POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Fairness in Combinatorial Optimization

PHILIPPE OLIVIER
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Mai 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Fairness in Combinatorial Optimization

présentée par **Philippe OLIVIER**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Daniel ALOISE, président
Gilles PESANT, membre et directeur de recherche
Andrea LODI, membre et codirecteur de recherche
Guy DESAULNIERS, membre
Swati GUPTA, membre externe

DEDICATION

To my family, and especially to my wife,
without whom this adventure would not have been possible.

À ma famille, et en particulier à ma femme,
sans qui cette aventure n'aurait pas été possible.

"You miss 100% of the shots you don't take.
– Wayne Gretzky"
– Michael Scott

ACKNOWLEDGEMENTS

One can achieve little on his own. The success of this project was made possible through the continued support and advice of my supervisors Gilles Pesant and Andrea Lodi over the several years during which we have worked together. I am very grateful for the time and energy which you invested in our endeavors. I am also thankful for the many opportunities which you have provided me to participate in scientific conferences. Je me souviendrai.

I would also like to thank the *Canada Excellence Research Chair (CERC) in Data Science for Real-time Decision-making* and the *Bourses de soutien à la formation doctorale (BSFD)* of Polytechnique Montréal, for their continued financial support throughout the years.

RÉSUMÉ

Il existe une grande variété au sein des problèmes d'optimisation combinatoire. Dans un contexte réel, les solutions de tels problèmes requièrent souvent une certaine forme d'équilibre. Cependant, la recherche dans le domaine de l'équilibre en optimisation combinatoire est assez limitée. Nous pouvons également observer que lorsqu'un problème comprend une dimension d'équilibre, elle n'est pas toujours intégrée de façon consciencieuse. Dans cette thèse, nous étudions et nous comparons différentes manières d'intégrer la notion d'équilibre en programmation par contraintes et en programmation en nombres entiers. Nous analysons également les caractéristiques et les propriétés associées au différentes formes d'équilibre.

Dans un premier temps, nous étudions le *problème du sac à dos multiple quadratique* avec des conflits et des contraintes d'équilibre. Dans ce problème, nous cherchons à remplir plusieurs sacs à dos avec des objets de différents poids, et dont il existe une valeur ou un conflit entre chaque paire d'objets. Les contraintes d'équilibre s'assurent que les poids des sacs à dos soient équilibrés. Nous présentons des modèles de programmation par contraintes et de programmation en nombres entiers. Nous touchons également à la génération de colonnes, et utilisons un algorithme de « branch and price. » Nous pouvons ainsi comparer la complexité d'implémentation et la performance de ces modèles. Ce problème peut être appliqué à la formation de plans de tables pour des événements, tels des réceptions de mariage.

Dans un deuxième temps, nous nous concentrons sur la méthodologie associée à l'équilibre, et étudions les caractéristiques associées aux différentes formes d'équilibres. Nous présentons quatre manières d'équilibrer des solutions et discutons de leur propriétés. Nous présentons aussi trois problèmes existants où la manière d'atteindre des solutions équilibrées pourrait être améliorée. Ceci nous permet d'émettre plusieurs lignes directrices qui permettraient d'aider un modélisateur à choisir une bonne manière d'équilibrer les solutions d'un problème.

Dans un troisième temps, nous étudions un problème particulier, où nous ne considérons pas l'équilibre local à une seule solution, mais plutôt l'équilibre global atteint à l'issue d'une séquence de solutions. À l'aide de certaines preuves théoriques, nous pouvons démontrer, par exemple, le nombre de solutions nécessaires à atteindre un équilibre global parfait, dans certains cas spéciaux. Nous introduisons aussi une structure nous permettant de résoudre ce problème pour des cas plus compliqués. Nous utilisons cette structure pour résoudre le *problème de répartition d'ambulances*. Un ensemble d'ambulances doit être réparti au travers de différentes zones d'une région, de manière à répondre de façon efficace aux besoins de la population. Cette répartition doit aussi être équitable, losque considérée sur une certaine période de temps. Pour résoudre ce problème, nous utilisons la génération de colonnes et la programmation par contraintes.

ABSTRACT

Combinatorial optimization problems come in a plethora of flavors. In a realistic setting, solutions to many of these problems require some sort of balance or fairness. Yet, there has been little research specifically on the concept of fairness and balance in the context of combinatorial optimization. Furthermore, it can be observed that when a problem includes a fairness component, that component is generally not constructed very carefully. This thesis studies and compares various ways of achieving balance in constraint programming and in integer programming, and analyzes the characteristics and properties of various balancing methods.

In the first part of this thesis, we study the *quadratic multiknapsack problem* with conflicts and balance constraints, that is, packing items of different weights and with pairwise preferences and conflicts into knapsacks, while satisfying constraints on the balance of the loads of those knapsacks. We present and compare constraint programming and integer programming models, including a model based on column generation and branch and price. This allows us to constrast these models by the difficulty of their implementation, and their resulting performance. The problem studied finds an application in event seating.

The second part of this thesis focuses on methodology, and is concerned with the characteristics associated with various ways of achieving balance. We present four different methods of finding balanced solutions in problems, and discuss some of their properties. We further present three cases showing that the way balance is achieved in some existing problems can be improved. These experiments allow us to make a list of general modeling guidelines, which should help a modeler decide on how to best integrate fairness into a problem.

For the last part of this thesis, we study a special problem where fairness is considered over a period of time. We show some theoretical proofs for special cases of this problem, allowing us, for instance, to know what length of time is needed to ensure a perfectly fair solution. We then introduce a general framework which allows us to tackle more complicated cases of this problem. We use this framework to solve the *ambulance allocation problem*: A number of ambulances must be allocated to some zones in a region, such that the population of the region is efficiently covered by the ambulance service. This coverage should also be fair, when considered over a given time horizon. To solve this problem, we introduce a model based on column generation and constraint programming.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

AWT	Ambulance-Without-Transition
BACP	Balanced Academic Curriculum Problem
BBSS	Balancing Bike Sharing Systems problem
CCG	Compact Configuration Graph
CG	Configuration Graph
CMP	Continuous Relaxation of the Master Problem
CP	Constraint Programming
ILP	Integer Linear Programming
IP	Integer Programming
LNS	Large Neighborhood Search
MILP	Mixed-Integer Linear Program
MINLP	Mixed-Integer Nonlinear Programming
MP	Master Problem
MT	Maximum Transition
NPAP	Nurse-Patient Assignment Problem
PP	Pricing Problem
QMKP	Quadratic Multiknapsack Problem
RMP	Restricted Master Problem
SPFA	Single-Period Fair Allocation
T-PFA	T-Period Fair Allocation
WSP	Wedding Seating Problem

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Combinatorial problems are notoriously difficult to solve efficiently. Yet, a variety of practical applications intimately related to these problems abound in industry. The main characteristic of these problems which is of interest to us is the aspect of fairness. There is a multitude of strategies to tackle the fairness facet of a combinatorial problem. Each strategy exhibits an assortment of advantages and drawbacks with regard to the properties of the solution. In this thesis, we design and compare new ways of finding fair and balanced solutions to various types of problems.

This chapter will provide the reader with the necessary knowledge to appreciate the remainder of this thesis. We begin by introducing fairness, which is not as plain a concept as one might assume. We follow by discussing, with examples, the basics of integer programming and constraint programming, two optimization methods which are used extensively in Chapters 3 to 5. We end this chapter by presenting the objectives of this research.

1.1 Fairness[1]

"And also Lot, who went with Abram, had flocks and cattle and tents. And the land did not bear them to dwell together, for their possessions were many, and they could not dwell together. And there was a quarrel between the herdsmen of Abram's cattle and between the herdsmen of Lot's cattle, and the Canaanites and the Perizzites were then dwelling in the land. And Abram said to Lot, 'Please let there be no quarrel between me and between you and between my herdsmen and between your herdsmen, for we are kinsmen. Is not all the land before you? Please part from me; if you go left, I will go right, and if you go right, I will go left.' "

– Genesis 13:5-9 (Torah)

The mythic story of Abraham, passed down in writing millenia ago, is central to Jewish, Christian, and Muslim traditions. In this episode, the resources of the land are insufficient to sustain the herds of both Abram and Lot. They are thus forced to part ways. Abram divides the land in two parts, and offers Lot the first choice: This is the simple *I cut, you choose* method of achieving fairness—a method which is still under active scientific study today [1].

Questions relating to fairness do not merely stand in the realm of cold, stolid scientific study: They are intrinsic to the human condition. We are constantly reminded of the ongoing struggle of black Americans, which is one for fairness. In another context, one's survival may hinge on a fair distribution of sparse resources, such as food, during a period of crisis: Wartime rationing, for example, ensures fairness in such trying times. Or perhaps one's well-being depends on access to medicine which is in short supply, as we have seen in the rush for vaccines during the recent COVID-19 pandemic. Finally, we need not look very far to find that historically, a lack of fairness often breeds unrest and triggers revolutions.

Now that we have established a *raison d'être* for fairness, this begs the question: "What does it mean to make things fair?" One may quickly conclude that fairness amounts to *making things as equal as possible*. While this seems entirely reasonable—and it is, on the surface—it is not the be-all and end-all of fairness. Because making things equal depends on one's opinion of *what* should be equal. Consider the issue of taxation, for instance. Some people contend that a flat-rate tax is fair, as everyone pays the same rate, regardless of income. Others advocate for a progressive tax rate wherein people with more income pay proportionally more taxes—here, fairness is understood as one's ability to pay taxes

[1] In this thesis, we will use the terms *fairness* and *balance* interchangeably, for reasons explained in Section 4.2.

with regard to their income. Or consider that some patients, after waiting in the emergency room of a hospital for several hours, might make the case that fairness for medical services should be on a first come, first served basis. Yet, most people (including these same patients, under different circumstances!), would on the contrary argue that the fair approach would be to first take care of those most in need, and would support the concept of triage.

The very nature of fairness, so it seems, is rooted in opinion. One man's *giving everyone the same thing* will be another man's *giving different people different things*. Further complications arise in the cases where those things that must be fairly divided among interested parties are heterogeneous, or if they hold different values to different parties, and so on. If we dig too deep in this direction we will reach the realm of game theory. While we will touch lightly on this subject later on, it is not among the main objectives of the present thesis.

Now that we have instilled some curiosity in the reader's mind as to the nature of fairness, let us entertain the simplest idea that fairness means to *make homogeneous things as objectively equal as possible*.[2] Unless we manage to make things perfectly equal, i.e., giving everyone exactly the same thing, we are faced with a new question which we must suitably answer: "How do we measure unfairness?" Because if we are in the business of making things perfectly equal, and that we are unable to achieve this, then, certainly, we will want to make things as least unequal as we possibly can. We need not scratch the surface with much vigor to realize that this is yet another question without a definitive answer. Let us illustrate this by solving the problem of fairly distributing candies to trick-or-treating children. Suppose that we have candy bags of assorted sizes which we want to hand out to six children. The contents of the bags total 60 candies, but the assortment of bag sizes makes impossible the task of handing each child an exact share of 10 candies. There are, then, many options to distribute the candy bags fairly among the children.

[2] In a fictional world where everyone agrees on what should be fair.

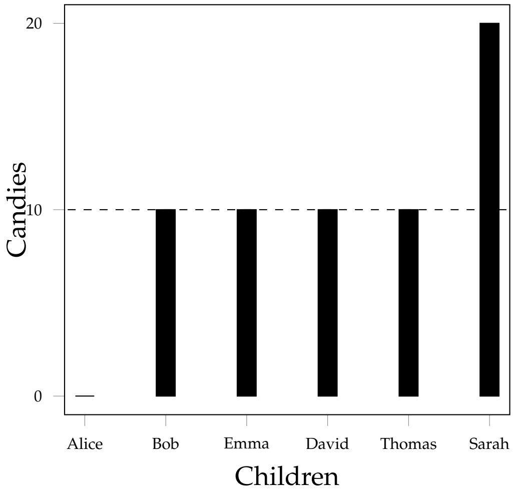


Figure 1.1 Most shares total exactly 10 candies.

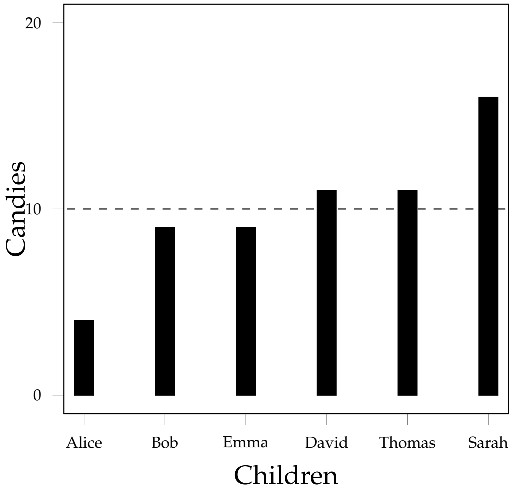


Figure 1.2 The sum of absolute or squared candy deviations from the mean is minimal.

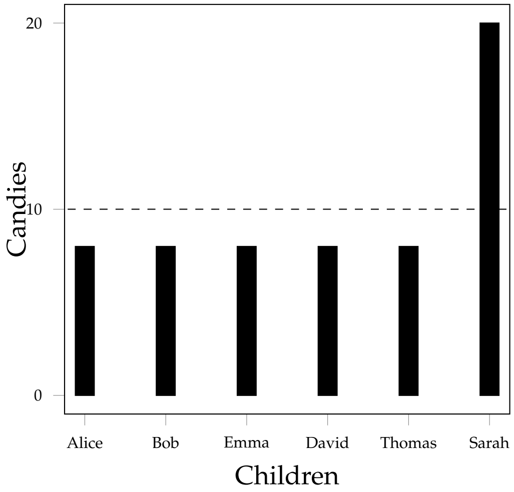


Figure 1.3 The smallest share of candy is maximal.

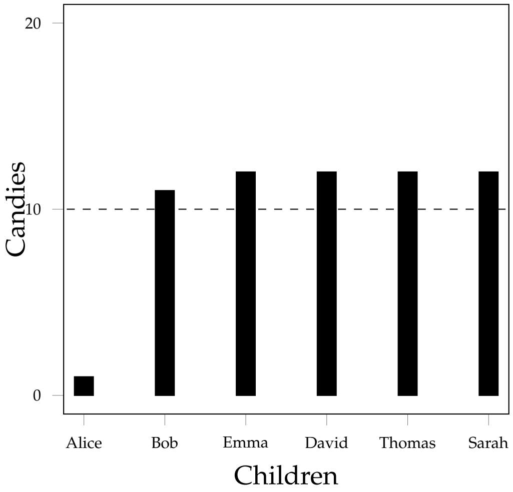


Figure 1.4 The largest share of candy is minimal.

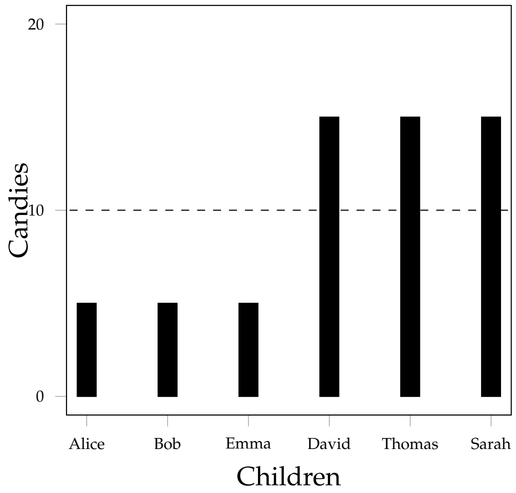


Figure 1.5 The candy disparity between the most extreme share and the mean is minimal.

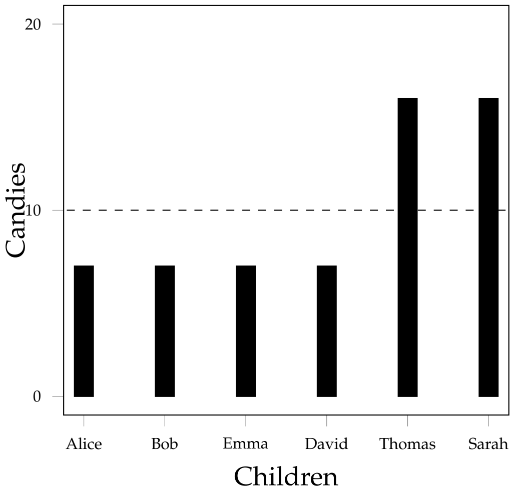


Figure 1.6 The candy disparity between the smallest and largest shares is minimal.

The first solution is perfectly fair to most children. The downside is that Alice has no candy while Sarah has twice as much candy as her friends (Figure 1.1). The second solution may initially appear fairer, but notice that no child is given an exact share of 10 candies and that Sarah has four times more candy than Alice (Figure 1.2). In the next solution, maximizing the smallest share of candy ensures that everyone gets at least some, but Sarah still has much more than the others (Figure 1.3).[3] In contrast, minimizing the largest share of candy mitigates the candy disparity between Sarah and the other children, at the cost of Alice getting almost nothing (Figure 1.4). Minimizing the candy disparity between the most extreme share and the mean does not necessarily ensure an adequate candy distribution as we can see that one half of the children have thrice as many candies as the other half (Figure 1.5). Minimizing the candy disparity between the smallest and largest shares may also have a similar effect (Figure 1.6). Looking back on these solutions, we can reasonably make a case that not one of these outcomes is intrinsically fairer than the others. We conclude that, even when considering the simplest understanding of fairness, we do not immediadely discern a clear path to this end.

The intent of this section was not to formally define fairness, but to show the multiple levels of nuances attached to it. This thesis is mainly concerned with the second question which we posed earlier, that is, how to deal with unfairness in the context of combinatorial optimization.

[3] By now, Sarah's friends must think that she's cheating at this whole candy thing!

1.2 Integer Programming[4]

Mathematical programming involves assigning real values to a set of variables, while satisfying a set of constraints and optimizing an objective. When these constraints and the objective are linear, this type of optimization is called linear programming (LP). LP gradually developed from Fourier's method for solving systems of linear inequalities, nearly two centuries ago.

1.2.1 Linear Programming

A linear program is of the form

$$\begin{aligned} \min \quad & c^{\mathrm{T}} x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0. \end{aligned}$$

The objective, in this case, is to minimize the scalar product of vectors of variables $x \in \mathbb{R}^n$ and their coefficients $c^{\mathrm{T}} \in \mathbb{R}^n$. This objective is *subject to* constraints $Ax \geq b$ with $A \in \mathbb{R}^{m \times n}$ a matrix of coefficients and $b \in \mathbb{R}^m$ a vector of values, and to constraints $x \geq 0$.

If there exists some combination of assignments for x which do not violate the constraints, the problem is said to be *feasible* and the value of this solution is $z = c^{\mathrm{T}} x$; Otherwise, the problem is *infeasible*. Since the objective is a minimization one, an assignment of values for x is said to be *optimal*[5,6] if no other assignment results in a strictly lower z, that is, $c^{\mathrm{T}} x^* = z^* \iff c^{\mathrm{T}} x^* \leq c^{\mathrm{T}} x, \forall x$.

Linear programs are most often solved using the *simplex* algorithm—the cornerstone of LP—which was introduced by Dantzig in 1947 [2]. This algorithm can solve substantially large programs, and is very efficient in practice.

A fundamental element of LP is the theory of *duality*. Every linear program (the *primal*) has an alternate and equivalent formulation called the *dual*. The translation of one problem into the other involves transforming variables into constraints and constraints into variables, and reversing the objective function. The dual formulation of the primal problem presented previously is

[4] Part of this section is inspired by the MTH6403 (Polytechnique Montréal) course notes of Charles Audet.

[5] There may be multiple optimal solutions.

[6] An optimal assignment is noted x^*, and its associated objective value is noted z^*.

$$
\begin{aligned}
\max \quad & b^{\mathrm{T}} y \\
\text{s.t.} \quad & A^{\mathrm{T}} y \leq c \\
& y \geq 0.
\end{aligned}
$$

Duality is the principle which defines the relation between the formulation of the primal and that of the dual. The *weak duality theorem* states that when minimizing (in the primal), any feasible solution of the dual problem is a lower bound for the primal problem. The *strong duality theorem* states that if there exist feasible solutions to both the primal and dual problems, then there also exist feasible solutions for both problems for which the objective values coincide, and that those solutions are optimal.

Constraints which actively restrict the optimal solution (i.e., for which alteration of the right-hand side of the inequality would change the optimal solution) are said to be *binding*. In contrast, constraints which have no effect on the optimal solution are *non-binding*. A *dual value* is associated with each constraint of a problem, and represents a rate of change in the objective, or the improvement of the value of the objective if that constraint were relaxed by one unit. In the minimization problem presented here, the dual value of constraint i represents the improvement in the objective achieved by relaxing that constraint to $\sum_{j=1}^{n} a_{ij} x_j \geq b_i - 1$. Note that since the relaxation of non-binding constraints does not affect the objective, their dual values are zero.

Each inequality constraint of a problem has an associated *slack* variable. The value of a slack variable represents the amount by which the right-hand side of the inequality would have to be increased in order for that constraint to become binding. The slack of constraint i is given by $\sum_{j=1}^{n} a_{ij} x_j - b_i$. A constraint which is binding has a slack of zero. A *reduced cost* is associated with each variable of the problem. This value represents the amount by which the coefficient of the associated variable in the objective function would have to improve (i.e., decrease, if we are minimizing) before it would be profitable for this variable to assume a nonzero value. The reduced cost $\bar{c}_j$ of variable x_j is given by $c_j - \sum_{i=1}^{m} a_{ij} y_i$.

1.2.2 Example

To put these concepts into perspective, we will consider as an example a simple diet problem. We want to devise a healthy diet of bacon, fries, and pie. With each food are associated various nutrition values and a cost (Table 1.1).

Table 1.1 The path toward ~~heart disease~~ affordable and healthy eating habits starts here.

Food	Energy (kcal)	Protein (g)	Cost ($)
Bacon	500	36	10
Fries	300	3	5
Pie	250	2	3

Various guidelines suggest a daily intake of about 2000 calories and 60 grams of protein for an average adult. Furthermore, due to the author's propension to eat bacon, we will insist on the consumption of at least one portion of it. Considering that last spring's tax return was not as generous as we had anticipated, we'll want to minimize the cost of such a diet in order to have enough money left over to cover our mortgage payments. We can model this problem as

$$\begin{aligned}
\min \quad & 10x_1 + 5x_2 + 3x_3 && (1.1)\\
\text{s.t.} \quad & 500x_1 + 300x_2 + 250x_3 \geq 2000 && (1.2)\\
& 36x_1 + 3x_2 + 2x_3 \geq 60 && (1.3)\\
& x_1 \geq 1 && (1.4)\\
& x \geq 0 && (1.5)
\end{aligned}$$

with x_1 the portions of bacon, x_2 the portions of fries, and x_3 the portions of pie, which need to be purchased. Constraints (1.2) and (1.3) ensure that we consume enough calories and protein. We are assured of at least one portion of bacon (1.4), and we want to minimize the price of this diet (1.1). Finally, we can obviously not consume a negative amount of food (1.5). The optimal solution of this problem is $x^* = (\frac{11}{8}, 0, \frac{21}{4})$ which yields an objective value of $z^* = \frac{59}{2}$. This means that the cheapest way we can meet our daily intake of nutrients using these three foods would be to consume 1.375 portions of bacon and 5.25 portions of pie, which would amount to a cost of $29.50. The dual associated with (1.1) to (1.5) is

$$\begin{aligned}
\max \quad & 2000y_1 + 60y_2 + y_3 && (1.6)\\
\text{s.t.} \quad & 500y_1 + 36y_2 + y_3 \leq 10 && (1.7)\\
& 300y_1 + 3y_2 \leq 5 && (1.8)
\end{aligned}$$

$$250y_1 + 2y_2 \leq 3 \tag{1.9}$$
$$y \geq 0. \tag{1.10}$$

Reduced Costs. Associated with x^* are the reduced costs $(0, \frac{53}{40}, 0)$, meaning that if the coefficient of x_2 in the objective function was lowered from 5 to $5 - \frac{53}{40} = \frac{147}{40}$, the value of x_2^* in the optimal solution would be nonzero. In other words, $5 for a portion of fries is too expensive for us, as we can get our nutrients more cheaply from the current prices of bacon and pie. But, were the grocery store to have a special deal reducing the price of fries by at least $\frac{53}{40} \approx \$1.33$, it would then make sense for us to include fries in our diet.

Dual Values. The dual values associated with constraints (1.2) to (1.5) are $y^* = (\frac{1}{90}, \frac{1}{8}, 0, 0, 0, 0)$, which we can confirm are also optimal since $2000y_1 + 60y_2 + y_3 + 0y_4 + 0y_5 + 0y_6 = \frac{59}{2} = z^*$.[7] The dual value $y_2^* = \frac{1}{8}$ implies that constraint (1.3) is binding (and thus has a slack of 0) and that, were it relaxed to $36x_1 + 3x_2 + 2x_3 \geq 60 - 1$, the objective value of the optimal solution would improve from $\frac{59}{2}$ to $\frac{59}{2} - \frac{1}{8} = \frac{235}{8}$. Put it another way, if the daily protein intake were 59g instead of 60g, the price of the diet would be about $\frac{1}{8} \approx \$0.13$ cheaper.

Slack. The non-binding constraint (1.4), on the other hand, has a dual value of 0 but a slack value of $\frac{3}{8}$, meaning that this constraint would become binding (forcing its dual value to be nonzero) if it were tightened to $x_1 \geq 1 + \frac{3}{8}$. Bacon-wise, this means that even though we insist on having at least one portion of bacon in our diet, this requirement is irrelevant since our cheapest diet option already includes a little bit more than one portion of it, simply because it is the cheapest way to get those nutrients. If we had an even higher opinion of bacon and decided to insist on eating more than $1 + \frac{3}{8} = \frac{11}{8}$ portions of it (the current bound plus the slack), constraint (1.4) would become binding, and its associated dual value would become nonzero.

1.2.3 Integer Linear Programming

The diet problem of the previous section, as well as a wide range of other problems, are trivial to solve to optimality with the simplex algorithm. Integer programming (IP) is a complex field evolved from mathematical programming in which the set of variables must be assigned integer values. The diet problem, a linear program, would become an integer linear program (ILP) by replacing constraints $x \geq 0$ with $x \in \mathbb{Z}_{\geq 0}$. We can easily observe the convenience of integer solutions in practical contexts: We can more readily find one

[7] Recall the *strong duality theorem* mentioned in the previous section.

pound of prepackaged bacon than $\frac{11}{8}$ pounds of it, it would be peculiar for a recruiter to post a job offer asking for two-thirds of an employee working full-time, and so on.

These integrality constraints are the characterizing complexity of IP. We notice that relaxing the integrality requirements of a program involves nothing more than removing the associated constraints. It follows logically that we cannot worsen the solution of a problem by *removing* constraints. Hence, every complex ILP has an associated simple LP which provides a bound for its more restricted version. The bound provided by the LP, when compared with a feasible solution (not necessarily optimal) of the ILP, allows us to gauge the potential quality of that solution: This is called the *gap*.

1.2.4 Related Fields

Integer programming is a vast field, which extends past ILP. Techniques exist to solve problems where the constraints or objective are nonlinear, for example. Also, certain properties of a problem may allow it to be reformulated in a special way. This is the case for column generation, of which we make a passing reference, since this technique is used in Chapters 3 and 5. The column generation algorithm is a powerful tool to solve problems which exhibit a special structure. We find this structure, for instance, in the bin packing problem [3]. Column generation considers only a small fraction of the variables, and generates more as needed. Integrality constraints are satisfied via branch and price, a combination of branch and bound and column generation. A description of column generation and branch and price is provided in Section 3.6; The reader may consult [4] for a fuller explanation of the column generation algorithm.

1.3 Constraint Programming[8]

Constraint programming (CP) is a programming paradigm for solving combinatorial problems. It is a form of declarative programming wherein an abstract model is constructed and then handled by a solver. CP traces its roots back to Ivan Sutherland's PhD thesis in the early 1960s. About a decade later, the introduction of the logic programming language Prolog marked the beginning of what we know today as constraint programming [5].

For a novice, CP is probably best introduced by drawing parallels with the popular game of *Sudoku*. The reader unfamiliar with Sudoku is urged to read the rules of the game below, and to complete the sample grid which can be found in Figure 1.7. This simple exercise will prove invaluable to understand the remainder of this section.

