of minimizing the peak storage requirement with given reorder sizes and individual cycle lengths. While it was known that

RSP is NP-hard even when the joint cycle length is constant [18], we prove here that the problem is strongly NP-hard for non-constant joint cycle length, and that the problem is weakly NP-hard for constant joint cycle length.

For constant joint cycle length RSP, we further present a pseudo-polynomial time algorithm that solves the problem optimally, and the first known FPTAS for the multi-cycle RSP, and the first known fixed-parameter tractable FPTAS for the single-cycle RSP. For non-constant joint cycle length single-cycle RSP, we devise here the first known PTAS. The question of whether there exists a PTAS for the respective cases of the multi-cycle RSP remains open.

Bibliography

[1] R. K. Ahuja, J. B. Orlin, and T. L. Magnanti. *Network flows: theory, algorithms, and applications*. Prentice-Hall, 1993.

[2] S. Anily. Multi-item replenishment and storage problem (mirsp): heuristics and bounds. *Operations Research*, 39(2):233–243, 1991.

[3] M. Bennett, J. P. Vielma, and J. R. Zubizarreta. Building representative matched samples with multi-valued treatments in large observational studies. *Journal of Computational and Graphical Statistics*, 0(0):1–29, 2020.

[4] F. F. Boctor. Offsetting inventory replenishment cycles to minimize storage space. *European Journal of Operational Research*, 203(2):321–325, 2010.

[5] M. A. Brookhart, S. Schneeweiss, K. J. Rothman, R. J. Glynn, J. Avorn, and T. Stürmer. Variable selection for propensity score models. *American journal of epidemiology*, 163(12):1149–1156, 2006.

[6] R. Busaker and P. J. Gowen. A procedure for determining minimal-cost network flow patterns. Technical report, ORO Technical Report 15, Operational Research Office, John Hopkins University, 1961.

[7] T. Carnes and D. Shmoys. Primal-dual schema for capacitated covering problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 288–302. Springer, 2008.

[8] R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, page 106115, USA, 2000. Society for Industrial and Applied Mathematics.

[9] F. A. Chudak and D. S. Hochbaum. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, 25(5):199–204, 1999.

[10] E. Croot and K. Huang. A class of random algorithms for inventory cycle offsetting. *International Journal of Operational Research*, 18(2):201–217, 2013.

[11] R. H. Dehejia and S. Wahba. Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs. *Journal of the American Statistical Association*, 94(448):1053–1062, 1999.

[12] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[13] G. Gallego, D. Shaw, and D. Simchi-Levi. The complexity of the staggering problem, and other classical inventory problems. *Operations Research Letters*, 12(1):47–52, 1992.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, USA, 1978.

[15] A. S. Gerber and D. P. Green. The effects of canvassing, telephone calls, and direct mail on voter turnout: A field experiment. *American Political Science Review*, 94(3):653–663, 2000.

[16] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *Journal of the ACM (JACM)*, 45(5):783–797, 1998.

[17] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of operations research*, 22(3):513–544, 1997.

[18] N. G. Hall. A comparison of inventory replenishment heuristics for minimizing maximum storage. *American Journal of Mathematical and Management Sciences*, 18(3-4):245–258, 1998.

[19] M. A. Hariga and P. L. Jackson. The warehouse scheduling problem: formulation and algorithms. *IIE transactions*, 28(2):115–127, 1996.

[20] R. Hartley and L. Thomas. The deterministic, two-product, inventory system with capacity constraint. *Journal of the Operational Research Society*, 33(11):1013–1020, 1982.

[21] M. C. Herron and J. Wand. Assessing partisan bias in voting technology: The case of the 2004 New Hampshire recount. *Electoral Studies*, 26(2):247–261, June 2007.

[22] D. E. Ho, K. Imai, G. King, and E. A. Stuart. Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political analysis*, 15(3):199–236, 2007.

[23] D. S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, 1997.

[24] D. S. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29(1):174–200, 1998.

[25] D. S. Hochbaum. Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations. *European Journal of Operational Research*, 140(2):291–321, 2002.

[26] D. S. Hochbaum. Applications and efficient algorithms for integer programming problems on monotone constraints. *Networks*, 2020.

[27] D. S. Hochbaum, A. Levin, and X. Rao. Algorithms and complexity for variants of covariates fine balance. *arXiv preprint arXiv:2009.08172*, 2020.

[28] D. S. Hochbaum, N. Megiddo, J. S. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematical programming*, 62(1-3):69–83, 1993.

[29] D. S. Hochbaum and X. Rao. Network flow methods for the minimum covariates imbalance problem. *arXiv preprint arXiv:2007.06828*, 2020.

[30] E. Homer. Space-limited aggregate inventories with phased deliveries. *Journal of Industrial Engineering*, 17(6):327, 1966.