At the beginning of the game, the correct digits to be assigned to the empty cells are yet unknown, but we know that they must take an integer value from 1 to 9. In CP, these empty cells are the *variables* of the problem, and the sets of values that these variables can assume are their *domains*. For instance, a variable x_{ij} represents the cell found at the intersection of row i and column j, and its domain would be $\{1,\ldots,9\}$, i.e., $x_{ij} \in \{1,\ldots,9\}$.

The rules of Sudoku are simple: The digits of each row, column, and 3×3 block should be different. In CP, these *rules* are called *constraints*. The constraint of the third row of the grid, for instance, could be expressed as

$$x_{31} \neq x_{32} \land x_{31} \neq x_{33} \land \cdots \land x_{38} \neq x_{39}.$$

Yet, CP offers a much more elegant way of expressing the above constraint, that is

$$\texttt{alldifferent}(x_{3*}).$$

These named constraints are known as *global constraints*, and they can encapsulate complex structures. In the case of `alldifferent`, this constraint ensures that a set of variables (in our case, the third row x_{3*} of the grid) assume different values. But the power of global constraints is still more extensive, and can represent any imaginable relationship between a set of variables. In fact, whole NP-hard problems can be embedded into a single con-

[8] Part of this section is inspired by the INF6101 (Polytechnique Montréal) course notes of Gilles Pesant. The reader interested in an in-depth reference for all things constraint programming is directed towards *The Handbook of Constraint Programming*, by Francesca Rossi, Peter van Beek, and Toby Walsh (Elsevier Science, 2006).

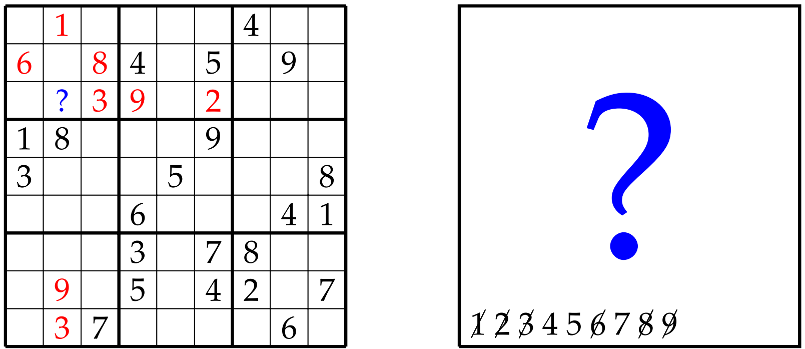
straint, such as the `binpacking` constraint, for example.

In more formal terms, a *constraint satisfaction problem* (CSP) is defined as follows: Let problem $\mathcal{P}$ be a triple such that $\mathcal{P} = \langle X, D, C \rangle$, where

- $X = \langle x_1, \dots, x_n \rangle$ is a tuple of variables,
- $D = \langle D_1, \dots, D_n \rangle$ is a tuple of domains such that $x_i \in D_i$,
- $C = \langle C_1, \dots, C_m \rangle$ is a tuple of constraints. A constraint C_j defines the relation between a subset of variables, restricting certain variable-value combinations.

A solution to $\mathcal{P}$ is a tuple $A = \langle a_1, \dots, a_n \rangle$ where $a_i \in D_i$ and such that all constraints $C_j \in C$ are satisfied. If at least one such solution exists then $\mathcal{P}$ is said to be *consistent*; otherwise it is *inconsistent*. To complement this basic definition, a CSP is solved by a *solver* which uses two major components to achieve its goal: *inference* and *search*.

In a game of Sudoku, when a player is not successful at logically inferring which digit should be assigned to a given cell, he will, however, know which digits *cannot* be assigned to that cell. A common habit is to indicate, in a corner of the empty cell, which digits still retain the possibility of being valid choices for that cell. As the game progresses and new information is revealed, a player will cross out those small digits of the empty cells when they become inconsistent with some other cells of the grid, until at last a single possibility remains. In CP, this is called *filtering*, and is illustrated in Figure 1.7.



	1					4		
6		8	4		5		9	
	?	3	9		2			
1	8				9			
3				5				8
			6				4	1
			3		7	8		
	9		5		4	2		7
	3	7					6	

A cell and its conflicts.

Filtering the domain of a cell.

Figure 1.7 An initial grid of Sudoku. We are interested in the blue cell, whose value depends on the red digits.

The process of a constraint filtering values from the domains of its associated variables is illustrated in Figure 1.8. We are interested in the third row, and in particular in the green, blue, and pink cells, whose domains (in the initial grid) were $\{4,5,7\}$, $\{4,5,7\}$, and $\{5,6\}$. By partly filling the grid with the orange digits, those previously-mentioned domains are filtered to $\{4\}$ (by the constraints of the first column and of the top-left block), $\{4,5\}$ (by the constraints of the second column and of the top-left block), and $\{5,6\}$. Let us now look at the constraint which is of interest to us, that of the third row. A single value remains in the green domain, and that cell is thus fixed to 4. This, in turn, triggers more filtering for the blue cell, whose domain becomes $\{5\}$. The same logic applies to the pink cell, whose domain is filtered down to a single value. The `alldifferent` constraint of the third row is now in a state of *local consistency*, meaning that no more values can be filtered from the domains of its variables.

	1					4		
6	7	8	4		5		9	
?	?	3	9	?	2	?	?	?
1	8				9			
3				5				8
			6				4	1
5			3		7	8		
	9		5		4	2		7
	3	7					6	

A partially-filled grid of Sudoku.

	1					4		
6	7	8	4		5		9	
4	5	3	9	?	2	?	?	6
1	8				9			
3				5				8
			6				4	1
5			3		7	8		
	9		5		4	2		7
	3	7					6	

Local consistency is achieved.

Figure 1.8 (Left) A partially-filled grid of Sudoku. We are interested in the constraint of the third row, in particular in these three cells. (Right) Logical inference results in filtering, and ultimately, local consistency for this constraint.

The attentive reader will notice that a state of local consistency for a constraint can be disturbed by the introduction of new information, i.e., by the filtering accomplished by other constraints. Whether such filtering results in a variable being fixed or not is of no matter: A single value filtered by a single constraint could trigger a cascade of values being filtered by many constraints in the ensuing filtering steps. This process is known as *constraint propagation*, and since the domains of the variables are finite, it is bound to finish at some point.

Let us digress for a moment, in order to be more detailed on the nature of *consistency*. In our Sudoku example, any value in the domain of a variable is killed off as soon as it falls victim to some inconsistency. This is called *domain consistency*, and while it is the strictest form of consistency, there exist others. *Bounds consistency*, for instance, only ensures that the highest and lowest values of the domain are consistent, and assumes that any value in-between is also consistent.[9] One may wonder about the motivation of using a looser level of consistency—the explanation for this is simple. The process of filtering is achieved via filtering algorithms, otherwise known as a *propagators*. For a particular constraint, various algorithms which are more or less computationally expensive can reach consistency levels which are more or less strict. In practice, some problems may exhibit a certain structure where bounds consistency is enough, or where the additional filtering provided by domain consistency is not enough to offset its computational cost. This may motivate the modeler into using one or another filtering algorithm.

There may come a point in a game of Sudoku where no more logical inference can be made as to which potential values can be safely crossed out in the remaining cells.[10] At this point, the player is forced to make an educated guess, and picks one cell where they fix the value, to see where the resulting logical inferences will lead them. If this results in an impossible situation later on (a cell with no valid values remaining), the player goes back to the initial problematic state, and makes another guess. In CP, picking a value that is not chosen by logical inference and seeing where it leads is known as *branching*. If this branching leads to an impossible situation later on (known as infeasibility in CP), the process of going back to the initial state is called *backtracking*.

In more technical terms, when the CP solver branches, this expands the *search tree* by creating two subproblems: one in which a variable is fixed with a value, and the other where that value is removed from the domain of the variable. At each node of the search tree, local consistency is achieved for all constraints. If this results in an empty domain for one of the variables, then the node cannot lead to any valid solution—it can be safely discarded and part of the search space is thus pruned. In such a case, the search backtracks to an earlier valid node and continues from there. The tree is fully explored once every node has led to a valid or inconsistent solution: Optimality or infeasibility is then proven.

We will soon see, starting in the next chapter, that CP includes several global constraints related to balance.

[9] Assuming that the domain values belong to some ordered set.

[10] This situation, however, arises in the case of poorly-designed Sudokus. In general, Sudokus should be solved using only logical deductions.

1.4 Purpose of this Thesis

The arguments presented in the introduction on fairness of Section 1.1 should have convinced the reader of its importance for some types of problems. Besides, a cursory review of papers offering solutions to combinatorial problems which include some kind of fairness requirement shows that, by and large, the type of fairness tends to be chosen impetuously. We believe that value can be gained in the study of fairness in the context of combinatorial optimization. For combinatorial problems with fairness requirements, this would translate into an improvement in the quality of solutions, as well as alleviated modeling efforts.

Our aim is to find and compare ways of implementing fairness in constraint programming and integer programming, and to investigate the properties of various types of fairnesses. With respect to fairness and exact optimization methods, our intention is essentially to fill the following gaps in the literature:

- The absence of rigorous comparisons in implementation effort and in performance, between CP and IP models which include fairness;
- An often unclear understanding of the inherent properties of various types of fairnesses;
- A lack of general guidelines on how to tackle models requiring fairness.

Addressing these challenges would improve the means of intelligently including fairness constraints and objectives in CP and IP models.

1.4.1 Research Objectives

This thesis studies balance in the context of combinatorial optimization, and aims to be a resource to model problems from CP and IP standpoints. In particular, a modeler:

- Will be aware of the intrinsic balance characteristics of the various balancing strategies, allowing for more informed choices;
- Will be informed on ways to integrate these strategies into models, easing the modeling effort;
- Will be forewarned of the complexity of such strategies, whether in implementation or in computational requirements.

1.4.2 Thesis Organization

This thesis is organized as follows:

Chapter 2 We begin with a review of the literature, which includes fairness in general, particulars in the CP and IP frameworks, and some concrete applications.

Chapter 3 The first article[11] explores how various measures of unfairness can be integrated into CP and IP models. This is studied through the *quadratic multiknapsack problem*, which somewhat resembles a *bin packing problem*. Here, we are mostly concerned with the computational aspects associated with the combination of an optimization method and a measure of unfairness.

Chapter 4 In the second article,[12] the focus is shifted to the properties of the measures of unfairness, and how these are often overlooked. We study three existing problems where the initial way chosen to achieve balance was flawed. We focus on methodology, offering some guidelines on how to choose one measure of unfairness or another.

Chapter 5 The last article[13] investigates fairness in a particular framework, that is, the fair allocation of resources over time. We present some theoretical results, after which we solve the *ambulance location problem* with a model combining IP and CP components.

Chapters 6 and 7 Finally, we discuss our results and their limitations, and conclude by considering avenues for future work.

[11] This chapter has been published as *The Quadratic Multiknapsack Problem with Conflicts and Balance Constraints* in *INFORMS Journal on Computing* on October 14, 2020.

[12] This chapter has been submitted for review as *Measures of Balance in Combinatorial Optimization* to *4OR* on June 29, 2020.

[13] This chapter has been submitted for review as *Fairness over Time in Dynamic Resource Allocation with an Application in Healthcare* to *Mathematical Programming* on January 10, 2021.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we break down the literature review into a structure resembling that of the introduction. We first present some works which argue for the importance of fairness, and which discuss various ways of achieving it. We then show the main global constraints associated with balance in CP. Finally, we cover how balance is achieved in IP.

The reader should take note that many papers presented in this chapter overlap with the literature reviews of Chapters 3 to 5. We will discuss some of them in relative detail; Others will be mentioned in passing, with a note referring the reader to the appropriate upcoming chapters.

2.1 Fairness

Before delving into details of the literature related to fairness, we will begin by presenting two contemporary problems where fairness plays a major role. We need not be exhaustive here, as these two problems should suffice to validate the claims that we made in Section 1.1, with respect to the critical importance of fairness for some applications.

In the American judicial system, a pretrial release is the mechanism by which a defendant is released back into the community while awaiting trial. One may be surprised to hear that in some jurisdictions of the United States, the decision of whether or not to grant a pretrial release to a defendant falls partly unto an algorithm. Research suggests that many such algorithms may make decisions which appear biased with respect to the race of the defendent [6]. To ensure equality of treatment, and a continuing public trust in the judicial system, it is therefore very important to ensure that such algorithms are as fair as possible [7].

In the United States, congressional apportionment is the mechanism by which a fixed number of seats in the House of Representatives are distributed among the states according to their populations. Since there is a great variation in the populations of the 50 states, and that there are currently only some 400-odd seats, it is not possible to have each seat represent exactly the same number of people. The problem then becomes one of allocating the seats as fairly as possible. Models aiming for fairness in the apportionment of the U.S. House of Representatives have been studied since 1792 [8]. This problem is a tricky one: Early improvements [9] to the initial method chosen to achieve fairness would sometimes result in an unexpected outcome, called the *Alabama paradox*.[1] Further developments were achieved over the years [11]. Given that democracy is a cornerstone of most modern countries [12], it is critical to ensure fairness in the democratic process.

Constructing a model for a problem such that its solutions exhibit *some* fairness is generally not a complicated task. Fairness in facility location, for instance, has been studied at least since the 1960s [13]. Marsh and Schilling pity that while various models present fair solutions to these types of problems, "there has been little agreement in these models as to how equity should be measured" [14]. The authors proceed to thoroughly analyze 20 measures of equity which they have identified in the literature (some of which are equivalent to others). Then, similarly to what we have done in Chapter 4, they provide seven criteria which they suggest to be considered when choosing an equity measure. Among these criteria are some technical ones: For example, they note that some measures of equity may be too hard to compute for some problem sizes, and that it should be considered if optimality is required over approximation. More interestingly, though, they note that some non-technical criteria should also be considered. For instance, an equity measure should be properly understood by the decision maker, as it is them who, in the end, will have to interpret it or explain it to others.

In his unpublished PhD dissertation, some extracts of which can be found in the appendix of a subsequent paper [15], Kumar surveys over 20 measures of imbalance found in the literature, and shows that some are equivalent to others, or dominated by them. In a follow-up paper, Kumar and Shanker study fairness in the context of flexible manufacturing systems, comparing various methods of balancing workload differences between machines [15]. They gauge the balancing objectives by what they call *efficacy*, or the number of balancing objectives one can dominate. Their conclusions are not as nuanced as those of Marsh and Schilling: They contend that one balancing objective (minimizing the average pairwise difference between machine workloads) is, in their words, "the best objective for workload balancing" (for this problem).

More recently, Bektaş and Letchford discuss fairness in the context of combinatorial optimization [16]. The authors mention diverse fields, among which economics, philosophy, and psychology, in which the implications of fairness have been researched. They cite several surveys on the topic, in the areas of facility location, vehicle routing, resource allocation, nurse rostering, and several others. They present some existing approaches to

[1] C. W. Seaton, Chief Clerk of the Census Office, reports to Congress in 1881: "While making these calculations, I met with the so-called 'Alabama' paradox, where Alabama was allotted 8 Representatives out of a total of 299, receiving but 7 when the total became 300." [10]

fairness, but mainly concentrate on L_p-norms.[2]

Some readers may feel that the present section, reviewing some of the literature related to fairness, is lacking. Fairness, indeed, covers a lot of ground. In machine learning, fairness might mean the removal of discrimination for people belonging to protected groups [17], or ensuring that similar individuals are treated similarly [18]. Fairness may also be considered in the context of multi-objective optimization. In some kidney allocation problems, for instance, utility (life year gains) and equity (fair access to organs) should be maximized [19]. The present thesis is concerned with a narrower slice of this field. For readers coming from a broader fairness background, this thesis mainly considers what is generally understood as *balance*, e.g., the equitable distribution of resources. The line between *balance* and *fairness* may thus seem blurred in this thesis—some details on why this is can be found in Section 4.2.

Some combinatorial problems involving fairness in the context of CP and IP have been studied in this thesis. To minimize the redundancy of citations, we refer the reader to the appropriate sections of this thesis. For the quadratic multiknapsack problem with balance constraints, and associated event seating problems, we refer the reader to Chapter 3. For the balanced academic curiculum problem, the nurse-patient assignment problem, the balancing bike sharing systems problem, and a short overview on social welfare functions, see Chapter 4. For fairness in resource allocation problems in general, and in the ambulance location and relocation problem in particular, the reader is referred to Chapter 5.

2.2 Constraint Programming

Some methods for achieving fairness have already been set forth in the candy distribution example of Section 1.1. We continue in this direction by introducing the notion of L_p-norm, which is requisite to correctly grasp the remainder of this section and chapter. Given a collection X of n points, the global distance between these points and their mean μ, according to a norm p, is defined by

$$\sum_{i=1}^{n} |x_i - \mu|^p.$$

In particular:

- L_0-norm: Minimize the number of nonzero deviations from the mean.

[2] The reader may refer to Section 2.2 for a definition of L_p-norm, which is functionally similar to what is understood here by these authors.

- L_1-norm (least absolute deviations): Minimize the sum of absolute deviations from the mean.
- L_2-norm (least squared deviations): Minimize the sum of squared deviations from the mean.
- L_∞-norm: Minimize the maximum deviation from the mean.

In CP, several global constraints implement these norms. The first attempt at constructing a constraint specifically for the purpose of balance took place with the `spread` constraint [20], which implements a bounds consistent[3] filtering algorithm for the L_2-norm. The general version of this constraint is of the form `spread`$(X, \mu, \sigma, \widetilde{x})$ with X a set of variables, μ the mean, σ the standard deviation, and $\widetilde{x}$ the median, all of which are continuous intervals. Simpler filtering algorithms were soon introduced for an unbounded median [21]. A more efficient filtering algorithm providing stricter bounds consistency (and requiring a constant mean and an unbounded median) was later developed [22].

The `deviation` constraint for the L_1-norm was introduced soon after [23, 24]. This constraint is of the form `deviation`(X, m, D) with X a set of variables, m a fixed mean, and D the sum of absolute deviations of X from m as a continuous interval. Its filtering algorithm is simpler and faster than that of the `spread` constraint, but it still only achieves bounds consistency.

The first generalization of these two constraints was accomplished more recently, with a propagator achieving bounds consistency for pairs of `sum` constraints [25,26]. Shortly after, the `dispersion` constraint was introduced [27], which not only generalizes the `spread` and `deviation` constraints, but which can also handle all L_p-norms with the exception of the L_0- and L_∞-norms.[4] This constraint is of the form `dispersion`(X, μ, Δ, p) with X a set of variables, μ the mean as a continuous interval, Δ the L_p-deviation as a continuous interval, and p the norm. Using a filtering algorithm based on dynamic programming, this constraint achieves domain consistency for a fixed mean[5] and an unbounded median.

Other "classical" constraints can be used to express simpler methods of achieving fairness. For instance, the `minimum`(v, X) constraint,[6] with X a set of variables and v a variable indicating the minimum value in this set, can be used to maximize the minimum value

[3] The reader will recall that *bounds consistency* only ensures that the smallest and largest values of the domains are consistent, while *domain consistency* ensures that all values of the domains are consistent.

[4] It could be extended to include them, however.

[5] It could also be extended to include a variable mean.

[6] See the Global Constraint Catalog: http://sofdem.github.io/gccat.

found in X by maximizing v. Note that such approaches are generally less favored in CP, as efficient constraints such as `dispersion` offer simple yet richer ways of balancing solutions.

The previous constraints focused on balancing a set of values around a mean using some norm. The `balance` constraint and its derivatives instead balance occurrences of values [28]. The original constraint is of the form $\texttt{balance}(X, B)$ with X a set of variables and B an interval bounding the difference in occurrences between the values occuring the most and those occuring the least. This constraint can be used, for example, in some resource allocation problems. The main derivative of the `balance` constraint is `allbalance`, which considers only a subset of values. Further variants of these two constraints only take into account a single lower or upper bound instead of an interval (such as `atmostallbalance`). The authors study the feasibility of domain consistency filtering for these constraints, and describe a filtering algorithm achieving domain consistency for `atmostallbalance`.

2.3 Integer Programming

With integer programming, and especially with integer linear programming, balancing a set of values often revolves around the linear paradigm. Such methods include maximizing the minimum value (*maximin*), minimizing the maximum value (*minimax*), and minimizing the range between the minimum and maximum values. These three methods have been studied by Gaudioso and Legato [29], in the form of three linear programs for balancing the workloads assigned to identical machines. The first model maximizes the workload of the least-loaded machine. The second model minimizes the workload of the most-loaded machine. For the last model, a target load is chosen, and the objective is to minimize the sum of the absolute difference between the minimum and maximum loads of the machines and this target. Ichimori [30] also studies the sharing of resources with maximin, minimax, and range minimization (which the author calls *optimal sharing*) objectives. In another instance of range minimization, the balanced traveling salesman problem, as described by Larusic and Punnen [31], aims to minimize the difference between the highest and lowest costs associated with edges in a Hamiltonian cycle. Such a balanced solution can be useful, for example, as part of the maintenance routine for gas turbine engines.

Earlier work on linear programs with maximin or minimax objectives can be found with Kaplan [32]. The author mentions military applications for these linear programs, in the form of mission effectiveness and vessel readiness. More recently, the maximin objective has been useful as part of a multiobjective model for response and recovery in postdisaster operations [33]. This objective ensures fairness in the distribution of supplies to the affected locations, maximizing the minimum rate of satisfied demand among those locations. The maximin and minimax paradigms can also be used as constraints, by enforcing an inequity threshold. This allows, for example, the fair and efficient distribution of donated food in a network of food banks [34]. Food banks have certain capacities (e.g., storage, transportation, budget, workforce, etc.), and constraining inequity below a given threshold allows for food distribution that is considered fair enough, while minimizing food waste.

There also exists a slight variant of the maximin and minimax objectives, called *lexicographic* maximin and minimax [35]. One weakness of the classical maximin and minimax is that while it guarantees, for example, maximum utility for the entity which is the worst off, it does not discriminate between the utilities distributed to the rest of entities in the solution. As such, it may treat many solutions as equivalent, when it could be argued that they are not.[7] Lexicographic versions of these objectives still guarantee maximum utility for the entity which is the worst off. But in addition, the lexicographic criterion dictates that the utility of an entity should be increased as much as possible, as long as doing so does not degrade the utility of any other entity which is worse off than itself. The authors mention several applications for which maximin or minimax objectives (or their lexicographic versions) could be used, among which the distribution of strategic resources, production planning and scheduling, and telecommunications network design. The lexicographic maximin objective has also been used, among other things, for fair bandwidth allocation in multi-application computer networks [36].

In the context of facility location, the maximin objective is used in the p-dispersion problem, while the minimax objective is used in the p-center problem [37]. The p-dispersion problem, which is to locate p facilities such that the minimum distance between any pair of them is maximized, has been studied since the 1970s [38]. This problem is particularly useful in the placement of dangerous facilities, which decreases the odds that a major accident occuring in one of the facilities is going to affect another facility. Another use is in the location of franchises in franchise networks, to minimize competition between members of the same franchise, or in the placement of an undesirable facility, such as a waste dump, ensuring that its distance from a set of existing sites is maximized [39]. The p-center problem, which is to locate p facilities such that the maximum distance between any of the demand points and its closest facility is minimized, sees its origins in the 1960s [13]. In contrast with the former problem, this problem is concerned with the central and fair

[7] For example, using maximin to divide 8 integer utilities fairly among 3 entities could result in equivalent solutions {2, 2, 4} and {2, 3, 3}. Yet, the latter is arguably more fair.

locations of desirable facilities, such as fire stations [40].

Another family of balancing objectives include *minisum* and *maxisum*, which minimize or maximize a sum of values. When those values represent distances from a fixed point of reference, this is then similar to the CP constraints covering the L_1-norm (least absolute deviations). Maxisum can be used instead of maximin for the p-dispersion problem, by maximizing the sum of distances between a set of existing sites and an undesirable facility [39]. However, experimental studies show that the optimal maxisum solution may be a poor one when considering the maximin criterion, and vice-versa [41]. Considering both maximin and maxisum in a biobjective model is a way to mitigate such an issue [42]. Kuby [43] also studies maximin and maxisum objectives in the context of the p-dispersion problem, but uses a maxisum objective coupled with the optimal maximin value as a constraint, in order to discriminate between the equivalent maximin solutions. Minimax and minisum can similarly be combined for the location of desirable facilities in the p-center problem [40]. Other, less popular approaches can also be found. Duin and Volgenant [44], for instance, discuss the minimum deviation problem, wherein it is the difference between the maximum value and the average which must be minimized.

Burkard has done much work on assignment problems in [45] and especially in [46]. He has mostly focused on the Linear Sum Assignment Problem (LSAP) and the Linear Bottleneck Assignment Problem (LBAP). Given n jobs and n machines, and a $n \times n$ cost matrix with c_{ij} the cost of assigning job i to machine j, the optimal assignment for the LSAP will minimize the total costs, while the optimal assignment for the LBAP will minimize the maximum cost (bottleneck). These two problems are in $\mathcal{P}$ and can be solved in polynomial time by various algorithms—however many of the balancing concepts presented in his work can be carried over to the $\mathcal{NP}$ problems which interest us. Burkard mentions multiple balancing objectives. The LBAP seeks to minimize the maximum cost of an assignment. The opposite of this problem can be considered, which is to maximize the minimum cost of an assignment. The Balanced Assignment Problem is the same problem albeit with the objective of minimizing the difference between the minimum and maximum assignment costs (range minimization), which he calls *spread*.

As mentioned previously, minisum is related to the L_p-norms, as they are a sum of distances which must be minimized. The L_p-norms have been under study in some form at least since the 1970s [47]. The most recent work on this, as far as we are aware, is that of Bektaş and Letchford [16]. They present a generic mixed-integer nonlinear program to solve the problem of assigning jobs to workers, such that the sum of the powers of p of the workloads is minimized. In addition, the authors briefly describe a model based on column generation that also achieves this purpose. For the problem of workload distribution on heterogeneous GPUs, Lin et al. [48] construct a mixed-integer nonlinear program which minimizes the sum of the squared deviations of the workloads from the mean.

CHAPTER 3 ARTICLE 1: THE QUADRATIC MULTIKNAPSACK PROBLEM WITH CONFLICTS AND BALANCE CONSTRAINTS

Philippe Olivier, Andrea Lodi, and Gilles Pesant

Published in *INFORMS Journal on Computing* (October 2020)

Abstract. The Quadratic Multiknapsack Problem consists of packing a set of items of various weights into knapsacks of limited capacities with profits being associated with pairs of items packed into the same knapsack. This problem has been solved by various heuristics since its inception, and more recently it has also been solved with an exact method. We introduce a generalization of this problem that includes pairwise conflicts as well as balance constraints, among other particularities. We present and compare constraint programming and integer programming approaches for solving this generalized problem.

3.1 Introduction

The *Quadratic Multiknapsack Problem* (QMKP) consists of packing a set of items of various weights into knapsacks of limited capacities with profits being associated with pairs of items packed into the same knapsack. We introduce a generalization of this problem that includes pairwise conflicts as well as balance constraints, among other particularities.

An application of this problem concerns event seating. Event organizers must often construct seating plans for hundreds or thousands of guests. These guests could be friends, business partners/competitors, acquaintances which share affinities (or not), and so on. These translate into costs and conflicts, and balancing a seating arrangement is nothing more than packing items into knapsacks. Multiple companies offer software solutions to solve this problem (such as Wedding Seat Planner,[1] Perfect Table Plan,[2] and others), and to the best of our knowledge the solution methods used by these services are metaheuristics such as tabu search or genetic algorithms [49]. [50] presents a detailed discussion of other QMKP applications.

In this paper, we present and compare constraint programming and integer programming approaches for solving this generalized problem. Section 3.2 gives a formal definition of

[1] www.weddingseatplanner.com
[2] www.perfecttableplan.com

our problem and Section 3.3 reviews current methods of solving the QMKP and similar problems. Sections 3.4 to 3.6 introduce, respectively, our constraint programming (CP) model and our two integer programming (IP) formulations. Section 3.7 discusses the results of our experiments.

3.2 Description of the Problem

Our problem is a generalization of the QMKP.[3] Let

- $\mathcal{I} = \{1, \ldots, n\}$ be the index set of items,
- $\mathcal{K} = \{1, \ldots, m\}$ be the index set of knapsacks,
- w_i denote the weight of item i with $w = \sum_{i \in \mathcal{I}} w_i$ representing the combined weight of all the items,
- ℓ and u be, respectively, the lower and upper bounds on the sum of the weights of the items placed in a knapsack (the load of a knapsack),[4]
- c_{ij} be the cost incurred if items i and j are packed into the same knapsack.

Entries in the cost matrix C can take any integer value. Namely,

$$c_{ij} \begin{cases} = M\,^5, & \text{if } i \text{ and } j \text{ are in conflict and must be packed into separate knapsacks,} \\ = 0, & \text{if } i \text{ and } j \text{ have no cost,} \\ < 0, & \text{if } i \text{ and } j \text{ would rather be packed into the same knapsack,} \\ > 0, & \text{if } i \text{ and } j \text{ would rather be packed into separate knapsacks.} \end{cases}$$

Since a conflict is expressed as having a prohibitive cost, the initial cost matrix can be enhanced by adding this prohibitive cost for each pair of items whose combined weights is greater than u, since they can never be packed together. The problem consists of packing all items into knapsacks such that their load is between the given bounds and additionally

[3] While knapsack problems are generally presented as maximizing profits, in this paper we instead take the approach of minimizing costs.

[4] Note that in this paper, these values do not bound the capacities of the knapsacks themselves. Rather, they are used for early pruning of some of the search space. See Section 3.5.5 for details.

[5] Chosen to be large enough that the items are ensured to be packed separately.

enforcing some load balancing constraints discussed in the next section. The objective is to minimize the combined cost of all available knapsacks.

3.2.1 Load Balancing

For a collection of points (the loads of the individual knapsacks in our case), the global distance between these points and their arithmetic mean μ, according to a norm p, is defined by the concept of L_p-deviation

$$\sum_{i=1}^{n} |x_i - \mu|^p.$$

For this problem, we will be considering load balancing constraints under four distinct norms which yield incomparable results. The L_0-norm minimizes the number of nonzero deviations from the mean knapsack load. The L_1- and L_2-norms minimize, respectively, the sum of absolute and squared deviations from the mean. The L_∞-norm minimizes the maximum deviation from the mean.

The various norms cover more than what the ℓ and u bounds can achieve by themselves, when manually specified by a decision maker. In our context, perhaps the decision maker is interested in looking at the tradeoffs associated with having specific numbers of knapsacks perfectly balanced (L_0). Maybe, one wants the knapsacks to be tightly balanced with few outliers (L_2). The decision maker possibly knows to have to face some outliers, and she does not want them to have too big of an impact on the general balance of the other knapsacks (L_1). Or, she does not want the worst case to be too bad (L_∞). Despite the fact that the ℓ and u bounds could be used directly for some balance, there is a richer way of approaching the balancing aspect of the problem (at the cost of introducing additional complexity). This is the direction that we follow in the paper.

Finally, note that to help assessing the impact of the load balancing parameters (through generation of Pareto sets, see Section 3.7), we set lower and upper bounds $d_{\min}$ and $d_{\max}$ on the deviation from the mean when we solve our problem.

3.3 Related Work

The QMKP is a relatively recent problem [51]. It is a generalization of the *Multiple Knapsack Problem* and of the *Quadratic Knapsack Problem*. This problem finds applications in project management and capacity planning, among many others. For example, a manager may be tasked with assigning members to projects, which have budgets on salaries. Members have some individual productivity, as well as some group productivity depending on who they work with. The manager will want to construct the most productive teams for the various projects while remaining within their respective budgets [51]. Since its inception, the QMKP has been solved by various heuristics such as genetic algorithms, artificial bee colony algorithms, and others. However, no exact methods to solve this problem had been published until very recently, when a branch-and-price approach was investigated [50]. Bergman constructed an exponentially-sized model derived from a set packing problem formulation (which imposes no constraint on the number of items to be packed), and a similar model derived from a set partitioning problem formulation (in which all the items are required be packed). For a thorough review of the literature concerning the QMKP, its heuristic solving methods, details on its exact solving method, and its applications, the reader is referred to Bergman's paper [50].

Little research has explicitly focused on table seating problems. As far as we can tell, the problem of constructing seating plans was informally introduced a few years ago [52]. The authors used an IP model to solve various instances of this problem for their own wedding. Their IP model was inefficient and could not solve a problem of practical size in reasonable time. This seating assignment problem was later formally described as the *Wedding Seating Problem* [49]. The author solved the problem with a metaheuristic model based on a two-stage tabu search. In a further paper, a quadratic IP model was devised, but it could not improve on the metaheuristic model in most cases [53].

The QMKP shares some similarities with bin packing. CP approaches to bin packing and load balancing have been surveyed fairly recently [54]. Most research on bin packing with conflicts (e.g. [55]), focuses on minimizing the number of bins used as in the classical problem (albeit with the added conflict dimension). In contrast, our version of the QMKP uses a fixed number of knapsacks of non-fixed capacities, the objective being to optimize a scoring function subject to additional balancing constraints.

Balancing with the L_1- and L_2-norms in CP can be achieved with the `dispersion` constraint [27], whose filtering algorithm can handle all norms. For other norms, balancing can be accomplished with classical constraints[6] such as `maximum`. In the field of IP, balancing is not encapsulated in a turnkey constraint as in CP, and its integration in a model is more often than not *ad hoc*. Specific examples of balance in IP include the *Nurse-Patient Assignment Problem* [56] and the *Balanced Academic Curriculum Problem* [57].

In the preliminary version of the present paper, three algorithms similar to those presented

[6] See the Global Constraint Catalog: http://sofdem.github.io/gccat

here have been introduced, but only the L_1-norm was considered. A CP model used a large neighborhood search (LNS) strategy while an IP model based on column generation did not implement a branch-and-price algorithm. Thus, [58] provided no provable guarantees of optimality.

3.4 CP Model

For each item i, we define a decision variable b_i whose value is the knapsack in which the item is packed. For each knapsack k, we define an auxiliary variable o_k whose value is the load of the knapsack. To pack the items into the knapsacks, the model uses a `binpacking` constraint (3.1)–(3.4)

$$\begin{aligned}
&\texttt{binpacking}\left(\langle b_i\rangle, \langle w_i\rangle, \langle o_k\rangle\right) && && (3.1)\\
&i \in \mathcal{I} && && (3.2)\\
&b_i \in \mathcal{K} && && (3.3)\\
&o_k \in \mathbb{N}, && k \in \mathcal{K}. && (3.4)
\end{aligned}$$

The `binpacking` constraint uses a filtering algorithm that ensures both that each item is assigned to a bin (knapsack), and that the load of each bin is equal to the sum of the weights of the items assigned to it [59]. The deviation, represented by variable σ, is taken care of by various constraints depending on the norm being used. For the L_0-norm, auxiliary binary variables keep track of the number of unbalanced knapsacks (3.5)–(3.6). The `dispersion` constraint is used for the L_1- and L_2-norms (3.7). The L_∞-norm uses the simple `maximum` and `minimum` constraints to determine its deviation (3.8)

$$\begin{aligned}
&\mathit{unbalanced}_k = (o_k < \lfloor w/m \rfloor \vee o_k > \lceil w/m \rceil), \quad \forall k \in \mathcal{K} && (3.5)\\
&\sigma = \sum_{k \in \mathcal{K}} \mathit{unbalanced}_k && (3.6)\\
&\texttt{dispersion}\left(\{o_k\}, w/m, \sigma\right) && (3.7)\\
&\sigma = \texttt{maximum}_{k\in\mathcal{K}}\{|o_k - w/m|\}. && (3.8)
\end{aligned}$$

Conflicting items are packed into different knapsacks with simple binary inequality constraints (3.9). However, in addition to this the model also uses a conflict graph to infer `alldifferent` constraints (3.10) [60]. In a conflict graph, each item is represented by a vertex, and an edge joins two vertices if they are conflicting. By extracting all the maximal cliques of this graph, it is possible to add `alldifferent` constraints to the model for each one of those cliques. Furthermore, the maximum clique (the largest of all the maximal cliques) determines a lower bound on the number of knapsacks necessary to find a feasible solution. The Bron-Kerbosch algorithm is an exact method that can be used to find all the maximal cliques of the conflict graph [61]. Let $\mathcal{M}$ be the set of all maximal cliques of size 3 or more (maximal clique r being, for example, $\mathcal{M}^r = \{b^r_1, \ldots, b^r_{|\mathcal{M}^r|}\}$)

$$\begin{aligned} & b_i \neq b_j, && \forall i, j \in \mathcal{I} : c_{ij} = M && (3.9) \\ & \texttt{alldifferent}\left(\{\mathcal{M}^r\}\right), && \forall r \in \{1, \ldots, |\mathcal{M}|\}. && (3.10) \end{aligned}$$

While we have not explored all specific edge cases in this paper, if we were to solve highly constrained instances of the problem (i.e., with a dense conflict graph) these could be intractable for the Bron-Kerbosch algorithm. A heuristic for finding a subset of the complete collection of maximal cliques could instead be applied. Constraints (3.9) would also allow the conflict graph to be dropped altogether without affecting the optimality of the results.

Symmetry is broken in two different ways. First, static symmetry breaking is achieved by fixing each item of the maximum clique $\mathcal{M}^{\max}$ of the conflict graph to a separate knapsack (3.11)

$$b^{\max}_i = i, \quad \forall i \in \{1, \ldots, |\mathcal{M}^{\max}|\}. \tag{3.11}$$

Further symmetry can be broken by forcing a nonincreasing order upon the loads of the knapsacks not involved in static symmetry breaking (3.12)

$$o_k \geq o_{k+1}, \quad \forall k \in \{|\mathcal{M}^{\max}| + 1, \ldots, m - 1\}. \tag{3.12}$$

Finally, the objective is to minimize the costs (3.14), subject to a constraint bounding the value of the global deviation of the knapsacks (3.13)

$$d_{\min} < \sigma \leq d_{\max} \tag{3.13}$$

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(b_i = b_j\right) c_{ij}. \tag{3.14}$$

Various custom branching heuristics have been tried for this model, but with little success. Finding a good branching heuristic is not a trivial task, for two main reasons. First, packing items into knapsacks introduces some amount of deviation. The various balancing constraints force the global deviation to be within a specific range, and it is hard to guess how much deviation to allow early on in the packing when many items are left to be packed later. Second, custom branching heuristics interfere with symmetry breaking, rendering it much less effective. In the end, we were not able to do better than CP Optimizer's default branching strategy, which uses LNS to improve the bound and failure-directed search to prove optimality [62].

We have also tried to use a LNS strategy [63] in the solving process, and the resulting algorithm, as already observed in our previous paper [58], finds very good solutions early on but quickly reaches a plateau, well before the optimum. Results from this previous paper show that increasing the time limit from 6 to 600 seconds only marginally improves the solution quality and, considering that the focus of the present paper is on proving optimality, no detailed experiments using the LNS strategy are reported.

3.5 IP Model A

The first IP model (IP$_\text{A}$) is a generalization of the natural quadratic IP model originally proposed to solve the *Wedding Seating Problem* [53]. A barebones model defined by (3.15)–(3.22) is common to all load balancing norms. A $n \times m$ matrix of decision variables x represents packing assignments of items into knapsacks (3.15)

$$x_{ik} = \begin{cases} 1, & \text{if item } i \text{ is packed into knapsack } k, \\ 0, & \text{otherwise.} \end{cases} \tag{3.15}$$

$$(\text{IP}_\text{A}) \quad \min \quad \sum_{k \in \mathcal{K}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ik} x_{jk} c_{ij} \tag{3.16}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K}} x_{ik} = 1, \qquad \forall i \in \mathcal{I} \tag{3.17}$$

$$x_{ik} + x_{jk} \leq 1, \qquad \forall i, j \in \mathcal{I} : c_{ij} = M, \quad \forall k \in \mathcal{K} \tag{3.18}$$

$$\sum_{i \in \mathcal{I}} x_{ik} w_i \geq \ell, \qquad \forall k \in \mathcal{K} \tag{3.19}$$

$$\sum_{i \in \mathcal{I}} x_{ik} w_i \leq u, \qquad \forall k \in \mathcal{K} \tag{3.20}$$

$$x_{ik} = 0, \qquad \forall i \in \mathcal{I}, \qquad \forall k \in \{i+1, \ldots, m\} \tag{3.21}$$

$$x_{ik} \in \{0, 1\}, \qquad \forall i \in \mathcal{I}, \qquad \forall k \in \mathcal{K}. \tag{3.22}$$

The items are required to be packed into one and only one knapsack (3.17), and conflicting items may not be packed into the same knapsack (3.18). Provided that the M value used in the objective function for pairwise conflicting items is selected carefully, constraints (3.18) are redundant. However, we keep them in the model for clarity and because in combination with the objective function, they are computationally useful (with respect to the continuous relaxation). Constraints (3.19) and (3.20) require that the load of every knapsack be within bounds ℓ and u. Some symmetry breaking is achieved with (3.21). Finally, constraints (3.22) ensure the integrality of the solution. The barebones model minimizes the sum of all pairwise costs between items for all knapsacks (3.16).

3.5.1 L_0-norm

Integration of load balancing according to the L_0-norm is achieved by adding constraints (3.23)–(3.33) to the barebones model

$$\sum_{i \in \mathcal{I}} w_i x_{ik} \geq \lfloor w/m \rfloor - M_1(1 - \underline{b_k}), \quad \forall k \in \mathcal{K} \tag{3.23}$$

$$\lfloor w/m \rfloor \geq \sum_{i \in \mathcal{I}} w_i x_{ik} - M_1 \underline{b_k} + \epsilon, \qquad \forall k \in \mathcal{K} \tag{3.24}$$

$$\underline{b_k} \in \{0, 1\}, \qquad \forall k \in \mathcal{K} \tag{3.25}$$

$$\sum_{i \in \mathcal{I}} w_i x_{ik} \leq \lceil w/m \rceil + M_2(1 - \overline{b_k}), \quad \forall k \in \mathcal{K} \tag{3.26}$$

$$\lceil w/m \rceil \leq \sum_{i \in \mathcal{I}} w_i x_{ik} + M_2 \overline{b_k} - \epsilon, \qquad \forall k \in \mathcal{K} \tag{3.27}$$

$$\overline{b_k} \in \{0, 1\}, \qquad \forall k \in \mathcal{K} \tag{3.28}$$

$$\underline{b_k} + \overline{b_k} - 2b_k \geq 0, \qquad \forall k \in \mathcal{K} \tag{3.29}$$

$$\underline{b_k} + \overline{b_k} - 2b_k \leq 1, \qquad \forall k \in \mathcal{K} \tag{3.30}$$

$$b_k \in \{0,1\}, \qquad \forall k \in \mathcal{K} \tag{3.31}$$

$$m - \sum_{k \in \mathcal{K}} b_k \geq d_{\min} \tag{3.32}$$

$$m - \sum_{k \in \mathcal{K}} b_k \leq d_{\max} \tag{3.33}$$

with $M_1 = |u - \lfloor w/m \rfloor\,|$ and $M_2 = |u - \lceil w/m \rceil\,|$. With the help of auxiliary variables $\underline{b_k}$ and $\overline{b_k}$ (not to be confused with variables b of the CP model), constraints (3.23)–(3.28) indicate if the load of a knapsack k is at least $\lfloor w/m \rfloor$ or at most $\lceil w/m \rceil$, respectively. A small value (ϵ) in constraints (3.24) and (3.27) ensures their effectiveness when the mean is integral. Auxiliary variables b in (3.29)–(3.31) use the values of $\underline{b}$ and $\overline{b}$ to keep track of the balanced and unbalanced knapsacks. Finally, constraints (3.32) and (3.33) ensure that the number of unbalanced knapsacks is bounded by $d_{\min}$ and $d_{\max}$.

3.5.2 L_1-norm

Integration of load balancing according to the L_1-norm is achieved by adding constraints (3.34)–(3.37) to the barebones model

$$\sum_{i \in \mathcal{I}} x_{ik} w_{ik} - w/m \leq o_k, \quad \forall k \in \mathcal{K} \tag{3.34}$$

$$w/m - \sum_{i \in \mathcal{I}} x_{ik} w_{ik} \leq o_k, \quad \forall k \in \mathcal{K} \tag{3.35}$$

$$\sum_{k \in \mathcal{K}} o_k \geq d_{\min} \tag{3.36}$$

$$\sum_{k \in \mathcal{K}} o_k \leq d_{\max}. \tag{3.37}$$

Constraints (3.34)–(3.35) model an absolute value function that provides the absolute deviation of each knapsack. Constraints (3.36) and (3.37) stipulate that the global deviation of the solution must be bounded by $d_{\min}$ and $d_{\max}$.

Due to the intrinsic weakness of (any) linearization of the absolute value function by constraints (3.34)–(3.35), constraint (3.36) is effective only if the objective function is amended by a small positive term $\epsilon \sum_{k \in \mathcal{K}} o_k$ (used to force tightness of (3.34)–(3.35)). It is, however, not trivial to define an appropriate value for ϵ, i.e., one that does not lead to produce suboptimal solutions. For this reason, in our experiments we do not strictly impose the $d_{\min}$

bound—the solutions remain feasible but construction of the Pareto set for this norm may cause the exploration of overlapping solution spaces.

3.5.3 L_2-norm

Enforcing load balancing according to the L_2-norm introduces nonconvex quadratic constraints into the model. Such a class of problems cannot be solved to optimality by our preferred mathematical programming solver CPLEX and neither by the other solvers in the same class like, for example, XPRESS and GUROBI. We have experimented with a convex relaxation of the problem obtained by using McCormick envelopes [64] but the deviations associated with solutions of this relaxed problem often unreasonably exceed the $d_{\min}$ and $d_{\max}$ bounds, thus leading to solutions with limited interest in comparison with the other approaches tested here. Of course, one could attack this nonconvex quadratically-constrained problem with a global optimization solver like BARON or SCIP but we have decided to avoid that because the IP model discussed in Section 3.6 turns out to be more suitable to deal with the L_2-norm.

3.5.4 L_∞-norm

Integration of load balancing according to the L_∞-norm is achieved by adding constraints (3.34)–(3.35), (3.38), and (3.39) to the barebones model

$$o_k \le o_{\max}, \qquad \forall k \in \mathcal{K} \tag{3.38}$$

$$o_{\max} \le d_{\max}. \tag{3.39}$$

Constraints (3.34)–(3.35) and (3.38) force auxiliary variable $o_{\max}$ to be at least as large as the maximum deviation. Constraint (3.39) forces an upper bound on the deviation. Note that, the same discussion of Section 3.5.2 applies on the difficulty of imposing the $d_{\min}$ bound.

3.5.5 Offline Optimization

We notice that depending on the value of $d_{\max}$, it may be possible to tighten the ℓ and u bounds with some arithmetic deductions.[7] This offline optimization should help prune

[7] Note that this is already taken into consideration by the `dispersion` constraint for the CP model.

nodes leading to infeasible solutions early, without eliminating any feasible solution. The L_0-norm minimizes the number of nonzero deviations from the mean knapsack load. To tackle fractional means with this norm, we allow some leeway in that any load equal to one of the two nearest integers of a fractional mean is considered balanced. For the L_0-norm, load bounds optimization is most effective when $d_{\max}$ is low, otherwise knapsacks can account for very high deviations (3.40)–(3.41)

$$\ell \geq [d_{\max} = 0] \cdot \lfloor w/m \rfloor + [d_{\max} = 1] \cdot (w - (m-1) \times \lceil w/m \rceil) + [d_{\max} \geq 2] \cdot 0, \quad (3.40)$$

$$u \leq [d_{\max} = 0] \cdot \lceil w/m \rceil + [d_{\max} \neq 0] \cdot (w - \max\{0, m - d_{\max}\}) \times \lfloor w/m \rfloor. \quad (3.41)$$

The L_1-norm minimizes the sum of absolute deviations from the mean. In the worst case, a single knapsack can account for at most half of the deviation, since the cumulative weight in excess of the mean knapsack load will always be equal to the cumulative weight short of the mean knapsack load (3.42)–(3.43)

$$\ell \geq \max\left\{0, \lceil w/m - d_{\max}/2 \rceil\right\}, \quad (3.42)$$

$$u \leq \lfloor w/m + d_{\max}/2 \rfloor. \quad (3.43)$$

The L_2-norm minimizes the sum of squared deviations from the mean. For this norm, we must also take the number of knapsacks into account in order to tightly bound the most deviative knapsack (3.44)–(3.45)

$$\ell \geq \max\{0, \lceil w/m - \sqrt{d_{\max} \times (m-1)/m} \rceil\}, \quad (3.44)$$

$$u \leq \lfloor w/m + \sqrt{d_{\max} \times (m-1)/m} \rfloor. \quad (3.45)$$

The L_∞-norm minimizes the maximum deviation from the mean. Here, the relationship between the bounds and $d_{\max}$ is simple since only the most deviative knapsack matters (3.46)–(3.47)

$$\ell \geq \max\left\{0, \lceil w/m - d_{\max} \rceil\right\}, \quad (3.46)$$

$$u \leq \lfloor w/m + d_{\max} \rfloor . \tag{3.47}$$

We also introduce the concept of *best balance*, which will be helpful for quickly proving infeasibility for the L_1- and L_2-norms when $d_{\max}$ is relatively low. Best balance refers to the best possible way of balancing loads, *theoretically*.[8] If the mean load were integral, then we *could* be able to achieve identical loads for all knapsacks. In such a case, best balance would have been reached as there would not exist a better way of balancing the loads. Similarly, if the mean load were fractional, then we *could* be able to assign loads equal to one of the two nearest integers of the mean for all knapsacks. With all loads assigned as such, best balance would be achieved as no other solution could offer a better balance.[9] The deviation of a solution which is best balanced represents the minimum feasible deviation for that problem. Let $\overline{f} = w \bmod m$ be the number of above-mean loads, $\underline{f} = m - (w \bmod m)$ be the number of below-mean loads, $\overline{g} = 1 - (w \bmod m)/m$ be the absolute deviation of an above-mean load, and $\underline{g} = (w \bmod m)/m$ be the absolute deviation of a below-mean load, the minimum feasible deviation of the L_p-norm is $d^p_{\text{best}} = \overline{f} \times \overline{g}^p + \underline{f} \times \underline{g}^p$. It is then obvious that if the condition $d_{\max} < d^p_{\text{best}}$ holds, then the problem can be deemed infeasible without having had to solve it.

3.6 IP Model B

Our second IP model (IP$_\text{B}$) is based on the definition of an exponential-size collection of possible knapsacks, similar to the classical bin packing problem. Indeed, as for bin packing, the resulting formulation can be solved by column generation with a set partitioning (SP) model.

3.6.1 Master Problem

Let $\mathbb{S}$ be the collection of all subsets of items that can be packed into the same knapsack

$$\mathbb{S} := \left\{ S \subseteq \{1, \ldots, n\} : \ell \leq \sum_{i \in S} w_i \leq u, \quad \forall i, j \in S : c_{ij} \neq M \right\}.$$

There is a binary variable for each subset $S \in \mathbb{S}$ representing a combination of items, or *pattern*, to be packed into the same knapsack

[8] Here, *theoretically* is used to mean *in the best case scenario*, i.e., supposing that the distribution of item weights allows it and barring other interfering constraints.

[9] All questions of symmetry aside, the distribution of loads achieving best balance is always unique.

$$x_S = \begin{cases} 1, & \text{if pattern } S \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.48}$$

The sum of all pairwise costs for the items of a pattern and its deviation are represented by α_S and β_S, respectively. Since the number of knapsacks is not variable, this grants us two advantages in regards to the balancing objective. First, the values of α_S and β_S need to be computed only once per pattern (when it is generated) and remain constant throughout the process. Second and more importantly, since β_S is computed outside of the program, the norm according to which it is computed may not complexify the problem (i.e., the program could remain linear even if balance were computed according to the L_2-norm).

One may be tempted to solve this problem with a set covering (SC) model. However, we can observe that it may not be possible to *only* construct maximal sets with regards to the knapsack capacity due to conflicts between items and the constraints enforcing them. While the solution of a SP model is always directly feasible, that of a SC model must be transformed in order to be feasible for our problem (i.e., we must remove all duplicate items from the knapsacks). The unfortunate effect of using a SC-based formulation is that neither feasibility nor optimality can be guaranteed. Example 1 illustrates the underlying issue of using a SC model to solve our problem.

Example 1 *Assume an instance of the problem with 2 knapsacks and 4 items A, B, C, and D. The pairwise costs of AB, BC, CD, and DA are 0, while the pairwise cost of AC is 1 and that of BD is -2. We also have $\ell = 2$ and $u = 3$. The two most profitable maximal subsets are ABD and BCD which both have a value of -2 and cover all the items. The initial solution of the SC model would be $(-2) + (-2) = -4$, which must be rendered feasible for our problem by removing B from one knapsack and D from the other. This modified solution would have a value of $(0) + (0) = 0$, while the solution of a SP model would be to pack AC and BD in separate knapsacks, for a value of $(1) + (-2) = -1$.* ■

The master problem (MP) is thus based on the definition of a SP model

$$\begin{aligned} (\mathrm{MP_P}) \quad \min \quad & \sum_{S \in \mathbb{S}} \alpha_S x_S && (3.49) \\ \text{s.t.} \quad & \sum_{S \in \mathbb{S}: i \in S} x_S = 1, \qquad \forall i \in \mathcal{I} && (3.50) \\ & \sum_{S \in \mathbb{S}} x_S = m && (3.51) \end{aligned}$$

$$\sum_{S \in \mathbb{S}} \beta_S x_S \geq d_{\min} \tag{3.52}$$

$$\sum_{S \in \mathbb{S}} \beta_S x_S \leq d_{\max} \tag{3.53}$$

$$x_S \in \{0, 1\}, \qquad \forall S \in \mathbb{S}. \tag{3.54}$$

The MP minimizes the combined costs of all knapsacks (3.49), under the conditions that each item be packed into one and only one knapsack (3.50), that a total of m knapsacks be used (3.51), and that the global deviation of a solution be within bounds $d_{\min}$ and $d_{\max}$ (3.52)–(3.53). Finally, constraints (3.54) ensure the integrality of the solution. It should be noted that for the L_∞-norm only the largest deviation matters. As such, that norm does not consider constraints (3.52)–(3.53).[10] The dual of the continuous relaxation of the MP obtained by replacing constraints (3.54) with $x_S \geq 0, \forall S \in \mathbb{S}$ is[11]

$$\begin{aligned}
(\text{MP}_\text{D}) \quad \max \quad & \sum_{i \in \mathcal{I}} y_i + m\zeta + d_{\min}\gamma + d_{\max}\delta && && (3.55)\\
\text{s.t.} \quad & \sum_{i \in \mathcal{S}} y_i + \zeta + \beta_S(\gamma + \delta) \leq \alpha_S, && \forall S \in \mathbb{S} && (3.56)\\
& y_i \text{ free}, && \forall i \in \mathcal{I} && (3.57)\\
& \zeta \text{ free} && && (3.58)\\
& \gamma \geq 0 && && (3.59)\\
& \delta \leq 0. && && (3.60)
\end{aligned}$$

3.6.2 Initial Columns

In order to begin solving the problem, the column generation algorithm needs m columns making up an initial feasible solution of the problem. These columns are generated by the first feasible solution found by the CP model presented in Section 3.4. These m columns make up the initial restricted master problem (RMP) with (3.54) relaxed. This is all that is needed to initialize column generation.

[10] For the L_∞-norm, the deviation need only be constrained in the pricing problem.

[11] For the L_∞-norm, the dual is slightly different since its MP does not consider constraints (3.52)–(3.53).

3.6.3 Column Generation

The RMP starts off with the small subset of variables previously generated by the CP model, and recall that its integrality constraints (3.54) are relaxed. The column generation algorithm then proceeds as follows:

1. The RMP is solved, yielding its dual values in the process.
2. A subproblem, or pricing problem (PP), makes use of these dual values to generate promising new columns.
3. If the PP can find negative reduced cost solutions (i.e., solutions which violate (3.56)), their α and β values are computed and these new columns are added to the RMP before going back to step 1. If no such column can be generated, then the current solution of the continuous relaxation of the RMP is also optimal for the full problem, and provides its lower bound.

Two IP-based and two CP-based pricing problems have been implemented and are presented below. All pricing problems have variants for each norm.