[31] G. W. Imbens. Nonparametric estimation of average treatment effects under exogeneity: A review. *Review of Economics and statistics*, 86(1):4–29, 2004.

[32] M. Iri. A new method of solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3(1):2, 1960.

[33] W. S. Jewell. Optimal flow through networks. In *Operations Research*, volume 6, pages 633–633, 1958.

[34] R. Kannan. Improved algorithms for integer programming and related lattice problems. In D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 193–206. ACM, 1983.

[35] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[36] H. Kellerer, U. Pferschy, and D. Pisinger. Multiple knapsack problems. In *Knapsack Problems*, pages 285–316. Springer, 2004.

[37] H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[38] I. K. Moon, B. C. Cha, and S. K. Kim. Offsetting inventory cycles using mixed integer programming and genetic algorithm. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 15(3):245–256, 2008.

[39] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[40] N. N. Murthy, W. Benton, and P. A. Rubin. Offsetting inventory cycles of items sharing storage. *European Journal of Operational Research*, 150(2):304–319, 2003.

[41] A. G. Nikolaev, S. H. Jacobson, W. K. T. Cho, J. J. Sauppe, and E. C. Sewell. Balance optimization subset selection (boss): An alternative approach for causal inference with observational data. *Operations Research*, 61(2):398–412, 2013.

[42] E. Page and R. Paul. Multi-product inventory situations with one restriction. *Journal of the Operational Research Society*, 27(4):815–834, 1976.

[43] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 86–92. IEEE, 2000.

[44] S. D. Pimentel, R. R. Kelz, J. H. Silber, and P. R. Rosenbaum. Large, sparse optimal matching with refined covariate balance in an observational study of the health outcomes produced by new surgeons. *Journal of the American Statistical Association*, 110(510):515–527, 2015.

[45] P. R. Rosenbaum. Optimal matching of an optimally chosen subset in observational studies. *Journal of Computational and Graphical Statistics*, 21(1):57–71, 2012.

[46] P. R. Rosenbaum, R. N. Ross, and J. H. Silber. Minimum distance matched sampling with fine balance in an observational study of treatment for ovarian cancer. *Journal of the American Statistical Association*, 102(477):75–83, 2007.

[47] P. R. Rosenbaum and D. B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[48] D. B. Rubin. Bias reduction using mahalanobis-metric matching. *Biometrics*, pages 293–298, 1980.

[49] R. A. Russell and T. L. Urban. Offsetting inventory replenishment cycles. *European Journal of Operational Research*, 254(1):105–112, 2016.

[50] J. J. Sauppe, S. H. Jacobson, and E. C. Sewell. Complexity and approximation results for the balance optimization subset selection model for causal inference in observational studies. *INFORMS Journal on Computing*, 26(3):547–566, 2014.

[51] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006.

[52] E. A. Stuart. Matching methods for causal inference: A review and a look forward. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 25(1):1, 2010.

[53] W. K. Tam Cho, J. J. Sauppe, A. G. Nikolaev, S. H. Jacobson, and E. C. Sewell. An optimization approach for making causal inferences. *Statistica Neerlandica*, 67(2):211–226, 2013.

[54] C.-P. Teo, J. Ou, and K.-C. Tan. Multi-item inventory staggering problems: Heuristics and bounds. In *Proceedings of the ninth annual ACM-SIAM symposium on discrete algorithms*, pages 584–593, 1998.

[55] L. Thomas and R. Hartley. An algorithm for limited capacity inventory problem with staggering. *Journal of the Operational Research Society*, 34(1):81–85, 1983.

[56] N. Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1(2):173–194, 1971.

[57] V. V. Vazirani. *Approximation algorithms.* Springer Science & Business Media, 2013.

[58] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms.* Cambridge university press, 2011.

[59] D. Yang, D. S. Small, J. H. Silber, and P. R. Rosenbaum. Optimal matching with minimal deviation from fine balance in a study of obesity and surgical outcomes. *Biometrics*, 68(2):628–636, 2012.

[60] M. Yao and W. Chu. A genetic algorithm for determining optimal replenishment cycles to minimize maximum warehouse space requirements. *Omega*, 36(4):619–631, 2008.

[61] M. Yao, W. Chu, and Y. Lin. Determination of replenishment dates for restricted-storage, static demand, cyclic replenishment schedule. *Computers & operations research*, 35(10):3230–3242, 2008.

[62] K. Zoller. Deterministic multi-item inventory systems with limited capacity. *Management Science*, 24(4):451–455, 1977.

[63] J. R. Zubizarreta. Using mixed integer programming for matching in an observational study of kidney failure after surgery. *Journal of the American Statistical Association*, 107(500):1360–1371, 2012.