First IP-Based Pricing Problem (IP1)

The first implementation is a 0–1 IP model with an objective. The following constraints are common to all norms. Let

$$z_i = \begin{cases} 1, & \text{if item } i \text{ is packed into the new knapsack/pattern,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.61}$$

$$\begin{aligned} & z_i + z_j \leq 1, \quad \forall i, j \in \mathcal{I} : c_{ij} = M && (3.62) \\ & \sum_{i \in \mathcal{I}} w_i z_i \geq \ell && (3.63) \\ & \sum_{i \in \mathcal{I}} w_i z_i \leq u && (3.64) \\ & z_i \in \{0, 1\}, \quad \forall i \in \mathcal{I}. && (3.65) \end{aligned}$$

These constraints stipulate that no conflicting items be packed together in the new knapsack (3.62), and that its load be within bounds ℓ and u (3.63)–(3.64). For the L_0-norm the PP is defined by (3.61)–(3.65), a straightforward adaptation of (3.23)–(3.31), and

$$
\begin{aligned}
\max \quad & \sum_{i \in \mathcal{I}} y_i^* z_i + \zeta^* + \beta(\gamma^* + \delta^*) - \alpha_S \quad \text{if } \gamma^* + \delta^* < 0 && (3.66)\\
\max \quad & \sum_{i \in \mathcal{I}} y_i^* z_i + \zeta^* - \beta(\gamma^* + \delta^*) - \alpha_S \quad \text{if } \gamma^* + \delta^* > 0 && (3.67)\\
\text{s.t.} \quad & \beta = b && (3.68)\\
& \beta \in \{0, 1\}. && (3.69)
\end{aligned}
$$

The objective is to minimize the value of a knapsack (3.66)/(3.67), with β being defined by the adapted (3.23)–(3.31) and (3.68)–(3.69). For the L_1-norm the objective is the same, but β is defined by

$$
\begin{aligned}
& \sum_{i \in \mathcal{I}} w_i z_i - w/m \leq \beta && (3.70)\\
& \sum_{i \in \mathcal{I}} w_i z_i - w/m \geq \beta && (3.71)\\
& \beta \in \mathbb{N}. && (3.72)
\end{aligned}
$$

Constraints (3.70)–(3.72) work in the same manner as constraints (3.34)–(3.35) and ensure that deviation β is always positive. The L_2-norm also uses (3.70)–(3.72) but its (nonlinear) objective is

$$
\begin{aligned}
\max \quad & \sum_{i \in \mathcal{I}} y_i^* z_i + \zeta^* + \beta^2(\gamma^* + \delta^*) - \alpha_S \quad \text{if } \gamma^* + \delta^* < 0 && (3.73)\\
\max \quad & \sum_{i \in \mathcal{I}} y_i^* z_i + \zeta^* - \beta^2(\gamma^* + \delta^*) - \alpha_S \quad \text{if } \gamma^* + \delta^* > 0. && (3.74)
\end{aligned}
$$

Finally, the L_∞-norm again uses (3.70)–(3.72) but also bounds β (3.76), and its objective is slightly different (3.75)

$$
\begin{aligned}
\max \quad & \sum_{i \in \mathcal{I}} y_i^* z_i + \zeta^* - \alpha_S && (3.75)\\
\text{s.t.} \quad & \beta \leq d_{\max}. && (3.76)
\end{aligned}
$$

Second IP-Based Pricing Problem (IP2)

The second implementation is a 0–1 IP model with no objective, i.e., a feasibility IP. In column generation (as well as cutting plane), it is often natural to maximize violation (the most negative reduced cost corresponds to the most violated constraint in the dual) but that is not strictly necessary for algorithmic convergence. Thus, this reformulation of the previous IP model simply moves the objectives into the constraints, forcing their value to be positive: Feasible solutions of IP2 necessarily violate (3.56), and if IP2 does not find feasible solutions, then (3.56) is not violated.

First CP-Based Pricing Problem (CP1)

The third implementation is a CP model using 0–1 decision variables. This new model is identical to IP2 save for a few modifications. The adapted constraints (3.23)–(3.31) as well as constraints (3.68)–(3.69) are replaced by

$$\beta = \left(\sum_{i \in \mathcal{I}} w_i z_i < \lfloor w/m \rfloor \right) \vee \left(\sum_{i \in \mathcal{I}} w_i z_i > \lceil w/m \rceil \right) \tag{3.77}$$

and constraints (3.70)–(3.71) are replaced by

$$\beta = \left| w/m - \sum_{i \in \mathcal{I}} w_i z_i \right| . \tag{3.78}$$

Second CP-Based Pricing Problem (CP2)

Differently from CP1, the fourth implementation is a more natural CP model. First, we define item set $\mathcal{I}' = \mathcal{I} \cup \{n+1\}$ which is equivalent to the original item set augmented with a weightless and costless dummy item.[12] The knapsack has q available positions into which items can be packed.[13] Decision variables $z_{p \in \mathcal{P}} \in \mathcal{I}'$ with $\mathcal{P} = \{1, \ldots, q\}$ and constraints

[12] For this model, it is assumed that w and C are also augmented with this dummy item.

[13] The number of available positions q is chosen to be as small as possible, yet still ensuring that all feasible knapsacks can be constructed.

$$\texttt{distribute}(z, \{0,1\}, \mathcal{I}) \tag{3.79}$$

$$\sum_{p \in \mathcal{P}} w_{z_p} \geq \ell \tag{3.80}$$

$$\sum_{p \in \mathcal{P}} w_{z_p} \leq u \tag{3.81}$$

$$\texttt{table}(\{z_{p_1}, z_{p_2}\}, \tau), \qquad \forall p_1, p_2 \in \mathcal{P} \tag{3.82}$$

$$z_p \leq z_{p+1}, \qquad \forall p \in \{1, \dots, q-1\} \tag{3.83}$$

with $\tau = \{\langle i,j \rangle : (c_{ij} = M) \wedge (i < j), \forall i,j \in \mathcal{I}\}$ are common to all norms. Items can be packed only once (3.79), and (3.80)–(3.81) bound the weight of the knapsack. Conflicting items are managed by a table of forbidden assignments (3.82), and some symmetry breaking is achieved with (3.83). For the L_0-norm, auxiliary variable β is defined by

$$\beta = \left(\sum_{p \in \mathcal{P}} w_{z_p} < \lfloor w/m \rfloor \right) \vee \left(\sum_{p \in \mathcal{P}} w_{z_p} > \lceil w/m \rceil \right) \tag{3.84}$$

while for the other norms it is

$$\beta = \left| w/m - \sum_{p \in \mathcal{P}} w_{z_p} \right|. \tag{3.85}$$

Furthermore, the L_∞-norm has an extra constraint enforcing a bound on its maximum deviation

$$\beta \leq d_{\max}. \tag{3.86}$$

As for the previous CP model, the objectives are moved into the constraints for the L_0- and L_1-norms (3.87)–(3.88), the L_2-norm (3.89)–(3.90), and the L_∞-norm (3.91)

$$\sum_{i\in\mathcal{I}:i\in z} y_i^* + \zeta^* + \beta(\gamma^* + \delta^*) - \sum_{i\in\mathcal{I}:i\in z}\ \sum_{j\in\mathcal{I}:(j\in z)\wedge(i>j)} c_{z_iz_j} > 0 \quad \text{if } \gamma^* + \delta^* < 0 \tag{3.87}$$

$$\sum_{i\in\mathcal{I}:i\in z} y_i^* + \zeta^* - \beta(\gamma^* + \delta^*) - \sum_{i\in\mathcal{I}:i\in z}\ \sum_{j\in\mathcal{I}:(j\in z)\wedge(i>j)} c_{z_iz_j} > 0 \quad \text{if } \gamma^* + \delta^* > 0 \tag{3.88}$$

$$\sum_{i\in\mathcal{I}:i\in z} y_i^* + \zeta^* + \beta^2(\gamma^* + \delta^*) - \sum_{i\in\mathcal{I}:i\in z}\ \sum_{j\in\mathcal{I}:(j\in z)\wedge(i>j)} c_{z_iz_j} > 0 \quad \text{if } \gamma^* + \delta^* < 0 \tag{3.89}$$

$$\sum_{i\in\mathcal{I}:i\in z} y_i^* + \zeta^* - \beta^2(\gamma^* + \delta^*) - \sum_{i\in\mathcal{I}:i\in z}\ \sum_{j\in\mathcal{I}:(j\in z)\wedge(i>j)} c_{z_iz_j} > 0 \quad \text{if } \gamma^* + \delta^* > 0 \tag{3.90}$$

$$\sum_{i\in\mathcal{I}:i\in z} y_i^* + \zeta^* - \sum_{i\in\mathcal{I}:i\in z}\ \sum_{j\in\mathcal{I}:(j\in z)\wedge(i>j)} c_{z_iz_j} > 0. \tag{3.91}$$

3.6.4 Branch and Price

The optimal solution of the continuous relaxation of the RMP provides a lower bound on the problem. This solution is, however, rarely integral. As such, a branch-and-price algorithm is used to find the optimal integral solution of the MP. This algorithm is essentially a branch-and-bound algorithm, with the caveat that fixing the value of a variable in a node of the search tree triggers a new round of column generation for that node. This process ensures that as the search goes deeper in the tree (where nodes are more constrained, i.e., have more fixed columns) the relaxed solutions remain not only optimal, but also become tight.

As the tree is explored, nodes of two types are solved: a *branch node* whose solution is fractional and which will split into two further nodes by fixing a fractional column at 0 in one branch and at 1 in the other, and a *leaf node* whose solution is either integral or infeasible. An upper bound (the best integral solution found so far) helps early pruning: If the lower bound of a branch node is worse than the upper bound, this is a proof that the node cannot lead to the optimal solution, and it can be fathomed. Leaf nodes which are either infeasible or whose solutions are integral but worse than the upper bound can also be safely discarded. The upper bound is updated when an integral solution offered by a leaf node is better than the current bound. The search tree will eventually only contain leaf nodes and pruned branch nodes, after which optimality is proven.

For our problem, an initial upper bound is computed by enforcing the integrality constraints and solving the root node of the search tree. We know such an integral solution must exist since we started off with one for our initial columns. The search tree is explored by a combination of depth-first and breadth-first searches. Each iteration begins with the topmost lowest-cost node available and branches on its lowest-cost fractional column. Moving deeper in the tree level by level, all nodes at a level are considered and the algorithm branches on the lowest-cost node from that level. This branching heuristic thus partially explores several sub-trees in parallel while being guaranteed to reach at least one leaf node each iteration. The fractional column selected for branching is the one with the lowest cost, as it is assumed that it is more likely to make a significant difference.

3.7 Experimental Results

In this section, we report our extensive computational experiments comparing all models proposed and discussed.[14]

3.7.1 Methodology

Our test instances were generated similarly to the description provided by Bergman [50], with some minor adjustments. Items of weights $\{2, \ldots, 5\}$ (chosen uniformly and independently at random) are continuously added to the instance until their combined weights exceed instance size $\widehat{p} \in \{75, 100, 125, 150\}$. The last item is then removed, after which pairwise items are assigned a cost $\{M, -1, -3, -5\}$ with probability $\{0.4, 0.4, 0.1, 0.1\}$ (chosen independently at random). The number of knapsacks allotted to an instance is $\lfloor (\widehat{p}/10) + 0.5 \rfloor$. Our knapsacks are not bound by a capacity but rather by our balance constraints, i.e., we allow knapsacks to be “overfilled” as long as the balance constraints are not violated. Five instances of each size $\widehat{p}$ are generated.

The experiments were performed on dual core AMD 2.2 GHz processors with 8 GB of RAM running CentOS 7, and all models were constructed with IBM ILOG CPLEX Optimization Studio v12.8. Pareto sets are constructed of 10 ranges of deviations ($d_{\min}/d_{\max}$): $\{0/0, 1/1, \ldots, 9/9\}$ for the L_0-norm, $\{0/2, 2/4, \ldots, 18/20\}$ for the L_1- and L_2-norms, and $\{0/1, 1/2, \ldots, 9/10\}$ for the L_∞-norm. Each range of deviations is allotted a maximum of 360 seconds, meaning that the global time limit for an instance is 3600 seconds and involves solving 10 independent problems (thus construction of the Pareto sets is easily parallelizable).

In order to compare the impact of the load balancing parameters, we have opted to construct Pareto sets using the ϵ-constraint method [65]. Recall that parameters $d_{\min}$ and $d_{\max}$ denote the lower and upper bounds on the global deviation of a solution. Using these parameters, we bounded the global deviation at successively higher intervals, each

[14] All data and results are available at https://github.com/PhilippeOlivier/ijoc-qmkp

time solving the problem by minimizing the objective. Since we solve a succession of independent problems, we consider disjoint intervals of deviation to ensure nonoverlapping solution spaces in each step of the construction of the Pareto set, thus preventing identical solutions from being found in different steps. Presenting the parameters of the load balancing constraints with a Pareto set has the advantage of offering decision makers multiple optimal solutions to choose from depending on their perception of the trade-offs among them.

3.7.2 Discussion

The detailed computational experiments are reported in Tables 3.1–3.4. They compare the performance of the models according to the L_0-, L_1-, L_2-, and L_∞-norms. Since an instance is solved as a Pareto set, column *Opt. %* indicates the percentage of points from that set which were proven optimal. The most obvious observation is that $\mathrm{IP_B}$ is always superior to CP and $\mathrm{IP_A}$, except for the smallest instances where its performance is hampered by the overhead associated with its branch-and-price algorithm. The comparison between CP and $\mathrm{IP_A}$ is more subtle. For norms that both models are able to solve, $\mathrm{IP_A}$ generally solves more ranges of deviations to optimality in a shorter time than its counterpart. When neither reaches optimality, however, the CP model tends to find solutions of higher quality, as illustrated in Figure 3.1. The CP model is also very flexible. While $\mathrm{IP_A}$ is unable to handle the L_2-norm because of its nonconvexity, the CP model is not affected by such properties. It would thus be a better candidate if the problem were to become more complicated by adding nonconventional or unusual constraints. Furthermore, the CP model detects infeasibility very quickly (this can be observed when a nonzero number of points is proven optimal with the time being a multiple of 360[15]).

[15] For example, if 20% of the points are proven optimal after $\sim$2880 seconds, this implies that two of the problems were very quickly proven to be infeasible, since $3600 - 2 \times 360 = 2880$.

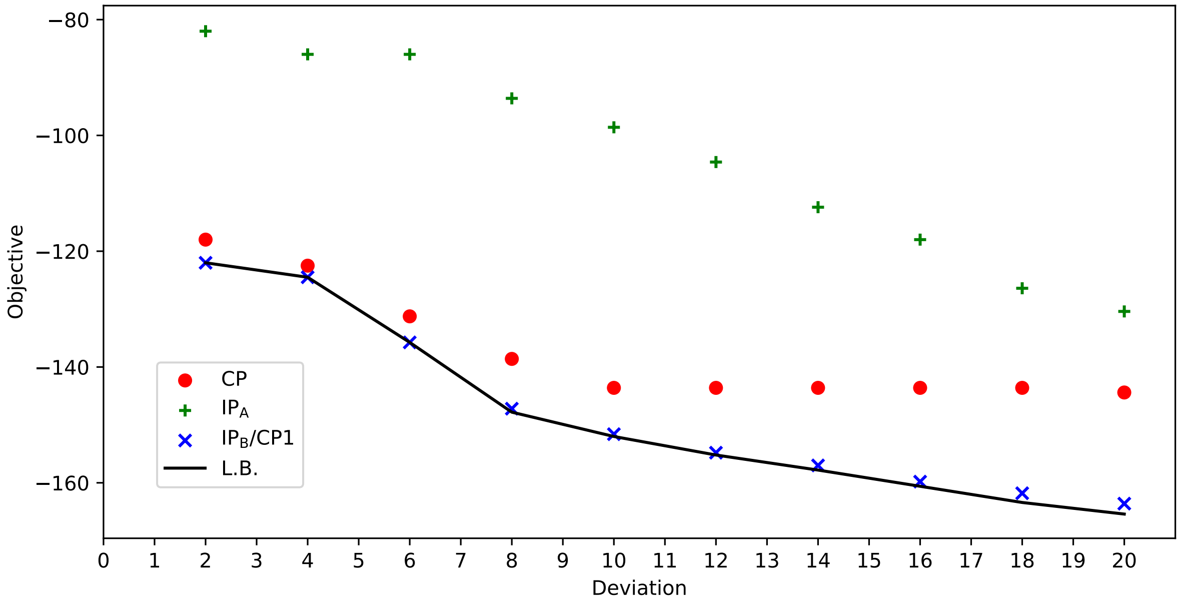


Figure 3.1 Pareto sets of the averaged results of the five instances of size $\widehat{p} = 150$, for the L_1-norm. Other norms show a similar trend.

The norm chosen to balance a problem has a major impact on the effort required to reach optimality. Solving problems optimally using the L_0-norm is difficult for all models. This norm only bounds the *number of deviative knapsacks* and not their *individual deviations*. As a result, as soon as a knapsack is allowed to deviate from the mean, it can account for a large *absolute* deviation, which leads to an explosion in the number of feasible item combinations for that knapsack. The L_∞-norm displays interesting results: For CP and IP$_\mathrm{A}$ solving is about as hard as for the L_0-norm, while for IP$_\mathrm{B}$ it is the easiest norm of all by a wide margin. This might be due to the fact that for the L_∞-norm in this model, all generated columns are feasible deviation-wise — only their item compositions constrain their compatibility with each other.

Reaching optimality with the L_1- and L_2-norms appears to be of similar difficulty for both the CP and IP$_\mathrm{B}$ models, which was unexpected. Indeed, for both models the complexity associated with the two norms is the same; albeit since the L_2-norm is by nature more constraining than the L_1-norm, its search space should be smaller and faster to explore. For both models, there appears to be a vague relation between the difficulties of both norms, i.e., if an instance is hard to solve with the L_1-norm it is likely to also be hard to solve with the L_2-norm, and vice-versa. However, our samples are in insufficient number to conclude on this with more certainty.

Table 3.1 Performance of the models for the L_0-norm.

				CP		**IP$_\mathbf{A}$**		**IP$_\mathbf{B}$/CP1**	
$\widehat{p}$	n	m	#	Opt. %	Time	Opt. %	Time	Opt. %	Time
50	13	5	1	**100**	**6.63**	**100**	**5.00**	**100**	**6.89**
50	12	5	2	**100**	**1.47**	**100**	**0.75**	**100**	**4.30**
50	15	5	3	**100**	**3.89**	**100**	**4.43**	**100**	**11.53**
50	13	5	4	**100**	**5.57**	**100**	**5.62**	**100**	**6.15**
50	13	5	5	**100**	**2.62**	**100**	**2.49**	**100**	**6.14**
75	21	7	1	30	2540.23	80	1446.29	**100**	**106.58**
75	20	7	2	40	2163.81	90	885.24	**100**	**61.86**
75	21	7	3	**100**	**63.55**	**100**	**701.11**	**100**	**41.36**
75	20	7	4	40	2163.31	**100**	**958.23**	**100**	**27.92**
75	23	7	5	20	2880.39	50	2406.13	**100**	**149.35**
100	29	10	1	0	3600.60	10	3710.38	**100**	**686.27**
100	28	10	2	0	3600.57	10	3457.23	**100**	**147.90**
100	28	10	3	0	3600.72	10	3465.22	**100**	**163.75**
100	28	10	4	0	3600.64	10	3336.29	**100**	**557.07**
100	24	10	5	0	3600.60	10	3423.31	**100**	**164.17**
125	38	12	1	0	3600.84	10	3291.92	50	2391.66
125	36	12	2	0	3600.87	10	3281.89	30	3036.98
125	33	12	3	0	3600.79	10	3375.65	20	3012.41
125	37	12	4	0	3601.01	10	3279.87	20	3118.37
125	33	12	5	0	3600.87	20	3002.05	70	1684.19
150	39	15	1	0	3601.47	10	3261.79	50	2246.76
150	42	14	2	0	3601.46	10	3306.48	30	2671.77
150	38	15	3	0	3601.31	10	3293.68	**100**	**974.34**
150	42	15	4	0	3601.47	10	3353.43	20	3052.48
150	40	15	5	0	3601.81	10	3285.62	20	2961.78

Table 3.2 Performance of the models for the L_1-norm.

				CP		$\mathbf{IP_A}$		$\mathbf{IP_B}$**/CP1**	
$\widehat{p}$	n	m	#	Opt. %	Time	Opt. %	Time	Opt. %	Time
50	13	5	1	**100**	**36.30**	**100**	**8.51**	**100**	**12.30**
50	12	5	2	**100**	**4.58**	**100**	**1.79**	**100**	**8.31**
50	15	5	3	**100**	**13.43**	**100**	**6.25**	**100**	**18.33**
50	13	5	4	**100**	**30.46**	**100**	**13.44**	**100**	**11.09**
50	13	5	5	**100**	**12.03**	**100**	**4.06**	**100**	**7.87**
75	21	7	1	20	2951.85	10	3287.36	**100**	**141.00**
75	20	7	2	10	3278.45	**100**	**1870.53**	**100**	**39.20**
75	21	7	3	**100**	**412.01**	**100**	**299.76**	**100**	**102.47**
75	20	7	4	10	3270.63	**100**	**1170.20**	**100**	**37.31**
75	23	7	5	10	3240.73	10	3275.31	**100**	**594.34**
100	29	10	1	20	2880.77	20	2921.54	**100**	**289.92**
100	28	10	2	20	2880.84	20	3029.81	**100**	**159.75**
100	28	10	3	10	3241.00	10	3599.62	**100**	**484.52**
100	28	10	4	20	2880.80	20	3021.73	**100**	**436.41**
100	24	10	5	20	2880.83	20	2921.83	**100**	**153.23**
125	38	12	1	20	2881.17	20	2909.61	50	2213.39
125	36	12	2	20	2881.14	20	2921.15	80	1022.79
125	33	12	3	20	2881.00	20	2906.17	80	1133.80
125	37	12	4	20	2881.58	20	2906.44	70	1315.00
125	33	12	5	0	3601.51	0	3717.72	**100**	**503.75**
150	39	15	1	20	2881.51	20	2919.57	80	1245.47
150	42	14	2	30	2521.19	30	2548.15	60	2020.61
150	38	15	3	10	3241.40	10	3273.41	60	2046.73
150	42	15	4	20	2881.56	20	2947.29	60	2094.31
150	40	15	5	0	3602.10	0	3639.66	90	1221.02

Table 3.3 Performance of the models for the L_2-norm.

				CP		$\mathbf{IP}_{\mathbf{A}}$[a]		$\mathbf{IP}_{\mathbf{B}}$**/CP1**	
$\widehat{p}$	n	m	#	Opt. %	Time	Opt. %	Time	Opt. %	Time
50	13	5	1	**100**	**7.01**	–	–	**100**	**13.45**
50	12	5	2	**100**	**1.80**	–	–	**100**	**9.92**
50	15	5	3	**100**	**3.64**	–	–	**100**	**23.88**
50	13	5	4	**100**	**5.69**	–	–	**100**	**22.46**
50	13	5	5	**100**	**5.03**	–	–	**100**	**29.37**
75	21	7	1	10	3261.35	–	–	**100**	**76.87**
75	20	7	2	30	3061.05	–	–	**100**	**50.99**
75	21	7	3	**100**	**135.84**	–	–	**100**	**75.35**
75	20	7	4	30	3001.85	–	–	**100**	**37.12**
75	23	7	5	0	3600.96	–	–	**100**	**455.67**
100	29	10	1	10	3241.12	–	–	**100**	**451.85**
100	28	10	2	10	3241.25	–	–	**100**	**163.50**
100	28	10	3	0	3601.06	–	–	**100**	**120.35**
100	28	10	4	10	3240.41	–	–	**100**	**293.36**
100	24	10	5	10	3240.48	–	–	**100**	**126.64**
125	38	12	1	10	3241.94	–	–	50	2143.08
125	36	12	2	10	3241.82	–	–	60	1929.41
125	33	12	3	10	3241.37	–	–	90	1182.97
125	37	12	4	10	3240.75	–	–	70	1591.46
125	33	12	5	0	3600.67	–	–	80	1707.88
150	39	15	1	10	3241.90	–	–	**100**	**952.71**
150	42	14	2	20	2882.36	–	–	80	1253.88
150	38	15	3	0	3602.41	–	–	**100**	**535.78**
150	42	15	4	10	3242.43	–	–	60	2274.94
150	40	15	5	0	3602.73	–	–	60	2005.08

[a] Results omitted since this model is unable to solve problems with this norm, see Section 3.5.3.

Table 3.4 Performance of the models for the L_∞-norm.

				CP		**IP$_\mathbf{A}$**		**IP$_\mathbf{B}$/CP1**	
$\widehat{p}$	n	m	#	Opt. %	Time	Opt. %	Time	Opt. %	Time
50	13	5	1	**100**	**13.90**	**100**	**6.64**	**100**	**5.14**
50	12	5	2	**100**	**2.52**	**100**	**1.64**	**100**	**3.08**
50	15	5	3	**100**	**7.62**	**100**	**5.87**	**100**	**8.02**
50	13	5	4	**100**	**11.23**	**100**	**7.18**	**100**	**4.49**
50	13	5	5	**100**	**4.74**	**100**	**4.17**	**100**	**5.17**
75	21	7	1	10	3306.89	50	2694.83	**100**	**43.34**
75	20	7	2	10	3304.28	**100**	**1079.16**	**100**	**22.56**
75	21	7	3	**100**	**95.62**	**100**	**124.08**	**100**	**23.90**
75	20	7	4	10	3302.98	**100**	**500.77**	**100**	**24.39**
75	23	7	5	0	3600.49	10	3526.89	**100**	**56.82**
100	29	10	1	0	3600.55	0	3814.61	**100**	**95.94**
100	28	10	2	0	3600.60	0	3699.25	**100**	**74.18**
100	28	10	3	0	3600.70	0	3763.05	**100**	**62.67**
100	28	10	4	0	3601.17	0	3870.28	**100**	**79.17**
100	24	10	5	0	3600.81	0	3640.24	**100**	**24.50**
125	38	12	1	0	3600.93	0	3633.59	**100**	**272.67**
125	36	12	2	0	3600.81	0	3630.22	**100**	**235.51**
125	33	12	3	0	3600.90	0	3700.69	**100**	**130.74**
125	37	12	4	0	3601.26	0	3646.64	**100**	**177.23**
125	33	12	5	0	3601.06	0	3687.49	**100**	**77.76**
150	39	15	1	0	3601.54	0	3677.97	90	469.56
150	42	14	2	0	3601.05	0	3673.37	**100**	**561.07**
150	38	15	3	0	3600.99	0	3650.54	**100**	**141.05**
150	42	15	4	0	3601.23	0	3659.79	90	656.60
150	40	15	5	0	3601.16	0	3643.91	90	493.69

Instances in which all pairwise costs are zero, independent of the number of conflicts, can usually be proven optimal by the CP model in a fraction of a second for all instance sizes. This property is due to the aggressive filtering of the `dispersion` constraint, which is very effective for strict feasibility problems. By extension, the IP_B model also displays this property since the CP model generates its initial columns (and without pairwise costs any feasible solution is optimal).

Table 3.5 compares the four subproblems of IP_B (CP1, CP2, IP1, and IP2). These are solved to optimality without a time limit. To keep results within a reasonable time span, testing was done on the five instances of size $\widehat{p} = 50$ for the L_0-norm, and of size $\widehat{p} = 100$, for the other norms. Using the four pricing problems of model IP_B with every norm, we have solved a sample of the instances to optimality without a time limit. The normalized results of Table 3.5, summarized in Table 3.6, show that CP1 is several times faster than the other pricing problems. We were surprised to find that this unusual 0–1 CP formulation proved to be the best. We are aware that CPLEX Optimization Studio performs many special optimizations behind the scenes, and for a time we suspected that it could have linearized this model and solved it as an IP. However, it turns out that this model is wholly solved by a CP solver.[16]

For IP_B, the branch-and-price algorithm provides a proof of optimality, given enough time. Yet, the relaxation bound is fairly tight at the root of the tree. Suppose that we solve the relaxation of the root node, generating some columns in the process, and that we then enforce integrality constraints on those columns. The relative optimality gap between the relaxation bound and the integral solution can be found in Table 3.7. As a general remark, this gap correlates with the balancing norm used, as can be observed from the gaps of L_∞-norm, which are much smaller than those of the other norms.

Table 3.6 Computation times of IP_B pricing problems, normalized w.r.t. $\text{IP}_\text{B}/\text{CP1}$.

	IP_B/CP1	**IP_B/CP2**	**IP_B/IP1**	**IP_B/IP2**
L_0-norm	1	31.6	6.3	–
L_1-norm	1	30.0	2.0	–
L_2-norm	1	17.1	4.3	–
L_∞-norm	1	125.7	6.9	–

[16] Confirmed through personal communication with Petr Vilím of IBM.

Table 3.5 Comparison of subproblems for model $\mathrm{IP_B}$.

					$\mathrm{IP_B}$/CP1	**$\mathrm{IP_B}$/CP2**	**$\mathrm{IP_B}$/IP1**	**$\mathrm{IP_B}$/IP2**[a]
	$\widehat{p}$	n	m	#	Time	Time	Time	Time
L_0-norm	50	13	5	1	4.53	119.00	35.99	–
	50	12	5	2	2.40	15.94	9.83	–
	50	15	5	3	7.12	260.02	41.71	–
	50	13	5	4	3.51	233.18	24.53	–
	50	13	5	5	4.23	60.30	26.10	–
L_1-norm	100	29	10	1	237.94	13728.71	502.17	–
	100	28	10	2	114.92	4779.30	358.10	–
	100	28	10	3	368.34	5080.59	636.67	–
	100	28	10	4	327.73	9986.30	488.41	–
	100	24	10	5	117.16	1395.20	313.80	–
L_2-norm	100	29	10	1	459.68	11059.23	1285.34	–
	100	28	10	2	155.10	680.28	653.67	–
	100	28	10	3	95.47	2313.29	648.13	–
	100	28	10	4	244.09	3397.70	1467.96	–
	100	24	10	5	81.81	294.48	401.18	–
L_∞-norm	100	29	10	1	65.22	17215.69	394.65	–
	100	28	10	2	45.07	2510.51	308.96	–
	100	28	10	3	40.38	2087.57	274.43	–
	100	28	10	4	45.94	4853.30	308.75	–
	100	24	10	5	18.47	358.65	187.82	–

[a] This model performed so poorly that it would have been impractical, timewise, to have it solve the instances—it has instead been omitted.

Table 3.7 Average relative optimality gap (%) at the root node for $IP_B/CP1$.

$\widehat{p}$	L_0-norm	L_1-norm	L_2-norm	L_∞-norm
50	1.02	3.62	11.13	0.05
75	1.98	2.25	2.63	0.29
100	1.78	1.40	1.71	0.38
125	3.19	0.84	1.68	0.15
150	3.20	1.06	1.26	0.31

The offline optimization of knapsack load bounds (3.40)–(3.47) makes a noticeable difference for both IP models, but is of no use for the L_1- and L_2-norms of the CP model, as the `dispersion` constraint already ensures that knapsack loads be consistent with $d_{\max}$. Finally, our simple approach to the generation of the Pareto sets for both IP models could be improved [66], although this is outside of the scope of the present paper.

As stated in Section 3.1, multiple metaheuristics-based software solutions exist to solve the problem of seating guests at events. One of the design goals of these services is to solve the problem *quickly* since clients could easily grow impatient after waiting just a few seconds in front of a seemingly unresponsive program or browser window. For this reason, exact methods may not seem, initially, to be the best candidates for this application. Nevertheless, Bergman [50] has shown that a branch-and-price algorithm is suitable to solve instances of a practical size to optimality in reasonable time, as he is able to solve instances of size $\widehat{p} = 150$ in a few seconds and instances of size $\widehat{p} = 300$ in several minutes. By removing the balancing aspect of $IP_B/CP1$ and solving similar instances, our model exhibits equivalent performance. Furthermore, we have shown in our previous paper [58] that a CP model using a custom branching heuristic and a LNS strategy, while not proving optimality, can also compete with metaheuristic methods on instances of practical size. All models presented in this paper exhibit a strong convergence after just a few seconds. The CP and IP_A models converge to fairly good solutions (albeit generally not particularly close to optimality for large instances), after which further improvements in solution quality become difficult. The $IP_B/CP1$ model converges to near optimality quickly, but proving optimality is not always easy. Comparative testing against commercial software solutions would probably conclude on the feasibility of using models such as those presented in this paper for commercial purposes.

Our methods contrast with that of Bergman [50]. Indeed, the latter approach makes more sense for event seating, as the capacity of the tables is typically a hard constraint, with the objective being to pack the tables to capacity or near-capacity as profitably as possible. Our method is a more general one, and while the objective is the same, the constraints aim for the balancing of under- and over-capacity knapsacks.

3.8 Conclusion

In this paper we presented three exact methods to solve a generalized version of the QMKP. We showed that a column generation algorithm with the pricing problem solved by CP is the most appropriate solution method in virtually all situations. The various tests that were performed demonstrated that the balancing norms have considerable yet disparate impacts on performance. These norms do not necessarily influence the models in the same way. An exact method similar to our column generation model could potentially be an alternative to metaheuristics for commercial applications.

CHAPTER 4 ARTICLE 2: MEASURES OF BALANCE IN COMBINATORIAL OPTIMIZATION

Philippe Olivier, Andrea Lodi, and Gilles Pesant

Accepted for publication in *4OR* (May 2021)

Abstract. The concept of balance plays an important role in many combinatorial optimization problems. Yet there exist various ways of expressing balance, and it is not always obvious how best to achieve it. In this methodology-focused paper, we study three cases where its integration is deficient and analyze the causes of these inadequacies. We examine the characteristics and performance of the measures of balance used in these cases, and provide general guidelines regarding the choice of a measure.

4.1 Introduction

It is natural to think of *balance* as *things being as equal as possible*. Yet this notion of equality is hard to define. Suppose that we have candy bags of assorted sizes (5, 5, 6, 7, 9, 12, and 12 candies) which we want to distribute fairly to four children. We easily observe that a perfect distribution of 14 candies per children is not possible. What, then, constitutes a fair distribution?

We could consider as our criterion of fairness that the largest share of candies be as small as possible, which would give us handouts of 12, 12, 16, and 16 candies. Or we could instead consider an alternative criterion and ensure that the sum of candy discrepancies from the mean is minimal, giving us handouts of 12, 13, 14, and 17 candies. We can notice that the optimal solution for one criterion of fairness is not optimal for the other, and vice-versa. These options are both *fair*, yet neither is intrinsically better or worse than the other.

This simple example illustrates how the notion of balance becomes more ambiguous after scratching the surface. Besides, this notion deserves special attention, as fairness is of paramount importance in several practical situations. In many jurisdictions of the United States, for instance, algorithms have taken the role of decision-makers for delicate matters such as deciding whether a defendant awaiting trial should be released or not. One study found that African-Americans were one and a half times more likely than Caucasians to be wrongly classified as high risk by one of these algorithms [6]. Racial disparities being a sensitive issue, these algorithms need to ensure fairness in this process [7].

While this paper is concerned with balance in the context of combinatorial optimization, issues of fairness also arise in the related field of game theory, where some criteria of fairness are of a different nature. *Envy-freeness* ensures that no player would want to trade their share for that of another, *Pareto efficiency* guarantees that no share can be improved without worsening some other share, and so on [1].

A tangential application combining balance types and fairness criteria is found in *social welfare functions*. These functions describe the collective welfare of a society based on the utilities, or satisfaction, of its individuals [67]. The utilitarian function measures collective welfare as the sum of all individual utilities, maximizing pure utility while disregarding any type of equality between individuals. In contrast, the egalitarian function considers the minimum of all individual utilities at the expense of a lower level of global welfare, putting everyone on an equal (albeit usually lower) footing [68]. The Nash social welfare function maximizes the product of all utilities, which provides a sort of middle ground between the two previous functions [69].

In this methodology-focused paper, we present three cases supporting the hypothesis that unfamiliarity with respect to the characteristics of measures of balance often leads to poor choices regarding modeling in combinatorial problems. We examine the characteristics of several measures, and provide general guidelines regarding the choice of an appropriate measure. We do this by keeping in mind two solving paradigms for combinatorial optimization, namely *constraint programming* (CP) and *integer programming* (IP), so as to gain, in addition, some understanding of the pros and cons of the analyzed measures with respect to the solution methods.

To start with, Section 4.1.1 briefly introduces CP and IP. Section 4.2 presents a few measures of balance and their characteristics, and outlines their use in the context of CP and IP. Three problematic cases are studied in Section 4.3. Finally, Section 4.4 discusses general guidelines for the application of the various measures of balance.

4.1.1 CP and IP Considerations

Constraint Programming is a programming paradigm for solving combinatorial problems. It is a form of declarative programming wherein an abstract model is constructed and then handled by a solver. This model is defined by the problem variables, their domains, and a set of constraints defining the relation between subsets of variables, restricting certain variable-value combinations. A solution to this model is a tuple of values consistent with the domains of the variables, and which satisfies the constraints.

There are several types of constraints: the `alldifferent` constraint, for example, ensures that a subset of variables are each assigned distinct values. Each constraint achieves its purpose through the action of filtering—the process of removing values inconsistent with the constraint from the domains of variables. While filtering reduces the domains of the variables, it is not enough to solve the problem by itself. Hence, a search tree is constructed, and is explored by branching on variables at each node, i.e., instantiating a variable with a specific value in its domain. Proving optimality in CP amounts to dynamically adding constraints during search, such that for a solution to be feasible it must be strictly better than the best one found so far. The reader is referred to [70] for a detailed methodological overview on CP.

The Integer (Linear) Programming paradigm is based on modeling by means of a linear objective function to be optimized over a set of linear constraints. The constraints are defined upon decision variables, a subset of which is restricted to take only discrete (integer) values. This latter requirement makes the problem hard from a complexity standpoint because the resulting feasible space is nonconvex. For this reason, IP technology is based on the simple idea of relaxing the integrality requirements on the variables and iteratively solving the so-called continuous relaxation, which, on the contrary, is easy, i.e., it is solvable in polynomial time. The value of the optimal solution of the continuous relaxation provides a dual bound on the optimal solution of the original IP. Several algorithmic techniques, like preprocessing and cutting planes, enhance the quality of the continuous relaxation with respect to the convex hull of (mixed-)integer solutions and are fundamental building blocks of the IP technology. Conversely, the basic solution scheme requires to enforce integrality by means of a divide-and-conquer algorithm called branch and bound. Finally, feasible solutions for the overall problem are computed throughout the process by primal heuristics, thus providing a primal bound and making the overall algorithm converge by closing the gap between primal and dual bounds. The reader is referred to [71] for a detailed methodological overview on IP.

In this paper, we are concerned with the quality of the solutions rather than the difficulty of finding them, which is partly influenced by the method used to solve the problem. CP and IP are two approaches that are distinct in nature. As such, the different ways of computing balance have a varying influence on these methods. A CP solver does not assume any particular property of the solution space (such as convexity), and the various measures of balance generally have a similar impact on the performance of the model. IP, on the other hand, has a strong preference for linear modeling, as a linear model can generally be solved much more efficiently than a nonlinear one.[1] The difficulty in comparing CP and IP is further exacerbated by other factors, such as the type of problem being solved, or one's modeling choices. Symmetry breaking constraints, for example, tend to have a major impact on performance; yet their behavior may vary wildly in CP and IP due to the different nature of these models. For these reasons, we will not be directly comparing the performance of CP and IP on the problems presented in this paper although some hints on solution aspects will be gained indirectly.[2]

4.2 Measures of Balance

There is little uniformity with respect to the labels associated with all things balance in the literature. Marsh and Schilling [14] use the term *equity*, which they equate to *fairness*. They describe their objective of maximizing equity through the process of minimizing *inequities*. Equity is also considered equivalent to fairness by Ogryczak [73], but maximizing equity is achieved by minimizing *inequalities*. This latter author further refers to the former authors' *inequities* as *inequalities*. In neither case is *balance* mentioned. Pesant [27] states that "in rostering we usually talk of fairness instead of balance, because of the human factor." However, it is not uncommon to find the term *balance* in such contexts [56]. In the present paper, we have opted to use *balance* as an umbrella term encompassing everything that should be fair and, in general, *as equal as possible*.

Given a finite collection of real variables $X = \{x_1, x_2, \ldots, x_n\}$, its *balance* has been defined in several ways in the literature. Some measures only take into account the extremal values, the rationale being that constraining the most extreme points forces the others into a shorter interval. Other measures consider all values, and while usually more computationally expensive this often results in an improved distribution (relative to the aims of the problem). We note that for some combinatorial optimization problems, the term *balanced* is often understood as minimizing the difference between the largest and smallest costs. This is especially true for some of the more well-known problems such as the traveling salesman problem (see, e.g., [31,74]). In this paper, we use the term *balance* more liberally, and we do not tie it to a specific method of achieving balance. This section covers four common measures of balance.

[1] Balancing with L_2-DEVIATION rather than L_1-DEVIATION turns an IP model into a (Mixed-)Integer Nonlinear Programming one (MINLP). Although impressive advances in solving MINLPs have been made in the last 15 years [72], MINLPs are still, in general, significantly more difficult to solve than ILPs.

[2] In this paper, the models presented in Appendices B, C, and D were solved via CP.

The MINMAX measure is rather crude and simply minimizes the maximum value

$$\min \max_{i=1}^{n} x_i.$$

For a collection of n points, the global distance between these points and their arithmetic mean μ, according to a norm p, is defined by the concept of L_p-deviation

$$\sum_{i=1}^{n} |x_i - \mu|^p.$$

In particular, L_1-DEVIATION minimizes the sum of absolute deviations from the mean

$$\min \sum_{i=1}^{n} |x_i - \mu|,$$

L_2-DEVIATION minimizes the sum of squared deviations from the mean

$$\min \sum_{i=1}^{n} (x_i - \mu)^2,$$

and L_∞-DEVIATION minimizes the maximum deviation from the mean

$$\min \max_{i=1}^{n} |x_i - \mu|.$$

By using definitions that are standard in statistics [75], we now introduce some characteristics exhibited by these measures of balance. Namely,

- The *dispersion* represents the size of the interval within which the points are located, and by extension is one measure of the sensitivity to outliers.[3]
- A distribution is *smooth* when its points appear evenly within this interval.
- The number of *outliers* near the edges of the dispersion interval is another measure of the sensitivity to outliers.

The above intuitive concepts are defined and discussed formally in Appendix A. In the remainder of the paper, they are used to characterize the performance of the four balance measures in the three case studies that we present.

[3] In this paper, we call an outlier any value equal to one of the two extremal values of the dispersion interval.

Generally, when optimized with MINMAX, values are only bounded on one side, and as such they show the largest dispersion. The other measures force bounds on both sides, with L_2- and L_∞-DEVIATION forcing especially tight bounds by nature. MINMAX offers little smoothness as many values will tend to be grouped around the bound. L_∞-DEVIATION is in contrast smoother—nothing is constraining the deviations apart from forcing them to be within the interval, so their associated values will appear somewhat randomly within this interval. Results are more varied for L_1- and L_2-DEVIATION, since there is a natural bias for values to be closer to the mean. The lack of minimum bound for MINMAX makes it robust against outliers, as does the linear expression of deviation for L_1-DEVIATION. Outliers have more influence on L_∞-DEVIATION since both small and large values disproportionately affect the objective, and are also more impactful on the quadratic expression of deviation of L_2-DEVIATION.

Marsh and Schilling [14] have surveyed measures of equity, in particular related to facility location. The authors record and briefly analyze some 20 measures of balance in use in various fields. They propose some guidelines to help in choosing a measure.

Balancing in constraint programming is usually achieved through special constraints. L_1-DEVIATION is handled by the `deviation` constraint, introduced by Schaus et al. [23]. Pesant and Régin [20] balanced with L_2-DEVIATION using the `spread` constraint. The `dispersion` constraint proposed by Pesant [27] encapsulates multiple measures of balance, including L_1-, L_2-, and L_∞-DEVIATION. Other measures, such as MINMAX, can be expressed with classical constraints such as `minimum` and `maximum`.

There does not seem to be any substantial work on general balancing techniques in the context of integer programming, outside of problem-specific cases. This is partly due to the fact that balancing in IP cannot be decoupled from a model as in CP, where balancing is tightly wrapped in a single constraint that can be generically reused. Early work on balancing in mathematical programming includes a short paper by Gaudioso and Legato [29] presenting a few balancing measures, among which MINMAX. Some examples include a mixed-integer linear program with L_1-DEVIATION as its objective that has been used to balance the loads on servers [76], and a quadratic integer program balancing completion times of jobs using L_2-DEVIATION [77]. A recent paper by Olivier et al. [78] covers the L_1-, L_2-, and L_∞-DEVIATION measures in the context of IP, and compares these with equivalent CP approaches, with special attention to quadratic versions of knapsack and bin packing problems.

4.3 Case Studies

This section presents practical problems that require some form of balancing: the assignment of courses to periods such that the loads of the periods are balanced, the assignment of patients to nurses such that the workloads of the nurses are balanced, and the distribution of bikes to stations in a bike sharing system such that the stations are balanced. We argue that when these problems were initially introduced, their measures of balance were deficient; we will show how they have been improved.

4.3.1 Balanced Academic Curriculum Problem

The *Balanced Academic Curriculum Problem* (BACP) attempts to find an assignment of courses over a number of periods such that the academic load of a student is balanced throughout the curriculum and that course prerequisite constraints are respected. Let[4]

- $\mathcal{P} = \{1, \dots, m\}$ be the index set of periods,
- w denote the combined loads of all the courses,
- $L = \{L_1, \dots, L_m\}$ denote the loads of the periods for an assignment.

The objective is to maximize the balance of an assignment of the n courses to the m periods. The BACP was originally introduced by Castro and Manzano [57], whose CP and IP models both achieved balance by minimizing the maximum academic load of the periods (MINMAX). Further papers by Hnich et al. [79,80] introduced new CP and IP models using the same balancing criterion. Monette et al. [81] not only used MINMAX but also explored other options, namely balancing using the L_1-, L_2-, and L_∞-DEVIATION measures, all with a CP model. The four objectives, in minimization form, studied by Monette et al. can be formalized as

$$\max_{k \in \mathcal{P}} L_k \tag{MINMAX}$$

$$\sum_{k \in \mathcal{P}} |L_k - w/m| \tag{L_1-DEVIATION}$$

$$\sum_{k \in \mathcal{P}} (L_k - w/m)^2 \tag{L_2-DEVIATION}$$

[4] A complete model for the BACP can be found in Appendix B.

$$\max_{k\in\mathcal{P}} |L_k - w/m| . \qquad (L_\infty\text{-DEVIATION})$$

Starting with the premise that "neither criterion subsumes the others and there is no *a priori* reason to prefer one of them" [81], Monette et al. aim to determine how well each balance criterion approximates the others. Their findings[5] are reproduced in Table 4.1, where rows represent optimized criteria and columns represent evaluated criteria. For example, at the intersection of row "L_1-DEVIATION" and column "MINMAX" is the value 2.63. This means that if the problem is optimized with respect to L_1-DEVIATION, and that we then evaluate MINMAX on that solution, it is on average 2.63% higher than if the problem was optimized with respect to MINMAX. In other words, optimizing a problem with respect to L_1-DEVIATION is a decent approximation of MINMAX, as that solution is on average only 2.63% worse than optimizing directly with MINMAX. The average of a row represents how well the balance criterion approximates the others in general, while the average of a column represents how well the balance criterion is approximated by the others in general.

Monette et al. observed that the optimal solutions of L_2-DEVIATION were often also optimal for the other measures, and thus that this measure was generally a good approximation of the others. For this reason, the authors conclude that L_2-DEVIATION is the superior measure of balance for the BACP. Further publications by various authors on this problem and its variants also use L_2-DEVIATION (see for example [27, 82, 83]). We have conducted similar experiments[6] as Monette et al., and reached comparable conclusions. Details of our findings can be found in Table 4.2.

[5] Guidelines to generate equivalent instances to those used can be found in [81].

[6] Our dataset can be found at `https://github.com/PhilippeOlivier/mobico`.

Table 4.1 Comparison of the balance criteria for the BACP (reproduced from [81]).

	MINMAX	L_1-DEV.	L_2-DEV.	L_∞-DEV.	Average
MINMAX	0.00	10.62	16.53	0.06	9.07
L_1-DEVIATION	2.63	0.00	6.27	0.12	3.00
L_2-DEVIATION	0.28	0.00	0.00	0.00	0.09
L_∞-DEVIATION	10.37	18.07	23.66	0.00	17.36
Average	4.43	9.56	15.48	0.06	

Table 4.2 Characteristics of the solutions of the BACP. The first column displays the normalized size of the dispersion intervals. In the second column, smoothness is described as the Wasserstein distance (see Appendix A for details) between the distribution of the solutions and a perfectly smooth distribution (lower is smoother). The last column shows the average percentage of values that are outliers (an outlier is characterized as being any value equal to the lower or upper bound of the dispersion interval).

	Dispersion	Smoothness	Outliers
MINMAX	2.25	1.69	59.40%
L_1-DEVIATION	1.01	1.82	14.90%
L_2-DEVIATION	1.00	1.77	17.10%
L_∞-DEVIATION	1.07	1.56	18.50%

Takeaway

The dispersion interval is more than twice as large for MINMAX than it is for the other measures. Since MINMAX does not use the mean as a point of reference in balancing, low values have no impact on the solutions, yet they may significantly increase the dispersion interval. No measure of balance offers solutions with a smooth distribution of values—this is understandable, as no measure for the BACP is better satisfied by spreading values evenly in the dispersion interval. MINMAX shows many more outliers than the other measures since, for this measure, the values tend to be grouped around the upper bound of the dispersion interval. As did Monette et al., we conclude that for the BACP, the L_2-DEVIATION performs very well but, considering in close detail Table 4.2, we also notice that the L_1-DEVIATION is an excellent balance choice. They both offer short dispersion intervals, fairly few outliers, and provide a good appoximation of the other measures.

4.3.2 Nurse-Patient Assignment Problem

The *Nurse-Patient Assignment Problem* (NPAP) seeks to assign patients to nurses within different zones in a hospital. The patients have various acuities, and should be assigned so as to best balance the workload among the nurses. The workload of a nurse is defined by the sum of their patients' acuities. Let[7]

- $\mathcal{N} = \{1, \ldots, n\}$ be the index set of nurses,
- $\mathcal{P} = \{1, \ldots, m\}$ be the index set of patients,

[7] A complete model for the NPAP can be found in Appendix C.

- a denote the combined acuities of all the patients,
- w_j be the workload of nurse j.

The patients are located in different zones, and as such the NPAP is twofold: Nurses must first be assigned to zones, and then patients to nurses. The objective is to find a staffing of nurses to zones combined with a nurse-patient assignment maximizing the balance of the nurses' workloads. The objectives, in minimization form, can be formalized similarly as for the BACP

$$\max_{j \in \mathcal{N}} w_j \tag{MINMAX}$$

$$\sum_{j \in \mathcal{N}} \left| w_j - a/m \right| \tag{L_1-DEVIATION}$$

$$\sum_{j \in \mathcal{N}} \left(w_j - a/m \right)^2 \tag{L_2-DEVIATION}$$

$$\max_{j \in \mathcal{N}} \left| w_j - a/m \right| . \tag{L_∞-DEVIATION}$$

The NPAP was introduced by Mullinax and Lawley [56], whose IP model expressed the measure of imbalance for a zone as the difference between its nurses' lightest and heaviest workloads. The objective was then to minimize the sum of imbalances for all the zones. Schaus et al. [84] have shown that while the previous model may do a good job in balancing the workloads within each zone, its objective function is deficient as this does not necessarily translate into a good balance of workloads between the different zones. The authors constructed CP models to solve this problem, and considered the L_1- and L_2-DEVIATION measures to minimize either the absolute or squared deviations of the workloads. They conclude that L_2-DEVIATION is more appropriate for the NPAP due to its increased sensitivity to outliers.

We have conducted similar experiments[8] on the NPAP as Monette et al. [81] did for the BACP by adapting the CP model of [85] with the four objectives. Our results are reported in Table 4.3.

The two problems share some similarities but are nevertheless unique in their own way. The BACP imposes assignment restrictions in the form of course prerequisites, while in the NPAP these restrictions are embedded in the staffing problem. Both problems have

[8] Our dataset can be found at `https://github.com/PhilippeOlivier/mobico`.

Table 4.3 Comparison of measures of balance for the NPAP.

	MINMAX	L_1-DEV.	L_2-DEV.	L_∞-DEV.	Average
MINMAX	0.00	0.61	7.45	22.81	7.72
L_1-DEVIATION	0.19	0.00	3.79	18.07	5.51
L_2-DEVIATION	0.42	0.87	0.00	1.30	0.65
L_∞-DEVIATION	0.35	0.94	0.29	0.00	0.40
Average	0.24	0.60	2.88	10.55	

similar assignment ratios (five courses per period and six patients per nurse, on average), but the range of patient acuities in the NPAP is much wider than the range of course credits in the BACP. L_∞-DEVIATION is the best approximator for the NPAP and the worst for the BACP, indicating that it is sensitive to the problem type. In contrast, L_2-DEVIATION is a very good approximator for both problems, suggesting robustness against various types of problems. As a general rule, MINMAX itself is not a very good approximator, but it can be well-approximated by the other measures. The opposite is true for L_2-DEVIATION, which is usually a good approximator for other measures of balance but which does not tend to be approximated very well most of the times.

Table 4.4 reports quantitative measures of dispersion, smoothness and outliers (as in Table 4.2 for BACP).

Takeaway

As shown in Table 4.4, the dispersion intervals of MINMAX and L_1-DEVIATION are, on average, around 10% larger than those of L_2- and L_∞-DEVIATION. A short dispersion interval is always preferable for the NPAP, as it ensures that the workloads are never too far apart from each other even when there is a lot of variation between them. While almost a fifth of the workloads are outliers with all measures of balance, they have a limited impact on L_2- and L_∞-DEVIATION due to their shorter dispersion intervals. As for the

Table 4.4 Characteristics of the solutions of the NPAP.

	Dispersion	Smoothness	Outliers
MINMAX	1.11	1.03	19.27%
L_1-DEVIATION	1.09	1.04	18.81%
L_2-DEVIATION	1.01	1.06	19.60%
L_∞-DEVIATION	1.00	1.03	19.86%

BACP, we observe that none of the measures offer smooth solutions, for reasons similar to those of the previous problem. We conclude that for the NPAP, the L_2- and L_∞-DEVIATION measures are preferrable, due to their short dispersion intervals and tight approximation of the other measures.

4.3.3 Balancing Bike Sharing Systems

A bike sharing system is a service that allows users to pick up and return bikes from and to bike stations around a city. This system is prone to imbalance since a station may not see the same number of bikes picked up and returned. To mitigate this issue, a fleet of vehicles rebalances the stations by redistributing the bikes more evenly between them. The *Balancing Bike Sharing Systems* (BBSS) problem aims to find optimal tours of the fleet of vehicles to rebalance bike stations.

While many variants of the BBSS problem can be found in the literature, a popular definition involves the minimization of a weighted combination of bike deviations from a target (using L_1-DEVIATION) and some routing cost combining the time required for loading/unloading and driving. This definition, or something very close to it, can be found in [86–90]. Several other definitions of the problem also make use of L_1-DEVIATION (see, e.g., [91, 92]). The static version of the BBSS problem, as presented here, assumes that the impact of customers using the service is negligible, as it would be if rebalancing was done during the night. For the purpose of this paper, we will limit ourselves to the static version of the BBSS problem, although a dynamic version of this problem has also been studied [91].

In this section, we solve the BBSS problem variant mostly as it is presented in the previous paragraph. To facilitate the analysis of the results, we move the routing cost of the objective into the constraints to isolate the balancing factor from the rest. Let[9]

- $\mathcal{S} = \{1, \dots, S\}$ be the set of S stations,
- for a station $s \in \mathcal{S}$, b_s be its initial number of bikes, t_s its target number of bikes, and $service_s$ the number of bikes that have been added or removed from it.

Several vehicles originating from their depots move bikes between stations; These capacitated vehicles are constrained by the duration of their routes. We consider the following alternatives to express our minimization balancing objective :[10]

[9] A complete model for the BBSS problem can be found in Appendix D.

[10] In contrast with the BACP and the NPAP, in the following objectives the mean is implicitly taken into account, since it is always 0.

$$\max_{s \in \mathcal{S}} (b_s + service_s - t_s) \tag{MINMAX}$$

$$\sum_{s \in \mathcal{S}} |b_s + service_s - t_s| \tag{L_1-DEVIATION}$$

$$\sum_{s \in \mathcal{S}} (b_s + service_s - t_s)^2 \tag{L_2-DEVIATION}$$

$$\max_{s \in \mathcal{S}} |b_s + service_s - t_s| \,. \tag{L_∞-DEVIATION}$$

We posit that in the case of the BBSS problem, L_2-DEVIATION is preferable to L_1-DEVIATION. From a practical standpoint, a station is fully functional if it can both provide and accept bikes. As such, it is more desirable to have all stations slightly unbalanced rather than to have most stations very balanced with a few very unbalanced ones. The system as a whole provides a better service in the former case than in the latter, as it boasts more functional stations. This particular understanding of system functionality is found elsewhere in the literature. The objective function of [93], which in part minimizes the number of events associated with a lack or an excess of bikes, suggests that these authors endorse this position. The constraining of station inventory to be within a serviceable interval, as found in [94], indicates that these authors also share a similar opinion. The scenario presented in Example 2 illustrates how L_2-DEVIATION has an increased tendency to prevent substantial imbalances when compared to L_1-DEVIATION.

Example 2 *In Fig. 4.1, the vehicle must depart from and arrive to depot D with a load of 0. With a vehicle capacity of 1 and a maximum duration of 4, the two feasible solutions are D–S_1–S_2–D (*$service_1 = -1$ *and* $service_2 = 1$*) and D–S_1–S_3–D (*$service_1 = -1$ *and* $service_3 = 1$*). Using* L_1-DEVIATION, *the solutions have equivalent objectives of* $0+0+2=2$ *and* $0+1+1=2$, *since for this norm unloading a bike at either S_2 or S_3 achieves the same purpose. With* L_2-DEVIATION, *however, the solutions have objectives of* $0^2+0^2+2^2=4$ *and* $0^2+1^2+1^2=2$. *Indeed, provided that other things are equal, it is preferable for this norm to visit the most unbalanced stations, as they have an increased impact on the objective.* ■

We generated 100 instances[11] of 10 stations each, served by a single vehicle and using a single depot.[12] The demands of the stations range from -10 to 10 and they sum to 0 in each instance, i.e., an uncapacitated vehicle unconstrained by the duration of its route

[11] Our dataset can be found at `https://github.com/PhilippeOlivier/mobico`.

[12] We are forced to keep the problem size small in order to solve all instances to optimality in a reasonable amount of time, for illustrative purposes.

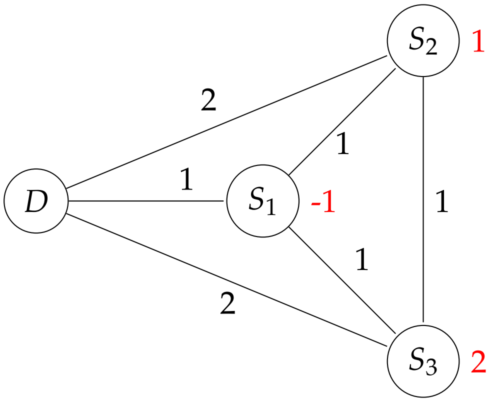


Figure 4.1 A simple BBSS problem. The demands of the stations $(t_s - b_s)$ are in red (a negative demand implies that the station has an excess of bikes).

would perfectly balance all stations. We solve these instances with a single vehicle in three different contexts: constrained vehicle capacity and unconstrained route length, unconstrained vehicle capacity and constrained route length, constrained vehicle capacity and constrained route length.[13] All instances are solved to optimality with the CP model of [86], which was slightly adjusted to fit our altered objective functions.

Fig. 4.2 shows the normalized distributions of bike deviations from the balance targets of their stations, in the three contexts mentioned previously. The fitted curves indicate that L_2-DEVIATION offers bike deviations slightly more closely grouped near the mean and with fewer outliers than L_1-DEVIATION. While the former measure offers an arguably marginal improvement in solution quality over the latter, this improvement is constant across variously constrained instances. As for the BACP and NPAP previously, Table 4.5 shows that L_2-DEVIATION approximates L_1-DEVIATION much better than the reverse.

Similarly to the previous two sections, Table 4.6 shows how the various measures of balance approximate each other, and Table 4.7 examines their characteristics. Considering

[13] When constrained, the capacity is set to 5, which is half the maximum demand of a station. When constrained, the route length is set to 100, which is roughly half the length of an optimal TSP tour of the stations.

Table 4.5 Approximation quality of L_1- and L_2-DEVIATION with respect to each other when the vehicle **C**apacity and/or **R**oute length are constrained.

	C	R	C&R
L_1-DEVIATION	15.45	7.97	9.18
L_2-DEVIATION	1.84	1.78	1.22

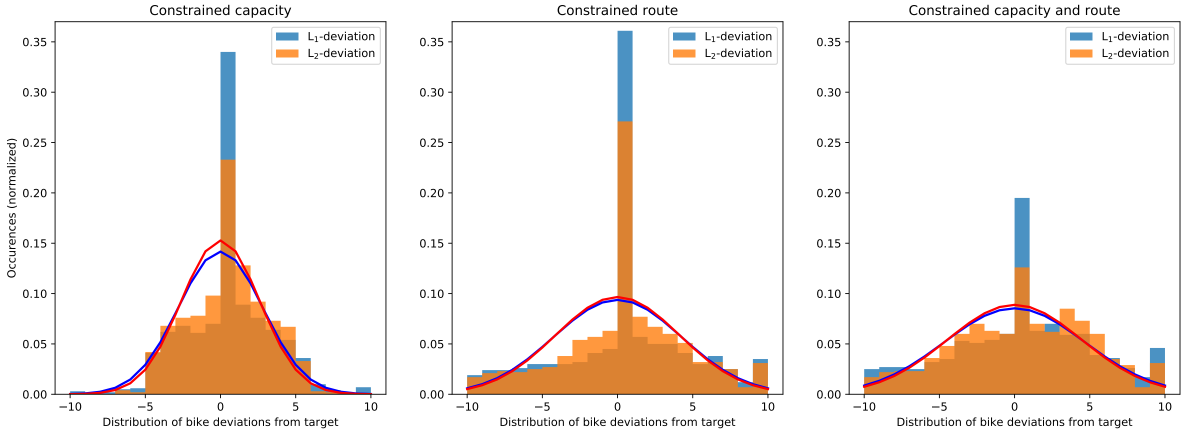


Figure 4.2 Normalized distributions of bike deviations from balance target, in various contexts.

the combined results of Tables 4.1, 4.3, and 4.6, we can confidently conclude that L_2-DEVIATION is a good approximator of the other measures, independent of context.

Takeaway

Due to the small size of the instances, the dispersion interval is the same for both L_1- and L_2-DEVIATION. However, for the 100 instances in the three contexts, 3% of the values are outliers for L_1-DEVIATION, compared to 2.2% for L_2-DEVIATION. For the BBSS problem, this directly translates to an increased number of functional stations. By extension, this means that the whole system offers a better service with L_2-DEVIATION than with L_1-DEVIATION. Smoothness is 3.57 and 3.61 for L_1- and L_2-DEVIATION, respectively. As for the two previous problems, this is unsurprising, as good solutions will not necessarily tend to be smooth.

Table 4.6 Comparison of measures of balance for the BBSS problem (capacity and route length are constrained).

	MINMAX	L_1-DEV.	L_2-DEV.	L_∞-DEV.	Average
MINMAX	0.00	14.71	23.97	9.30	12.00
L_1-DEVIATION	21.03	0.00	9.18	10.03	9.94
L_2-DEVIATION	6.15	1.22	0.00	2.59	2.52
L_∞-DEVIATION	15.56	17.78	26.23	0.00	14.89
Average	10.69	8.46	14.73	5.48	

Table 4.7 Characteristics of the solutions of the BBSS problem (capacity and route length are constrained).

	Dispersion	Smoothness	Outliers
MINMAX	1.00	4.05	4.20%
L_1-DEVIATION	1.00	3.57	4.20%
L_2-DEVIATION	1.00	3.61	3.30%
L_∞-DEVIATION	1.00	4.15	3.30%

4.3.4 Runtimes

The reader will recall that this paper is concerned with analyzing the balance properties of optimal solutions, rather than constructing efficient models to find them. Yet, one can wonder about the difficulty of reaching optimal solutions using the various balancing methods, and how this is affected by the size of the instances.

We observe a trend with the balancing methods where MINMAX and L_∞-DEVIATION are noticeably easier to solve than L_1- and L_2-DEVIATION. The difference between these two groups varies by problem: For the BACP, the first group is solved several hundred times faster than the second, while for the NPAP and the BBSS, the first group is solved around five times faster than the second.

For the BACP and the NPAP, the instance sizes found in our datasets match the largest sizes that one can reasonably expect to be faced with in a practical context. Both models can solve these instances in reasonable, if not negligible time. We have, however, severely limited the size of the instances for the BBSS problem in order to garner provably optimal solutions. This limited size of 10 stations per instance is close to the limit of what can be solved in reasonable time (around one minute). By increasing the size to 15 stations, the instances become unsolvable, even after an hour. We note that the simple CP models we use to solve these problems do not necessarily match what can be achieved by the state-of-the-art CP or IP solving methods.

4.4 Practical Considerations

Examination of the characteristics of balance, coupled with the conclusions of the case studies, indicate that no measure of balance is systematically better than the others. Nevertheless some general guidelines concerning the choice of a measure can be derived from the lessons learned in the previous sections.

Rules of Thumb. As a general rule, MINMAX has a large dispersion interval and is robust against outliers. L_1-, L_2-, and L_∞-DEVIATION tend to be more sensitive to outliers, but these outliers have a limited impact due to the shorter dispersion intervals. L_2-DEVIATION is particularly appealing for its good approximation of the other measures (which is not necessarily true the other way around), but it may suffer in performance in an IP model due to its nonlinearity and increased complexity.

Reconsidering Balance. When balance is initially introduced in a problem, it is often chosen to fit more the optimization method used to solve the problem than the problem requirements themselves. It also tends to become the *de facto* standard, and further research into the problem usually achieves balance using the same means, as *this is the way that balance is done for this problem*. The fact that reusing the same measure of balance makes it easier to compare with previous work also contributes to perpetuate this problem. Balance should nonetheless be reevaluated—the most popular way of achieving balance for a problem is not necessarily the best.

Linearity. If a problem is modeled with IP and already has nonlinear constraints or a nonlinear objective, L_2-DEVIATION can be used without introducing much more complexity. Otherwise, it may be better to compromise and use a linear measure of balance to remain in the linear realm. If using L_2-DEVIATION cannot be avoided, a CP model offers high flexibility.

Practical Constraints. While in theory a given measure of balance can be the best one, in practice there may be time and resource restrictions to consider. A non-optimal solution using an ideal measure of balance may be inferior to an optimal solution using a non-ideal measure of balance. The theoretical quality of a solution may not correlate with its quality in practice.

Neutral Characteristics. Most characteristics, such as the size of the dispersion interval, the number of outliers, and so on, are neutral—they are not inherently desirable nor detrimental. For instance, the distribution of well-balanced workloads would form a narrow bell curve around the mean, since we are interested in having each worker share a similar workload. However, diversity in the occurrences of values could also be desirable, for example to ensure a wide coverage in the test suites of some large software projects [95].

Hybridization. Multiple measures of balance can be combined to suit particular needs. For example, the most extreme values could be bounded with MINMAX, and the result further balanced with L_1-DEVIATION. With enough domain knowledge, such hybrids could present characteristics specifically tailored to a particular problem.

Other Considerations. Marsh and Schilling [14] consider other types of characteristics, some of which have to do with the human factor. For instance, they consider that a user should understand the measures of balance well enough to realize their implications, allowing them to make an informed choice among them. This highlights the fact that in a real-world setting, there is often more to balance than the mathematical facet.

Limitations. In the three cases studied, all concluded that L_2-DEVIATION was a good measure of balance for those problems due to its sensitivity to outliers. However, this specific characteristic is not necessarily desirable at all times. For instance, consider a factory with an equal number of workers and machines. Workers have various proficiencies on the machines, and a commodity is produced when all machines have been operated. Intuition may dictate that balancing the proficiencies of worker-machine pairs will ensure an efficient production. However, Fig. 4.3 shows that avoiding outliers may be counterproductive in some situations. The inverse of sensitivity to outliers, robustness against outliers, is itself a desirable characteristic at times as shown in this example.

$$\begin{array}{c|ccc|}
 & m_1 & m_2 & m_3 \\
w_1 & {\color{red}10} & 8 & {\color{blue}5} \\
w_2 & {\color{blue}1} & {\color{red}10} & 9 \\
w_3 & 7 & {\color{blue}3} & {\color{red}10}
\end{array}$$

Figure 4.3 Time required for each worker w_i to operate machine m_j. Perfect balance is achieved when all workers operate the machines on which they are the least proficient (red). In contrast, optimal throughtput is attained when the workloads are most unbalanced (blue).

4.5 Conclusion

Oftentimes balance is attached to a problem as a side constraint or as a secondary objective without much thought. This paper shows that balancing a solution is not as straightforward as it seems, and highlights a few properties for some types of measures of balance. Three problematic modeling choices have been shown and studied, after which general guidelines have been proposed to prevent modelers from succumbing to pitfalls when selecting a measure of balance.

CHAPTER 5 ARTICLE 3: FAIRNESS OVER TIME IN DYNAMIC RESOURCE ALLOCATION WITH AN APPLICATION IN HEALTHCARE

Andrea Lodi, Philippe Olivier, Gilles Pesant, and Sriram Sankaranarayanan

Submitted to *Mathematical Programming* (January 2021)

Abstract. Decision making problems are typically concerned with maximizing efficiency. In contrast, we address problems where there are multiple stakeholders and a centralized decision maker who is obliged to decide in a fair manner. Different decisions give different utility to each stakeholder. In cases where these decisions are made repeatedly, we provide efficient mathematical programming formulations to identify both the maximum fairness possible and the decisions that improve fairness over time, for reasonable metrics of fairness. We apply this framework to the problem of ambulance allocation, where decisions in consecutive rounds are constrained. With this additional complexity, we prove structural results on identifying fair feasible allocation policies and provide a hybrid algorithm with column generation and constraint programming-based solution techniques for this class of problems. Computational experiments show that our method can solve these problems orders of magnitude faster than a naive approach.

5.1 Introduction

A resource allocation problem can be defined as the problem of choosing an allocation from a set of feasible allocations to a finite set of stakeholders to minimize some objective function.

Generally, this objective function represents the efficiency of an allocation by considering, for example, the total cost incurred by all the stakeholders [96].

However, in some cases, such an objective may be inappropriate due to ethical or moral considerations. For instance, to whom should limited medical supplies go in a time of crisis [97]? These situations call for a different paradigm of decision-making that addresses the need of *fairness* in resource allocation.

Resource allocation problems trace their roots to the 1950s when the division of effort between two tasks was studied [98]. Fairness in resource allocation problems has been studied since the 1960s. A mathematical model for the fair apportionment of the U.S. House of Representatives considered the minimization of the difference between the disparity of representation between any pairwise states [99]. This problem was revisited about two decades later [100, 101]. The so-called *minimax* and *maximin* objectives in resource allocation, which achieve some fairness by either minimizing the maximum (or maximizing the minimum) number of resources allocated to entities, have been studied at least since the 1970s [102, 103]. A recent book on equitable resource allocation presents various models and advocates a lexicographic minimax/maximin approach for fair and efficient solutions [104]. This work further discusses multiperiod equitable resource allocation. Fair resource allocation over time is also considered in the dynamic case where resource availability changes over time and is solved using approximation algorithms [105]. A combination of efficiency and fairness in resource allocation is common in communication networks [106, 107], and can be found in various other fields, such as in some pickup and delivery problems [108].

Our Contributions. Our first contribution is to formally set up an abstract framework for repeatedly solving a fair allocation problem such that enhanced fairness is achieved over time, as opposed to a single round of allocation. Typically, a fair allocation could turn out to be an inefficient allocation. The framework addresses this trade-off by explicitly providing adequate control to the decision-maker on the level of the desired efficiency of the solutions.

Then, we prove that the concept of achieving fairness over time is not useful should the set of feasible allocations be a convex set, as long as the value each stakeholder obtained is a linear function of the allocation. We prove this result irrespective of the measure of fairness one uses, as long as the measure has crucial technical properties mentioned formally later. Next, we show closed-form and tight solutions for the number of rounds required for perfectly fair solutions, for special choices of the set of allocations that could be of practical interest.

However, there might also be other cases of practical interest where such closed-form solutions are harder, if not impossible, to obtain. We provide an integer programming formulation to solve these problems. The formulation hence provided can be combined directly with delayed column generation techniques in case of large instances.

With the above results and ideas, we consider the problem of allocating ambulances to multiple areas of a city, where the residents of each area are the stakeholders. This family of problems is easier than the general case as there exists an efficient greedy algorithm that provides reasonable solutions fast. However, such solutions could be far from optimal, and we provide examples where the greedy algorithm fails. Besides that, in this family of problems, we also impose restrictions on how allocations in successive rounds could be. This is reminiscent of the real-life restriction that one does not want to relocate too many ambulance vehicles each day. This added constraint makes identifying good solutions much harder as the proposed column generation-technique becomes invalid now. To solve this problem, we identify a graph induced by any candidate solution and show that deciding the feasibility of a candidate solution provided by column generation is equivalent to identifying Hamiltonian paths in this graph. Hence, we provide multiple subroutines to retain the validity of the column generation-based approach. We also provide experimental results on the computational efficiency of our method.

The paper is organized as follows. Section 5.2 introduces some basic definitions and concepts. Section 5.3 presents some theoretical proofs for simple cases, and Section 5.4 establishes a general framework which allows to tackle more complicated cases. This framework is used to solve a practical problem in Section 5.5, whose results are presented in Section 5.6. Finally, some conclusions are drawn in Section 5.7.

5.2 Preliminaries

In this section, we formally define the terms used throughout the manuscript. We study resource allocation in the context of fairness over time, meaning that resources are allocated to stakeholders over some time horizon. We use $\mathcal{X}$ to denote the set of all feasible allocations, with $n \in \mathbb{Z}_{\geq 0}$, $n \geq 2$ being the number of stakeholders among whom the allocations should be done in a fair way.

Definition 3 (Benefit function) *The benefit function $\tau : \mathcal{X} \to \mathbb{R}^n$ is a function such that the i-th component $[\tau(x)]_i$ refers to the benefit or utility enjoyed by the i-th stakeholder due to the allocation $x \in \mathcal{X}$.*

Assume ϕ to be a function that measures the unfairness associated with a given benefit pattern for the n stakeholders—a concept formalized in Definition 4—which is to be minimized. In this context, a basic fair resource allocation problem can be defined simply as

$$\min_{x \in \mathcal{X}} \qquad \phi(\tau(x)). \tag{5.1}$$

Many decision-makers are public servants who are either elected or nominated, and who serve in their position for a predetermined amount of time. In the context of serving the public, these officials must often juggle with limited means to ensure that everyone they serve is catered for. Public satisfaction naturally plays a role in the evaluation of their performance. Herein lies the motivation to solve resource allocation problems that consider fairness over time.

Assuming that there are T rounds of decision making, model (5.1) can be expanded to

$$\min_{x(t),y} \quad \phi(y) \tag{5.2a}$$

$$\text{s.t.} \quad y = \frac{1}{T}\sum_{t=1}^{T} \tau(x(t)) \tag{5.2b}$$

$$x(t) \in \mathcal{X}, \qquad \forall\, t = 1, \dots, T. \tag{5.2c}$$

Here, we implicitly assume that it is sensible to add the benefits obtained at different rounds of decision making, and use the average benefit over time to compute fairness.

Now, in this section, we first define the properties that a general measure of unfairness should have for it to be considered valid in our context. To start with, we state that (i) if all the stakeholders of the problem get the same benefit or utility, then the unfairness associated with such an allocation of benefits should be 0, and (ii) the unfairness associated with benefits should be invariant to permutations of the ordering of stakeholders.

Definition 4 (Unfairness function) *Given $y \in \mathbb{R}^n$, $\phi : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ determines the extent of unfairness in the allocations, if the i-th stakeholder gets a benefit of y_i. Such a function ϕ satisfies the following:*

1. $\phi(y_1, \dots, y_n) = 0 \iff y_1 = y_2 = \dots = y_n$,
2. $\phi(y_1, \dots, y_n) = \phi(\pi_1(y), \dots, \pi_n(y))$ *for any permutation π.*

In addition, if ϕ is a convex function, we call ϕ a convex unfairness function.

In general, trivial solutions where no benefit is obtained by any of the stakeholders, i.e., when $\tau(x)$ is a zero vector, are perfectly fair solutions! However, this results in a gross loss of efficiency.

Definition 5 (Inefficiency function) *Given $\mathcal{X}$ and the benefit function τ, the inefficiency func-*

tion $\eta : \mathcal{X} \to [0,1]$ is defined as

$$\eta(x) \quad = \quad \begin{cases} 0 & \text{if } \overline{f} = \underline{f}, \\ \frac{\overline{f} - \sum_{i=1}^{n}[\tau(x)]_i}{\overline{f} - \underline{f}} & \text{otherwise,} \end{cases} \tag{5.3}$$

where $\overline{f} = \sup_{x \in \mathcal{X}} \sum_{i=1}^{n} [\tau(x)]_i$, $\underline{f} = \inf_{x \in \mathcal{X}} \sum_{i=1}^{n} [\tau(x)]_i$, and $\overline{f}$ and $\underline{f}$ are assumed to be finite.

Remark 6 *Note that for all feasible $x \in \mathcal{X}$, we indeed have $\eta(x) \in [0,1]$. For the most efficient x, i.e., the x (or allocation) that maximizes the sum of benefits, $\eta(x) = 0$, while for the least efficient x, we have $\eta(x) = 1$. Thus, η serves as a method of normalization of the objective values.*

Definition 7 *Equivalence classes defined due to the function η and partitioning $\mathcal{X}$ are, for any $\overline{\eta} \in [0,1]$*

$$\widetilde{\mathcal{X}}_{\overline{\eta}} \quad := \quad \{x \in \mathcal{X} : \eta(x) = \overline{\eta}\}. \tag{5.4}$$

The above definition enables to succinctly represent a target to find fair solutions *only* if they are also sufficiently efficient. For instance, one might restrict the feasible set to $\cup_{\eta : \eta < \overline{\eta}} \widetilde{\mathcal{X}}_{\eta}$.

We now define a single-period fair allocation problem, subject to efficiency constraints. One might always choose $\eta = 1$ to retrieve the problem in model (5.1) without efficiency constraints.

Definition 8 (Single-period fair allocation (SPFA) problem) *Given $\overline{\eta} \in [0,1]$, the single-period fair allocation problem is to solve*

$$\min_{x} \quad \phi(\tau(x)) \tag{5.5a}$$

$$s.t. \quad x \in \bigcup_{\eta : \eta \in [0, \overline{\eta}]} \widetilde{\mathcal{X}}_{\eta}. \tag{5.5b}$$

Example 9 motivates and provides a mean of validation for model (5.5). It shows that with reasonable choices, we retrieve the intuitively fair solution.

Example 9 *Consider the case where $\mathcal{X} = \{x \in \mathbb{R}_+^n : \sum_{i=1}^{n} x_i = 1\}$, i.e., $\mathcal{X}$ is a simplex. Further, assume that $[\tau(x)]_i = \tau_i x_i$ for some scalars $\tau_i \gneq 0$, and that, w.l.o.g., $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n$.*

For the choice of $\overline{\eta} = 1$, i.e., with no efficiency constraints, the solution for the SPFA problem is given by

$$x_i^\star \quad = \quad \frac{g}{\tau_i}, \quad \forall i \tag{5.6a}$$

$$\textit{where} \quad g \quad = \quad \frac{1}{\sum_{i=1}^{n} \frac{1}{\tau_i}}. \tag{5.6b}$$

Clearly, each stakeholder enjoys a benefit of g, and hence the unfairness associated with this allocation is 0. We can notice that $\eta(x^\star) = \frac{\tau_1 - ng}{\tau_1 - \tau_n}$.

Note that for the case where $n = 2$, the above simplifies to

$$\begin{aligned} x_1^\star &= \frac{\tau_2}{\tau_1 + \tau_2} \\ x_2^\star &= \frac{\tau_1}{\tau_1 + \tau_2} \\ \eta(x^\star) &= \frac{\tau_1^2 - \tau_1\tau_2}{\tau_1^2 - \tau_2^2}. \end{aligned}$$

We now define the fairness-over-time problem.

Definition 10 (T-period fair allocation (T-PFA) problem) *Given $\overline{\eta} \in [0,1]$ and $T \in \mathbb{Z}_{\geq 0}$ with $T \geq 2$, the T-period fair allocation problem is to solve*

$$\min_{x(t),y} \quad \phi(y) \tag{5.7a}$$

$$\text{s.t.} \quad y = \frac{1}{T}\sum_{t=1}^{T} \tau(x(t)) \tag{5.7b}$$

$$x(t) \in \bigcup_{\eta:\eta\in[0,\overline{\eta}]} \widetilde{\mathcal{X}}_\eta, \quad \forall\, t = 1, \ldots, T. \tag{5.7c}$$

We say that a fair-allocation problem (SPFA or T-PFA) has *perfect fairness* if the optimal objective value of the corresponding optimization problems is 0.

Example 11 (Usefulness of the T-PFA) *Let $\mathcal{X} = \{x \in \{0,1\}^2 : x_1 + x_2 = 1\}$. Further, let $\tau(x) = (2x_1, x_2)$. Let $\overline{\eta} = 1$. Note that, in the case of the SPFA, the optimal objective is necessarily nonzero, since the benefits of the two stakeholders are unequal for every feasible solution. However, consider the 3-PFA. Now, if $x(1) = (1,0)$, $x(2) = (0,1)$, $x(3) = (0,1)$, $y = \left(\frac{2}{3}, \frac{2}{3}\right)$. So, for any choice of ϕ, the optimal objective value is 0, which is strictly better than the SPFA solution.*

5.3 Simple Cases

5.3.1 Convex $\mathcal{X}$

We have motivated in Example 11 that ensuring fairness over multiple rounds could offer better results than seeking fairness on a per-round basis. We now show that this holds only for a nonconvex $\mathcal{X}$. In other words, if $\mathcal{X}$ is convex, we necessarily get no improvement in T-PFA over SPFA.

Theorem 12 *Let $\mathcal{X}$ be convex. Further, let τ be a linear function. Let $\overline{\eta}$ be given. Let $f^\star$ be the optimal value of the SPFA problem and let $f_2^\star, f_3^\star, \dots$ be the optimal values of the T-PFA problem with $T = 2, 3, \dots$. Then $f^\star = f_2^\star = f_3^\star \dots$.*

Proof 13 *Given that τ is a linear function, we can write $[\tau(x)]_i = \tau_i^\mathsf{T} x$ for an appropriately chosen τ_i, for $i = 1, \dots, n$. Suppose that $x^\star(t)$, for $t = 1, \dots, T$, and $y^\star$ solve the T-PFA problem. By our notation, this has an objective value of $\phi(y^\star) = f_T^\star$. Now, we construct a solution for the SPFA problem with an objective value equal to $f_T^\star$. For this, consider the point $\overline{x} = \frac{1}{T}\sum_{t=1}^{T} x^\star(t)$. First we claim that $\overline{x} \in \mathcal{X}$.*

We have that $x(t) \in \widehat{\mathcal{X}}$ where $\widehat{\mathcal{X}} = \cup_{\eta:\eta\in[0,\overline{\eta}]}\widetilde{\mathcal{X}}_\eta$. Now, we observe that $\widehat{\mathcal{X}}$ is a convex set given by $\{x \in \mathcal{X} : (\sum_{i=1}^{n} \tau_i)^\mathsf{T} x \geq \overline{f} - (\overline{f} - \underline{f})\overline{\eta}\}$, where $\overline{f}$ and $\underline{f}$ are constants. Since $\overline{x}$ is obtained as a convex combination of $x^\star(t) \in \widehat{\mathcal{X}}$, it follows that $\overline{x} \in \widehat{\mathcal{X}}$. Finally, from the linearity of τ, we can set $\overline{y} = y^\star$. This shows that $(\overline{x}, \overline{y})$ is feasible. Since $\overline{y} = y^\star$, it follows that their objective function values are equal. ■

Having proven that multiple rounds of allocation cannot improve the fairness when $\mathcal{X}$ is convex, we show that it is not necessarily due to the fact that perfect fairness is obtainable in SPFA. Example 14 shows an instance where the unfairness is strictly positive with a single round of allocation, irrespective of the choice of ϕ. Naturally, due to Theorem 12, perfect fairness is neither possible with multiple rounds of allocation.

Example 14 *Consider a fair allocation problem where $\mathcal{X} = \{x \in \mathbb{R}^3_{\geq 0} : x_1 + x_2 + x_3 = 1\}$. Clearly $\mathcal{X}$ is convex. Consider linear τ defined as*

$$
\begin{aligned}
[\tau(x)]_1 &= x_1 + \frac{3}{4}x_2 + \frac{3}{4}x_3 \\
[\tau(x)]_2 &= x_2 \\
[\tau(x)]_3 &= x_3.
\end{aligned}
$$

Let $\overline{x}$ be a fair allocation. In such a case, we need $[\tau(\overline{x})]_1 = [\tau(\overline{x})]_2 = [\tau(\overline{x})]_3$. Thus, we need

$$\begin{aligned} \rho &= x_1 + \frac{3}{4}x_2 + \frac{3}{4}x_3 \\ \rho &= x_2 \\ \rho &= x_3 \end{aligned}$$

for some $\rho \in \mathbb{R}_+$. Solving this linear system gives the unique *fair allocation as allocations of the form $(x_1, x_2, x_3) = (-0.5\rho, \rho, \rho)$, which necessarily violates the non-negativity constraints in the definition of $\mathcal{X}$. Any other allocation necessarily has $\phi(\tau(x)) > 0$.*

5.3.2 Simplicial $\mathcal{X}$

The first part of Theorem 15 states that for any n and a $\mathcal{X}$ of the specified form, perfect fairness is possible precisely after $\overline{T}$ periods, provided the efficiency requirements are as stated. The second part shows tightness of the results saying, for any other stricter efficiency requirements, there exists a counterexample with just two stakeholders, such that perfect fairness is impossible in $\overline{T}$ steps.

Theorem 15 *Let $\mathcal{X} = \{x \in \mathbb{Z}^n_{\geq 0} : \sum_{i=1}^n x_i \leq a\}$ for some positive integer a. Let the benefit function be $[\tau(x)]_i = \tau_i x_i$, where each τ_i is a positive integer. Assume, w.l.o.g., that $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n > 0$. Let $L = LCM(\tau_1, \ldots, \tau_n)$. Then,*

1. *Perfect fairness cannot be obtained in T periods where T is strictly less than $\overline{T} = \left\lceil \frac{1}{a} \sum_{i=1}^n \frac{L}{\tau_i} \right\rceil$. However, if $1 > \overline{\eta} \geq \frac{\tau_1 - \min(\tau_n, \tau_1/a)}{\tau_1}$, and if $\overline{T} > 1$, perfect fairness can be achieved in exactly $\overline{T}$ periods.*

2. *Given any $n \geq 2$, there exist $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n$ and $a \in \mathbb{Z}_{>0}$ such that perfect fairness is unattainable for T-PFA where $T \leq \overline{T}$ as above with $\overline{\eta} < \frac{\tau_1 - \min(\tau_n, \tau_1/a)}{\tau_1}$.*

Proof 16 *For the first part, observe that an unfairness of* 0 *can be achieved if and only if the total benefit over time for each stakeholder should add up to the same value. Since x_i and τ_i are all integers and we have integer number of periods, the smallest value that the benefits can all add up to is* 0*, and the next smallest value is L. But the strict inequality $\overline{\eta} < 1$ excludes the trivial allocation of $(0, 0, \ldots, 0)$. A total benefit of L can be achieved for stakeholder i if and only if $\sum_{t=1}^T x_i(t) = L/\tau_i$. Summing this over all stakeholders, we get*

$$\sum_{i=1}^n \frac{L}{\tau_i} = \sum_{i=1}^n \sum_{t=1}^T x_i(t)$$

$$
\begin{aligned}
& & = & \sum_{t=1}^{T}\sum_{i=1}^{n} x_i(t) \\
& & \leq & \sum_{t=1}^{T} a \\
& & = & aT \\
\implies & \frac{1}{a}\sum_{i=1}^{n}\frac{L}{\tau_i} & \leq & T
\end{aligned}
$$

and the ceiling follows from the fact that T is an integer. Now that we have shown that we need at least $\overline{T}$-PFA, we show that the unfairness can be made to 0 *in exactly $\overline{T}$ periods. For this, consider a reverse-lexicographic allocation method where all allocations are made to stakeholder n until the total allocation sums to L/τ_n. Next, allocate similarly for $n-1$, $n-2$ and so on up to stakeholder* 1*. Observe that each of these allocations at any time t is in $\mathcal{X}$ by construction. To show that the allocation in each step has an inefficiency value at most $\overline{\eta}$, consider the two most inefficient allocations. This could happen in the first period, when all allocations are to the stakeholder n, i.e., $x(1) = (0, 0, \dots, 0, a)$. The inefficiency function value for this allocation is $\frac{a\tau_1 - a\tau_n}{a\tau_1 - 0} = \frac{\tau_1 - \tau_n}{\tau_1} \leq \overline{\eta}$. Alternatively, the most inefficient allocation could be in the last period, where a single allocation is made to the first stakeholder, i.e., $x(T) = (1, 0, 0, \dots, 0)$. The inefficiency function value for this allocation is $\frac{a\tau_1 - \tau_1}{a\tau_1 - 0} = \frac{\tau_1 - \tau_1/a}{\tau_1} \leq \overline{\eta}$. This proves the first part.*

We prove the second part for $n = 2$ first, and then generalize it to $n \geq 3$. Let the target $\overline{T} = \widehat{T} \geq 2$. Choose $\tau_1 = (\widehat{T}-1)a$ and $\tau_2 = 1$. The LCM $L = (\widehat{T}-1)a$. From the first part, perfect fairness is not obtainable for $T < \frac{1}{a}\left(\frac{(\widehat{T}-1)a}{(\widehat{T}-1)a} + \frac{(\widehat{T}-1)a}{1}\right) = (\widehat{T}-1) + \frac{1}{a}$ periods. But from the integrality of the number of periods, perfect fairness is not possible using up to $\widehat{T}-1$ periods. We now show that perfect fairness is not possible with $\widehat{T}$ periods either. Note that the set of solutions where perfect fairness is achieved in $\widehat{T}$ time steps are all in the form

$$
\begin{aligned}
x(1) &= (0, a - \delta_1) \\
x(2) &= (0, a - \delta_2) \\
&\vdots \\
x(\widehat{T}-1) &= (0, a - \delta_{\widehat{T}-1}) \\
x(\widehat{T}) &= \left(\sum_{i=1}^{\widehat{T}-1} \delta_i, 1\right)
\end{aligned}
$$

and its permutations over time for some integers $\delta_i \in \{1, 2, \dots, a\}$ such that $0 \leq \sum_{i=1}^{\widehat{T}-1} \delta_i \leq a-1$. Now consider the inefficiency function of the allocation $x(1)$; $\eta(x(1)) = \frac{(\widehat{T}-1)a - 1 + \delta_1/a}{\widehat{T}-1} > \overline{\eta}$

shows that it is infeasible. Thus, any perfectly fair allocation for $\overline{T}$-PFA problem is an infeasible allocation, providing the necessary counterexample. To extend this for $n > 2$, choose $\tau_2 = \tau_3 = \ldots = \tau_{n-1} = 2$, and analogous arguments provide the counterexample. ■

Example 17 *Consider Theorem 15 applied to Example 11. We have $\mathcal{X} = \{(1,0),(0,1)\}$, and thus $a = 1$. Then, $\tau(x) = (2x_1, x_2)$, so $\tau_1 = 2$ and $\tau_2 = 1$, and $\tau_1 \geq \tau_2 \geq 0$ holds. In addition, $L = LCM(2,1) = 2$. Then, $\overline{T} = \left\lceil \frac{1}{a}\sum_{i=1}^{n} \frac{L}{\tau_i} \right\rceil = \left\lceil \frac{1}{1}(\frac{2}{2}+\frac{2}{1}) \right\rceil = 3$.*

5.3.3 Sparse Simplicial $\mathcal{X}$

The sparse simplicial form for $\mathcal{X}$ is another interesting case from a practitioner's perspective. For example, a government might want to allot money to n possible projects, but if it divides among all of them, then it might be insufficient for any of them. So, there could be a restriction that the government allots it to at most r projects.

Theorem 18 *Let $\mathcal{X} = \{x \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^{n} x_i \leq a, \|x\|_0 \leq r\}$ where $\|\cdot\|_0$ is the sparsity pseudo-norm, which counts the number of non-zero entries in its argument. Assume that the benefit function is $[\tau(x)]_i = \tau_i x_i$, where each τ_i is a positive integer. Assume, w.l.o.g., that $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n > 0$. We denote by (p,q) the quotient and the remainder for $\frac{n}{r}$. Let $1 > \overline{\eta} \geq \frac{a\tau_1 - qv}{a\tau_1}$ where $v = a\min\left\{\sum_{i=1}^{q}\frac{1}{\tau_i}, \sum_{i=0}^{r-1}\frac{1}{\tau_{n-i}}\right\}$. Then, $\overline{T} = \lceil \frac{n}{r} \rceil$ is the smallest number of periods required to achieve perfect fairness.*

Proof 19 *Consider the following allocation: In period t for $1 \leq t \leq \overline{T} - 1$, allocate $\frac{v}{\tau_i}$ for $i = n-(t-1)r-1, n-(t-1)r-2, \ldots, n-tr$ and 0 to the rest. Note that in each period, we allocate exactly to r stakeholders, and that the inequality constraint is satisfied due to the definition of v and the ordering of $\tau_1 \geq \tau_2 \ldots \geq \tau_n$. We can ensure that in the first $\overline{T} - 1$ periods, each of the allocated player receives a value of v, and hence the total benefit of the allocation is rv, giving the $\eta(x(t)) = \frac{a\tau_1 - rv}{a\tau_1} \leq \overline{\eta}$. In the last period, we allocate $\frac{v}{\tau_i}$ for $i = 1, 2, \ldots, r$. Feasibility follows as before and $\eta(x(\overline{T})) = \frac{a\tau_1 - qv}{a\tau_1} \leq \overline{\eta}$, which is feasible. Perfect fairness follows since the benefit to each player is invariant.*

To prove that the given $\overline{T}$ is the smallest number of periods required to obtain perfect fairness, observe that, for any fewer number of periods, it is impossible for all players to get a non-zero allocation. ■

Remark 20 *Note that, unlike earlier, we do not have tightness in η now.*

Remark 21 *We note that the above results, Theorems 12, 15 and 18, are agnostic to the choice of ϕ, and only use the fact that $\phi(x_1, \ldots .x_m) = 0 \iff x_1 = \ldots = x_n$. Any result that holds for $\phi(\cdot) \neq 0$ has to necessarily depend upon the choice of ϕ.*

Example 22 *Consider Theorem 18 applied to Example 11. We have $a = 1$ and $r = 2$. Again, $\tau(x) = (2x_1, x_2)$, so $\tau_1 = 2$ and $\tau_2 = 1$, and $\tau_1 \geq \tau_2 \geq 0$ holds. $\overline{T} = \lceil \frac{n}{r} \rceil = \lceil \frac{2}{2} \rceil = 1$. Indeed, the solution $(\frac{1}{3}, \frac{2}{3})$ is feasible and fair.*

5.4 General Combinatorial $\mathcal{X}$

In many practical areas of interest, $\mathcal{X}$ could be more complicated than the sets presented in Section 5.3. Let $\mathcal{X} = \{x^1, x^2, \ldots, x^k\}$. We assume that $\mathcal{X}$ is finite, but typically has a large number of elements, given in the form of a solution to a combinatorial problem. We consider a benefit function where $\tau(x) = \Gamma x$, for some matrix Γ. We thus have $[\tau(x)]_i = \tau_i^\mathsf{T} x$ for $i = 1, \ldots, n$.[1] The efficiency constraint here is trivial, as the inefficient allocations x^j can be removed from $\mathcal{X}$ to retain another (smaller) finite $\mathcal{X}$. In fact, with linear τ, the efficiency constraint is a linear inequality.

Let q_j be the *number of times* the allocation $x^j \in \mathcal{X}$ is chosen over T rounds of decision. In this setting, the T-PFA problem can be restated as[2]

$$\begin{aligned}
\min_{q,y} \quad & \phi(y) && && \text{(5.8a)}\\
\text{s.t.} \quad & y_i = \tau_i^\mathsf{T} \left(\sum_{j=1}^{k} q_j x^j \right), && \forall\, i = 1, \ldots, n && \text{(5.8b)}\\
& \sum_{j=1}^{k} q_j = T && && \text{(5.8c)}\\
& q_j \in \mathbb{Z}_{\geq 0}, && \forall\, j = 1, \ldots, k. && \text{(5.8d)}
\end{aligned}$$

Note that since the theorems of Section 5.3 do not extend to this case, and since Example 14 shows that a perfectly fair solution may not even exist, we know neither the value of the fairest feasible solution, nor the smallest value of T that guarantees such a solution. We now present a two-phase integer program that finds the smallest value of T that guarantees the fairest feasible solution. In the first phase, we are interested in finding the fairest feasible solution. This is achieved by solving model (5.8), replacing (5.8c) with $\sum_{j=1}^{k} q_j \geq 1$ to ensure that not only the fairest solution is returned, but that this solution not be $(0, 0, \ldots, 0)$. Let $\phi^\star$ be the value of the solution found in the first phase. In the second

[1] Matrix Γ can implicitly be seen as an adjacency matrix in order to consider a graph representation of the problem.

[2] The reader may have observed that an approach based on column generation would lend itself well for such a model. This is discussed in Section 5.5.

phase, we are interested in finding the smallest T which can accommodate a solution of value $\phi^\star$. This is achieved by solving model (5.8), again replacing (5.8c) with $\sum_{j=1}^{k} q_j \geq 1$, adding $\phi(y) = \phi^\star$, and changing the objective to $\min_{q,y} \sum_{j=1}^{k} q_j$.

We should note that the value of $T^\star$ found by the second phase could be quite high, even for simple choices of $\mathcal{X}$. If perfect fairness can be attained in a time horizon of size, say, 10000, and that a discrete time point represents a day, then this period of 27-odd years would probably not be suitable for most practical applications, since it is only by reaching the end of the time horizon that optimal fairness is guaranteed. In practice, then, large time horizons can be problematic. This motivates the need for consistent fairness on a smaller scale.

Some compromises can be made which would mitigate this issue. Manually fixing T to a value not higher than the expected duration of the problem would ensure a fair distribution of resources, since reaching the end of the time horizon would be assured. Accepting some degree of unfairness would also decrease the length of the time horizon and achieve similar results. Example 23 illustrates these two options.

Example 23 *Consider a fair allocation problem where* $\mathcal{X} = \{x \in \mathbb{Z}^2_{\geq 0} : x_1 + x_2 = 1\}$. *Further consider* τ *defined as*

$$\begin{aligned} [\tau(x)]_1 &= x_1 + \frac{15}{37} x_2 \\ [\tau(x)]_2 &= x_2 + \frac{15}{47} x_1. \end{aligned}$$

Perfect fairness can be achieved when $T = 1109$, *but if we fix* $T = 5$, *the difference between* y_1 *and* y_2 *will be a mere* ~ 0.42, *and if we allow a small difference of* 0.01 *between* y_1 *and* y_2, *this could be achieved when* $T = 15$.

Another compromise would be to choose a fair way of reaching the solution. Given that we know the value of $T^\star$ which grants optimal fairness, we could then identify the *best path* leading to the optimal solution. The best path is the one which ensures that unfairness is kept as low as possible in the intermediate time points, without sacrificing optimality at the end of the time horizon. This would, however, require solving another linear program, which may prove burdensome if the time horizon were too large.

One may think that greedily aiming for the fairest solution at each time point—while considering any unfairness incurred in previous time points, and enforcing minimal efficiency constraints by ensuring that all available resources are used at each time point—could lead to optimality at the end of the time horizon. Theorem 24 shows that it is not the case.

Theorem 24 *A greedy approach does not provide any guarantee of optimality for the T-PFA problem.*

Proof 25 *Consider a fair allocation problem where* $\mathcal{X} = \{x \in \mathbb{Z}^3_{\geq 0} : x_1 + x_2 + x_3 = 1\}$*. Further consider* τ *defined as*

$$\begin{aligned}
[\tau(x)]_1 &= \epsilon x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 \\
[\tau(x)]_2 &= x_2 \\
[\tau(x)]_3 &= x_3.
\end{aligned}$$

where $0 < \epsilon < \frac{1}{2}$*. If* $T = 2$*, it is obvious that the only perfectly fair solution (and, as it happens, the most efficient too) is achieved by allocating the resource in turn to* x_2 *and* x_3*, over T. Yet, for any reasonable measure of unfairness, the greedy choice would be to allocate the resource twice to* x_1*. Thus, a greedy approach does not necessarily lead to an optimal solution.* ■

We should note that there is a trade-off between fairness and efficiency: As illustrated in Example 26, enforcing constraints to increase fairness will generally decrease efficiency, and vice-versa.

Example 26 *Consider a fair allocation problem where* $\mathcal{X} = \{x \in \mathbb{Z}^3_{\geq 0} : 1 \leq x_1 + x_2 + x_3 \leq 3\}$*. Further consider* τ *defined as*

$$\begin{aligned}
[\tau(x)]_1 &= x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 \\
[\tau(x)]_2 &= x_2 + \frac{1}{2}x_1 \\
[\tau(x)]_3 &= x_3 + \frac{1}{2}x_1.
\end{aligned}$$

By enforcing maximum efficiency, the only feasible solution is $x = (3, 0, 0)$ *for* $y = (3, \frac{3}{2}, \frac{3}{2})$ *and an efficiency of 6, which is not very fair. In contrast, by enforcing maximum fairness, the only feasible solution is* $x = (0, 1, 1)$ *for* $y = (1, 1, 1)$ *and an efficiency of 3, which is not very efficient (one resource is even wasted due to the fairness constraints).*

We should also note that the value of T may have a noticeable impact on both fairness and efficiency, as illustrated in Example 27.

Example 27 *Consider a fair allocation problem where* $\mathcal{X} = \{x \in \mathbb{Z}^4_{\geq 0} : 1 \leq x_1 + x_2 + x_3 + x_4 \leq 3\}$. *Further consider* τ *defined as*

$$\begin{aligned}[\tau(x)]_1 &= x_1 + x_2 \\ [\tau(x)]_2 &= x_2 + x_1 \\ [\tau(x)]_3 &= x_3 + x_4 \\ [\tau(x)]_4 &= x_4 + x_3.\end{aligned}$$

The smallest T *accommodating perfect fairness is* $T = 1$ *with* $x = (1,0,1,0)$[3] *for* $y = (1,1,1,1)$ *and an efficiency of 4 (with one wasted resource). However,* $T = 2$ *also accommodates solutions with perfect fairness, albeit with an increased efficiency. Take, for instance,* $x = (3,0,0,0)$ *and* $x = (0,0,0,3)$ *over two time points, for a combined* $y = (3,3,3,3)$ *and an efficiency of 12. This represents a noticeable increase in efficiency over choosing the initial solution twice to cover the same time horizon.*

In other words, reaching perfect fairness in the shortest possible horizon might not lead to the most efficient solutions.

5.5 Ambulance Location and Relocation

The task of allocating ambulances to a set of bases in a region is a well-known problem in operations research [109]. The objective of this problem is generally one of efficiency: Ambulances should be allocated to prime spots such that they can quickly reach the maximum number of people.

The definition of this problem is not unified across the literature [109, 110].[4] Most definitions, however, agree that the population should receive an efficient service, which is generally translated into some form of coverage. In this paper, we are not solely concerned by efficiency, but we are further interested in fairness: Ambulances should be allocated such that the same set of people is not always at a disadvantage with respect to access to a quick service.

[3] Or any other equivalent solution.

[4] This is due to the fact that research projects often work with datasets associated with specific regions, each of which must follow local guidelines and regulations.

5.5.1 Preliminary Problem Formulation

In this manuscript, we adopt the following definition. The region to which ambulances are allocated is divided into n demand zones, the travel time between each pair being a known quantity. Ambulances can only be allocated to bases, which are located within a subset of the zones. Each zone (equivalently, the people living in each zone) are individual stakeholders in this model. A pre-chosen T rounds of decision is modeled to occur, over which the ambulances should be allocated in a fair and efficient manner. In each round, a configuration x of how m ambulances are placed is chosen. Next, we define the benefit function for the zones. In this model, each zone i, based on its population, has a demand of ζ_i ambulance vehicles. A zone i is said to be *covered* in configuration x if there are at least ζ_i ambulances located in bases from which zone i could be reached in a time less than a chosen time limit called the *response threshold time*. Now, $[\tau(x)]_i = 1$ if zone i is covered by the configuration x and $[\tau(x)]_i = 0$ otherwise. We note that we use a nonlinear benefit function in this case, as opposed to the simpler cases in Section 5.3 to both reflect the fact that more number of ambulances are essential to sufficiently serve certain regions and also to demonstrate the robustness of our methods to even certain classes of nonlinear (piecewise linear) benefit functions. The nonlinearity, as shown below, can be modeled using auxiliary variables and linear functions, to conform to the form of (5.8). Furthermore, efficiency constraints analogous to (5.7c) are enforced by stating that at least a fraction f of the zones should be covered in each allocation. The unfairness metric ϕ used is the difference between the two zones which are most often and least often covered, over T.

With the above, the problem is a standard T-PFA problem in Definition 10. However, the ambulance allocation problem involves additional constraints on allocations between two consecutive periods. Decision-makers prefer policies that ensure that not too many ambulances have to shift bases on a daily basis. Thus, between two consecutive allocations, we would ideally like to have not more than a fixed number r of ambulances to shift bases. We refer to this constraint as the *transition constraint*.

Now, the problem can be formally cast as follows:

$$\min_{y,v,x,\tau,\phi} \quad \phi(y) \tag{5.9a}$$

$$\text{s.t.} \quad x_i(t) = 0 \qquad \forall\, i \notin \mathcal{B};\ \forall\, t \in \mathscr{T} \tag{5.9b}$$

$$\sum_{i=1}^{n} x_i(t) \le m \qquad \forall\, t \in \mathscr{T} \tag{5.9c}$$

$$
\begin{aligned}
&v_i(t) = \sum_{j=1}^{n} a_{ji} x_j(t) && \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} && (5.9d)\\
&v_i(t) \leq (\zeta_i - 1) + m\tau_i(t) && \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} && (5.9e)\\
&v_i(t) \geq \zeta_i - m(1 - \tau_i(t)) && \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} && (5.9f)\\
&y_i = \frac{1}{T} \sum_{t=1}^{T} \tau_i(t) && \forall\, i = 1, \ldots, n && (5.9g)\\
&\sum_{i=1}^{n} \tau_i(t) \geq fn && \forall\, t \in \mathscr{T} && (5.9h)\\
&x_i(t) \in \mathbb{Z}_{\geq 0} && \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} && (5.9i)\\
&\tau_i(t) \in \{0, 1\} && \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} && (5.9j)\\
&\sum_{i=1}^{n} |x_i(t+1) - x_i(t)| \leq 2r \forall\, t \in 1, \ldots, T-1 && && (5.9k)
\end{aligned}
$$

Here, $\mathscr{T} = \{1, \ldots, T\}$, $x_i(t)$ is the number of ambulances allotted to zone i at time t. Constraints (5.9b) ensure that allocation occurs only if i is an ambulance base. Here, $\mathcal{B}$ is the set of ambulance bases. Constraints (5.9c) limit the number of ambulances available for allocation in each round. Moreover, a_{ij} is a binary parameter which is 1 if an ambulance can go from i to j within the response threshold time and 0 otherwise. Through constraints (5.9d), $v_i(t)$ counts the number of ambulances that can reach zone i within the response threshold time. Here, $\tau_i(t)$ is 1 if $v_i(t)$ is at least ζ_i, i.e., if the demand of ambulances in zone i is satisfied, and 0 otherwise. This is accomplished by constraints (5.9e) and (5.9f). Constraints (5.9h) take care of efficiency and ensure only allocations that cover at least a fraction f of all zones are to be considered. Finally, constraints (5.9k) are the transition constraints, which can easily be reformulated through linear inequalities. It ensures that not too many ambulances shift bases between consecutive time periods.

The problem in (5.9) is a mixed-integer linear program (MILP). However, the problem is symmetric with respect to certain permutations of the variables. Symmetry makes it particularly hard for modern branch-and-bound-based solvers [111]. Even if one relaxes the transition constraints (5.9k), the symmetry exists. One can observe that, relaxing the transition constraints (5.9k) in (5.9), we have a T-PFA problem. We call this relaxed problem defined by (5.9a) to (5.9j) as the Ambulance-without-transition constraints (AWT) problem.

5.5.2 A Branch-and-Price Reformulation of the AWT

Since the AWT in (5.9a) to (5.9j) is a T-PFA problem, it can be readily written in the form shown in (5.8) and hence can be solved using branch-and-price. First, we note that without the constraint in (5.9k), any feasible solution or optimal solution remains feasible or optimal after permutations to t. Thus, the following version of the problem could be used to solve the relaxed problem without the symmetry. Here we only *count* the number of times each configuration might be used over T periods.

The Master Problem (MP).

$$\begin{aligned}
\min \quad & g - h & & & (5.10a)\\
\text{s.t.} \quad & g \geq y_i & (\alpha_i) \quad & \forall\, i = 1, \dots, n & (5.10b)\\
& y_i \geq h & (\beta_i) \quad & \forall\, i = 1, \dots, n & (5.10c)\\
& y_i = \frac{1}{T} \sum_{j=1}^{k} \tau_{ij} q_j & (\lambda_i) \quad & \forall\, i = 1, \dots, n & (5.10d)\\
& \sum_{j=1}^{k} q_j = T & (\mu) \quad & & (5.10e)\\
& q_j \geq 0 & & \forall\, j = 1, \dots, k & (5.10f)\\
& q_j \in \mathbb{Z} & & \forall\, j = 1, \dots, k. & (5.10g)
\end{aligned}$$

where q_j counts the number of times the configuration defined by x^j is used. The benefit obtained by stakeholder i due to the configuration x^j is τ_{ij}. y_i is the average benefit that stakeholder i enjoys through the time horizon of planning. Note that, once we solve (5.10), an equivalent solution to the AWT could be obtained by arbitrarily considering the allocations x^j for q_j times in $x(1), \dots, x(T)$ of (5.9). Similarly given a solution to AWT, one could immediately identify a corresponding solution to (5.10).

However, since the number of configurations are typically exponentially large and we only might use a handful of them in a solution, we could resort to a branch-and-price approach where (5.10) only contains a subset of the configurations.

Referring to the continuous relaxation of MP as CMP, we write the dual of CMP below.

$$\max \quad T\mu \qquad (5.11a)$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i = 1 && (g) && && \text{(5.11b)}\\
& \sum_{i=1}^{n} \beta_i = 1 && (h) && && \text{(5.11c)}\\
& \lambda_i - \alpha_i + \beta_i = 0 && (y_i) && \forall\, i = 1, \dots, n && \text{(5.11d)}\\
& \mu \le \frac{1}{T} \sum_{i=1}^{n} \tau_{ij} \lambda_i && (q_j) && \forall\, j = 1, \dots, k && \text{(5.11e)}\\
& \alpha_i,\ \beta_i \ge 0 && && \forall\, i = 1, \dots, n. && \text{(5.11f)}
\end{aligned}
$$

Considering only a subset of columns in the CMP is equivalent to considering only a subset of constraints in (5.11e). Given some dual optimal solution $(\alpha^\star, \beta^\star, \lambda^\star, \mu^\star)$ to the dual of the restricted CMP, one can find the most violated constraint in (5.11e) and include the corresponding column in the restricted CMP.

The Pricing Problem.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \tau_i \lambda_i^\star && && \text{(5.12a)}\\
\text{s.t.} \quad & (\tau_i = 1) \iff (a_i^{\mathsf{T}} x \ge \zeta_i), && \forall\, i = 1, \dots, n && \text{(5.12b)}\\
& (\tau_i = 0) \iff (a_i^{\mathsf{T}} x \le \zeta_i - 1), && \forall\, i = 1, \dots, n && \text{(5.12c)}\\
& \sum_{i=1}^{n} x_i \le m && && \text{(5.12d)}\\
& \sum_{i=1}^{n} \tau_i \ge fn && && \text{(5.12e)}\\
& \tau \in \{0, 1\}^n && && \text{(5.12f)}\\
& x \in \mathbb{Z}^n_{\ge 0}. && && \text{(5.12g)}
\end{aligned}
$$

The minimal efficiency constraints are embedded in the pricing problem. The time horizon is fixed in (5.10e).

Constraints (5.12b) and (5.12c) help compute the benefits to each zone, τ and can clearly be rewritten with integer variables and linear constraints. No more than m ambulances may be used (5.12d), and the configuration must cover at least a fraction f of the zones (5.12e).

The reformulation (5.10) of the AWT in (5.9a) - (5.9j), can now be solved very efficiently using branch-and-price. However, a solution thus obtained might violate (5.9k). Further, given the solution in the space of variables in (5.10) it is not immediate if one can easily verify whether the constraint (5.9k) is or is not satisfied. So, given a feasible point for (5.10), we define the configuration graph as follows, and then show that the point satisfies (5.9k) if and only if the configuration graph has a Hamiltonian path.

5.5.3 Checking (5.9k)

Definition 28 (Configuration graph) *Given a solution $\overline{q}$ to* (5.10)*, define $\mathcal{I} = \{j : \overline{q}_j \geq 1\}$. The configuration graph (CG) is defined as $G = (V, E)$, where $V = \{(j, j') : j \in \mathcal{I}; j' \in \{1, 2, \dots, \overline{q}_j\}\}$ and $E = \{((j_1, j_1'), (j_2, j_2')) : \left\|\overline{x}^{j_1} - \overline{x}^{j_2}\right\|_1 \leq 2r\}$ where $\overline{x}^{j_1}$ and $\overline{x}^{j_2}$ are the configurations corresponding to the variables $\overline{q}_{j_1}$ and $\overline{q}_{j_2}$.*

Theorem 29 *Given a point $\overline{q}$ that is feasible to* (5.10)*, there exists a corresponding $\overline{x}$ that is feasible to* (5.9) *if the CG defined by $\overline{q}$ contains a Hamiltonian path. Conversely, if there is a feasible solution to* (5.9) *which uses only the configurations with indices in $\mathcal{I} = \{j : \overline{q}_j \geq 1\}$, then the corresponding CG has a Hamiltonian path.*

Proof 30 *Observe that G has exactly T vertices, since if a configuration is found more than once in $\overline{q}$, it is split into as many distinct vertices in V, which are distinguished by the second element of the tuple. An edge $((u_1, u_2), (v_1, v_2))$ in E indicates that movement between configurations indexed by u_1 and v_1 does not violate the transition constraints. A Hamiltonian path is a path that visits each vertex exactly once. As such, any Hamiltonian path in G proves that a feasible sequence of transitions which does not violate the transition constraints exists between the configurations in $\overline{q}$. Given a Hamiltonian path $(v_1, w_1), (v_2, w_2), \dots, (v_T, w_T)$ in G, a feasible sequence $x(1), x(2), \dots, x(T)$ for* (5.9) *would be $\overline{x}^{v_1}, \overline{x}^{v_2}, \dots, \overline{x}^{v_T}$.*

Conversely, given a feasible sequence $x^{j_1}, x^{j_2}, \dots, x^{j_T}$ for (5.9)*, a Hamiltonian path can be constructed for G as $(j_\alpha, \beta) : \alpha \in \{1, \dots, T\}, \beta = 1 + \max_{\beta'}((j_{\alpha'}, \beta') : \alpha' \in \{1, \dots, \alpha - 1\} \wedge x^{j_\alpha} = x^{j_{\alpha'}})$.* ■

Following Theorem 29, one can construct the CG with exactly T vertices and can check if the solution to (5.10) is feasible to (5.9). If yes, we are done. If not, the way we can proceed is detailed in the rest of this section.

Cutting Planes and Binarization

The most natural way to eliminate a solution that does not satisfy a constraint is by means of a cutting plane. This is a common practice in the computational MILP research, for example, the subtour elimination inequalities in a travelling salesman problem.

In cases where a more sophisticated cutting plane is not available, but every feasible solution is determined by a binary vector, no-good cuts could be used to eliminate infeasible solutions one by one [112]. However, the variables x in (5.10) are general integer variables. It is possible that a point x that violates the transition constraint could lie strictly in the convex hull of solutions that satisfy the transition constraint. Hence it could be impossible to separate x using a cutting plane. Example 31 demonstrates the above phenomenon.

Example 31 *Consider the case where* (5.10) *and* (5.12) *has an optimal solution given by* $q = (q_1, q_2, q_3, q_4) = (1, 1, 1, 1)$ *and the allocations* x_1, x_2, x_3, x_4 *(in the context of* (5.9)*) corresponding to* q_1, q_2, q_3, q_4 *are* $(3, 3, 3, 3)$, $(4, 2, 3, 3)$, $(2, 4, 3, 3)$, $(3, 3, 2, 4)$ *respectively. Let* $r = 1$. *Then, the CG corresponding to this solution is a tree as shown in Figure 5.1 and hence does not have a Hamiltonian path. On the other hand, for the choices* q^i, q^{ii}, q^{iii} *and* q^{iv} *being* $(4, 0, 0, 0)$, $(0, 4, 0, 0)$, $(0, 0, 4, 0)$, $(0, 0, 0, 4)$, *the CGs are all* K_4, *i.e., complete graphs and hence have a Hamiltonian path, trivially.*

Now, it is easy to see that q *lies in the convex hull of* q^i, q^{ii}, q^{iii} *and* q^{iv} *and while each of the latter solutions satisfies the transition constraint,* q *does not. Thus no valid cutting plane can cut only the infeasible point.*

While Example 31 shows that an infeasible solution cannot always be separated using cutting planes, it could still be possible that there is an extended formulation where the infeasible point could be separated.

Naive Binarization. A natural choice of an extended formulation comes from binarizing each integer variable q_j in (5.10). This is possible because each q_j is bounded above by T and below by 0. Thus one can write constraints of the form $q_j = b_j^1 + 2b_j^2 + 4b_j^4 + 8b_j^8 + \dots$ where the summation extends up to the power of 2 less than or equal to T. If each b_j^ℓ is binary, any integer between 0 and T can be represented as above. Having written the above binarization scheme, one can separate any solution q by adding a no-good cut on the corresponding binary variables. A no-good cut is a linear inequality that separates a single vertex of the $0 - 1$ hypercube without cutting off the rest [112, 113]. An example is shown in Example 32.

Example 32 *Consider the problem in Example 31. Binarization to separate the solution* $q = (1, 1, 1, 1)$ *can be done by adding the following constraints to* (5.10).

$$q_j = b_j^1 + 2b_j^2 + 4b_j^4 \qquad \forall\, j = 1, \dots, 4 \tag{5.13a}$$

$$\sum_{j=1}^{4} \left(b_j^1 + (1 - b_j^2) + (1 - b_j^4) \right) \leq 3 \tag{5.13b}$$

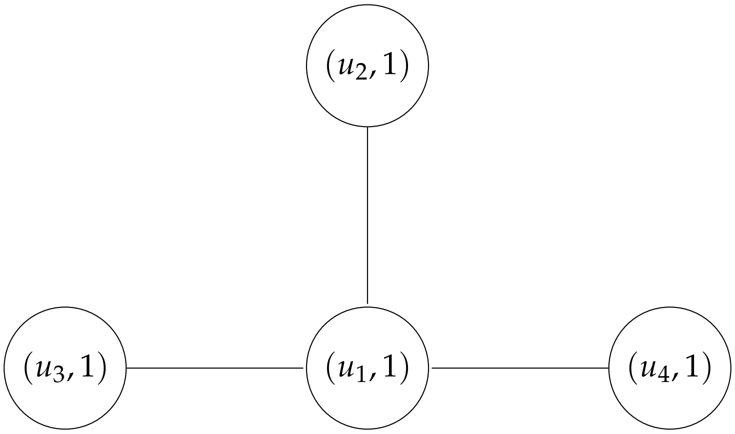


Figure 5.1 Configuration graph for the solution in Example 31.

Note that the second constraint above (the no-good constraint) is violated only *by the binarization corresponding to the solution $u = (1, 1, 1, 1)$ and no other feasible solution is cut off.*

The potential downside with the above scheme is that one might have to cut off a prohibitively large number of solutions before reaching the optimal solution. And with the column generation introducing new q-variables, this could lead to an explosion in the number of new variables as well as the number of new constraints.

Strengthened Binarization. When the CG corresponding to q is not just lacking a Hamiltonian path, but is a disconnected graph (a stronger property holds), one could add a stronger cut, which could potentially cut off multiple infeasible solutions. In this procedure, we add a binary variable b_j for each q_j such that $b_j = 1$ if and only if $q_j \geq 1$. Now, observe that if a CG corresponding to the solution $\overline{q}$ is disconnected, then it will be disconnected for all q whose non-zero components coincide with the non-zero components of $\overline{q}$. i.e., no solution with the same support as that of $\overline{q}$ could satisfy the transition constraints. Thus one could add a no-good cut on these b_j binary variables, which cuts off all the solutions with the same support as $\overline{q}$.

Unlike naive binarization, while this cuts off multiple solutions simultaneously, it could happen that the CG is connected, but just does not have a Hamiltonian path. In such a case, no cut could be added by the strengthened binarization scheme, and one might have to resort to the naive version.

Three-Way Branching

An alternative to the naive binarization scheme is three-way branching. While this could work as a stand-alone method, it could also go hand in hand with the strengthened binarization mentioned earlier. This method takes advantage of the three-way branching feature that some solvers, for example, SCIP [114, 115], have.

In this method, as soon as a solution $\overline{q}$ satisfying all the integrality constraints but violating the transition constraint is found, the following actions are performed. First, if the lower bound and the upper bound for every component of $\overline{q}$ match, then we are at a leaf that can be discarded as infeasible. If not, find a component j (in our case, a configuration j) such that $\overline{q}_j$ is strictly different from at least one of the bounds. Now, we do a three-way branching on the variable q_j where the new constraints in each of the three branches are (i) $q_j \leq \overline{q}_j - 1$ (ii) $q_j = \overline{q}_j$ (iii) $q_j \geq \overline{q}_j + 1$.

Finite termination follows from the fact that $\overline{q}_j$ is infeasible for branches (i) and (iii) and hence will never be visited again. For branch (ii), we have q_j such that its lower and upper bounds are equal to $\overline{q}_j$. Thus, we have one more fixed variable and this variable will never be branched on again.

Three-way branching is used as opposed to regular two-way branching but on integer variables because of the following reasons. First, branching with (i) $q_j \leq \overline{q}_j - 1$ (ii) $q_j \geq \overline{q}_j + 1$ is invalid, as it could potentially cut off other feasible solutions with $q_j = \overline{q}_j$ but the solution differing in components other than j. Branching with (i) $q_j \leq \overline{q}_j - 1$ (ii) $q_j \geq \overline{q}_j$ could cycle, as we have not fixed any additional variable in the second branch, nor have we eliminated the infeasible solution. Thus, the LP optimum in the second branch is going to be $\overline{q}$ again, and cycling ensues.

Constraint Programming

The final alternative we can use to enforce transition constraints (5.9k) is constraint programming (CP). Namely, CP is a programming paradigm for solving combinatorial problems. A CP model is defined by a set of variables, each of which is allowed values from a finite set, its *domain*. The relationship between these variables is determined by the constraints of the problem. These succinct constraints can encapsulate complex combinatorial structures, such as the packing of items into bins, for example. A solver then solves the problem by enforcing consistency between the variables and the constraints, and using branching and backtracking techniques to explore the solution space. Before defining the CP model to enforce constraints (5.9k), we define the compact configuration graph, and explain its relationship with the CG.

Definition 33 (Compact configuration graph) *Given a feasible solution $q^\star$ to the continuous relaxation of* (5.10) *(CMP), the compact configuration graph (CCG) is defined as $G = (V, E)$, where $V = \mathscr{A} := \{j : q_j^\star > 0\}$ and $E = \{(v, w) : \|\overline{x}^v - \overline{x}^w\|_1 \leq 2r, \forall v, w \in \mathscr{A}\}$.*

Definition 34 (Walk) *A* walk *in an undirected graph $G = (V, E)$ is a finite sequence of vertices $v_1, \ldots, v_k$, not necessarily distinct, such that for each i, $(v_i, v_{i+1}) \in E$.*

Theorem 35 *Every walk of length T in a CCG constructed from a solution $q^\star$ to CMP corresponds to a feasible solution of* (5.9)*.*

Proof 36 *Let $G = (V, E)$ be the CCG given a solution $q^\star$. Let $W = v_1, v_2, \ldots, v_T$ be a walk of length T in G. Since we allow revisiting vertices, it is possible that $v_j = v_{j'}$ for some $j \neq j'$.*

Now define $\widetilde{q}$ component-wise where $\widetilde{q}_j$ corresponds to the number of times the vertex j is visited in the walk W. Since the walk has a length T, trivially $\sum_j \widetilde{q}_j = T$, satisfying (5.10e)*. Then, $\widetilde{y}, \widetilde{q}$ can be defined so that we have a feasible solution to* (5.10)*. Hence, if we now show that the CG defined by the nonzero components of $\widetilde{q}$ has a Hamiltonian path, then the corresponding $\widetilde{x}$ will be feasible for* (5.9) *due to Theorem 29.*

Now, in the CG, construct the path $P = (v_1, n(v_1) + 1), (v_2, n(v_2) + 1), \ldots, (v_T, n(v_T) + 1)$ where $n(v_j)$ records the number of times v_j has appeared in the path P earlier so far. We note that each term in the path P is indeed a vertex of the CG (which has T vertices) and that they are all visited exactly once, implying that P is the required Hamiltonian path. ■

The general mechanism involving the CP component is provided in Algorithm 1, which is the entire algorithm we test in Section 5.6. The CP component checks whether the solution returned by the CMP can be made valid in some way, i.e., if there exist solutions using only the configurations in $\mathscr{A}$ (i.e., the configurations that appear in the optimal CMP solution, see Definition 33) with integer values, such that they do not violate the transition constraints. By providing these feasible solutions, it provides an upper bound to (5.9). As soon as a set of configurations is given to CP, a cut is added to the CMP, which eliminates all solutions to CMP which only consist of the configurations provided to CP. Namely, if $\mathscr{A}$ indices the configurations given to CP, we add a cut $\sum_{j \in \mathscr{A}} q_j \leq T - 1$, indicating that at least one of the configurations must be outside the set indexed by $\mathscr{A}$.

Let k' be the cardinality of $\mathscr{A}$. A CCG is associated with solution q — this CCG forms the basis for a deterministic finite automaton. This automaton A is defined by a tuple $(Q, \Sigma, \delta, q_0, F)$ of states Q, alphabet Σ, transition function $\delta : Q \times \Sigma \rightarrow Q$, initial state

$q_0 \in Q$, and final states $F \subseteq Q$, with $Q = \{0,\ldots,k'\}$, $\Sigma = \{1,\ldots,k'\}$, $\delta = \{(u,v) \to v : (u,v) \in E\} \cup \{(0,u) \to u : u \in F\}$, $q_0 = 0$, and $F = \{1,\ldots,k'\}$. In other words, this automaton accepts any valid sequence of k' configurations, with dummy configuration $q_0 = 0$ being the initial state. Let LB be the lower bound given by the CMP, and UB be an upper bound. Finally, let Ω be the collection of all index sets $\mathscr{A}'$ for which the CP model has been previously solved. There are T decision variables z with domains $\{0,\ldots,k'-1\}$, with z_t indicating which CMP configuration from $\mathscr{A}$ is used at time point t. These variables are constrained by

$$\begin{aligned}
&\min \phi && && (5.14a)\\
&\phi = \texttt{max}(c) - \texttt{min}(c) && && (5.14b)\\
&LB \leq \phi \leq UB - 1 && && (5.14c)\\
&\texttt{cost\_regular}(z, A, \tau_{i\star}, c_i) && i = 1,\ldots,n && (5.14d)\\
&\texttt{at\_most}(T-1, z, \omega) && \forall \omega \subset \mathscr{A} : \exists \mathscr{A}' \in \Omega,\ \omega \subseteq \mathscr{A}' && (5.14e)\\
&z_t \in \mathscr{A} && t = 1,\ldots,T && (5.14f)\\
&c_i \in \mathbb{Z}_{\geq 0} && i = 1,\ldots,n && (5.14g)
\end{aligned}$$

The objective (5.14a) is to minimize the unfairness, i.e., the difference between the zone which is covered the most, and that which is covered the least (5.14b). Any feasible solution should be strictly better than the upper bound, and search by the CP solver can be interrupted as soon as a feasible solution is found whose objective value coincides with LB (5.14c). By Theorem 35, any sequence of configurations indexed by $\mathscr{A}$ and of length T must correspond to a feasible solution of (5.9). This requirement is enforced by the `cost-regular` [116] constraints (5.14d): A `cost-regular` constraint holds for zone i if z forms a word recognized by automaton A and if variable c_i is equal to the coverage of zone i over T: $c_i = \sum_{t=1}^{T} \tau_{i,z_t}$. Note that the CP model does not require that all configurations indexed by $\mathscr{A}$ be used at least once: Since it checks all possible subsets of $\mathscr{A}$, the cut added to CMP does not remove any unexplored part of the search space. Finally, all subsets $\omega \subset \mathscr{A}$ previously explored, i.e. being as well a subset of some index set $\mathscr{A}'$ in Ω, are considered by the model using the `at_most` constraint (5.14e): At most $T-1$ variables in z can take on values in ω. The motivation for this is illustrated in Example 37.

Example 37 *Assume that $\mathscr{A} = \{j', j'', j'''\}$. If* (5.14) *had previously solved $\mathscr{A}' = \{j', j''\}$, this solution space would be explored again since the configurations in $\mathscr{A}$ are not constrained to be used* *at least once. Adding constraint* $\texttt{at\_most}(T-1, z, \{j', j''\})$ *in the current iteration avoids this.*

The CP solver we use is OR-Tools, which currently does not implement the `cost-regular` constraint. We thus replaced (5.14b) and (5.14d) with the equivalent but less efficient[5] reformulation

$$\begin{aligned}&\phi = \max_{i=1}^{n} \sum_{t=1}^{T} \tau_{i,z_t} - \min_{i=1}^{n} \sum_{t=1}^{T} \tau_{i,z_t} \\ &\texttt{regular}(z, A).\end{aligned}$$

5.5.4 The Final Algorithm

The general working of the final algorithm is presented formally in Algorithm 1.

In each iteration of the final algorithm, we solve a linear program and make a call to the constraint programming solver. The linear program is basically the continuous relaxation of (5.10) with any subsequent cutting planes (Step 6 in Algorithm 1). This is solved using column generation. The configurations which receive non-zero weights (indexed by $\mathscr{A}$) in the optimal LP solution are passed to the constraint programming solver to detect whether there exists a solution to (5.9) that uses only configurations indexed by $\mathscr{A}$. Meanwhile, a cut is added to the linear program, that eliminates all solutions which use only the configurations indexed by $\mathscr{A}$.

With this, the large set of possible configurations are managed by the linear programming solver, which is powerful and efficient for large problems, while the CP-based solver only solves instances with a handful of configurations at a time.

5.6 Computational Experiments

In this section, we introduce the setting that we have used to evaluate computationally Algorithm 1 and we discuss the results of such an evaluation in the context of ambulance location and relocation. In particular, we use real data from the city of Utrecht, Netherlands, as well as synthetically-generated instances of varying sizes. Moreover, we consider a predetermined time horizon, i.e., a fixed T of size 30, with the understanding that a decision maker could reasonably make plans on a monthly basis.

[5] In practice, the CP component of Algorithm 1 remains very fast, so this loss in efficiency is negligible.

Algorithm 1 The Final Algorithm

```
Input: The ambulance allocation problem with number of zones $n$, the bases $\mathcal{B}$, $\zeta_i$, for
    $i = 1, \ldots, n$, $f$ and $a_{ih}$ for $i, h = 1, \ldots, n$, $r \in \mathbb{Z}_{\geq 0}$ and $T \in \mathbb{Z}_+$.
Output: $x(1), \ldots, x(T)$ that is optimal to (5.9).
$LB \leftarrow -\infty, UB \leftarrow +\infty$. $\mathscr{C} \leftarrow \emptyset$. $x^\star(1), \ldots, x^\star(T) = NULL$.
while $UB > LB$ do
    Solve CMP, i.e., the continuous relaxation of (5.10) after adding each constraint in
    $\mathscr{C}$. Let $q^\star$ be the optimal solution and $o^\star$ be the optimal objective value.
    $LB \leftarrow o^\star$.
    $\mathscr{A} \leftarrow \{j : q_j^\star > 0\}$.
    $\mathscr{C} \leftarrow \mathscr{C} \cup \left\{ \sum_{j \in \mathscr{A}} q_j \leq T - 1. \right\}$.
    $(x^\dagger(1), \ldots, x^\dagger(T)), o^\dagger \leftarrow$ ConstraintProgramming$(q^\star, n, a, T, LB, UB, \mathscr{C})$.
    if $o^\dagger < UB$ then
        $UB \leftarrow o^\dagger$ and $(x^\star(1), \ldots, x^\star(T)) \leftarrow (x^\dagger(1), \ldots, x^\dagger(T))$.
    end if
end while
return $x^\star(1), \ldots, x^\star(T)$

function ConstraintProgramming$(q, n, a, T, LB, UB, \mathscr{C})$
    $G \leftarrow$ CCG defined by $\mathscr{A}$.
    Solve (5.14) on the graph $G$ with appropriate values of $\tau, c$.
    if (5.14) is infeasible then
        return $NULL$, $+\infty$
    else
        return $(x^\dagger(1), \ldots, x^\dagger(T)), o^\dagger$.
    end if
end function
```

Utrecht instance. We use the model defined in Section 5.5 to determine ambulance allocation in the city of Utrecht, Netherlands, using a dataset provided by the RIVM.[6] The Utrecht instance contains 217 zones, 18 of which are bases. Since calls should be reached within 15 minutes, we consider that a zone is connected to another if an ambulance can travel between the two zones in under 15 minutes. This makes the graph about 28.4% dense.

The efficiency measure we impose is that at least 95% of all the zones should be covered at all times. We consider that a zone is covered if sufficiently many ambulances are in the vicinity of that zone. In turn, we define the sufficient number of ambulances for a zone to

[6] National Institute for Public Health and the Environment of the Netherlands.

be 1, 2, 3, or 4, based on the population density of the zone.

Synthetic instances. Reproducing the ratio of bases and edges to zones, we also generated synthetic instances[7] of sizes 50, 100, 200, and 400 zones, using the Matérn cluster point process [117], which helps in generating a distribution of zones mimicking a realistic urban setting. The number of ambulances for the instances is chosen to be just enough to ensure feasibility. The results are averaged over five instances of each size.

Testing environment and software. Testing was performed on an Intel 2.8 GHz processor with 8 GBs of RAM running Arch Linux, and the models were solved with Gurobi 9.0.1 and OR-Tools 8.0. A time limit of 1,200 seconds was imposed.

Results. It is easy to see that there are two parameters that are likely to influence the difficulty of the solving process: the size of the instance (number of zones) and the transition constraints, i.e., the flexibility we allow for relocating ambulances between zones on a daily basis. We would typically expect the size of the instance to be a significant factor, but this is likely to be mitigated by the column generation approach, which tends to scale well. The effects that the transition constraints will have on the solving process, however, are less clear. In order to assess such an effect, for each instance, we vary the number of ambulances that can shift bases on consecutive days (r in constraints (5.9k)) from $0.1m$ to m in increments of $0.1m$, where m is the total number of ambulances in the instance. We call this ratio maximum transition, MT. For each of these cases, we record the average of the time taken to solve these instances to optimality (capped at 1,200 CPU seconds). We record the final relative gap left for the instance, which is defined by $\frac{|UB_f - LB_f|}{UB_f}$, where LB_f and UB_f represent the values of the best lower and upper bounds at the time limit, respectively. We also record the initial relative gap for the instance, which is defined by $\frac{|UB_i - LB_i|}{UB_i}$, where LB_i represents the value of the initial lower bound (without any cuts), and UB_i represents the value of the initial, trivial upper bound. We present the final relative gaps associated with different classes of instances in Figure 5.2.

One can immediately observe from Figure 5.2 that the transition constraints are affecting the difficulty of the instances for $MT \leq 0.5$, while everything can be solved to optimality within the time limit for $MT \geq 0.5$ independently of the number of zones. The larger MT the smaller the final gap, i.e., when there is more freedom in moving the ambulances

[7] These instances can be found at https://github.com/PhilippeOlivier/ambulances.

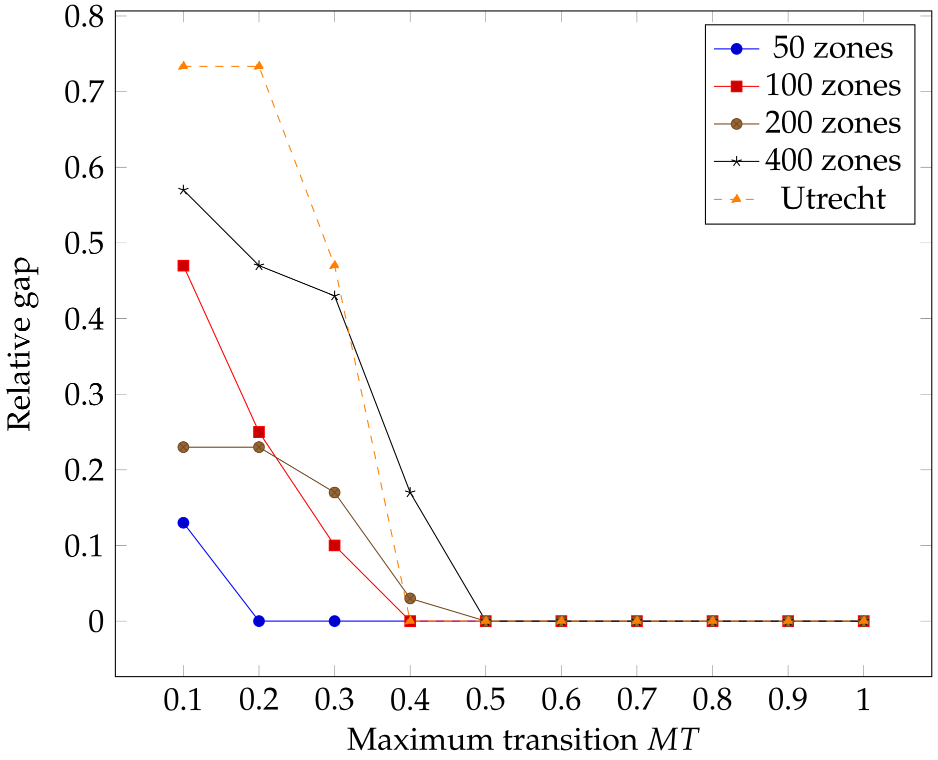


Figure 5.2 Final relative gaps with respect to the maximum transition MT. Gaps in the synthetic instances are averaged over 5 instances. Time limit of 1,200 CPU seconds.

around on a daily basis, Algorithm 1 becomes very effective in computing optimal solutions.

We observe that for varying values of MT, the Utrecht instance has a higher relative gap than synthetic instances of comparable and even larger size. This discrepancy is the result of the distribution of the population densities across the zones. In the synthetic instances, the population densities are randomly distributed among the zones. The Utrecht instance, in contrast, exhibits several distinct clusters of varying sizes and population density distributions.[8]

Unsurprisingly, Figure 5.2 also suggests that larger instances are more difficult because, typically, the relative gap is large when time limits are hit. This is confirmed in more details from the results in Table 5.1, where we report, for every group of instances and every MT value, the initial gap (“i.gap"), the final gap at the time limit (“f.gap"), the number of

[8] Constructing these sophisticated clusters is not a trivial task, which is why we settled on a random distribution of population densities for the synthetic instances. By randomizing the placement of the population for the Utrecht instance, its gap becomes similar to that of the synthetic instances.

generated columns (“#cols"), the number of solved instances (“#solved"),[9] and the average time for the instances solved to optimality (“time").

Table 5.1 Detailed results for Algorithm 1 on both synthetic and real-world Utrecht instances. Time limit of 1,200 CPU seconds. The results of the synthetic instances are averaged over 5 instances.

	50 zones					100 zones					200 zones				
MT	i.gap	f.gap	#cols	#solved	time	i.gap	f.gap	#cols	#solved	time	i.gap	f.gap	#cols	#solved	time
0.1	0.13	0.13	738	4	0.3	0.55	0.47	1120	1	0.2	0.23	0.23	714	3	2.5
0.2	0.13	0.00	3	5	0.4	0.55	0.25	674	2	1.2	0.23	0.23	710	3	2.5
0.3	0.13	0.00	3	5	0.4	0.55	0.10	11	4	2.5	0.23	0.17	537	3	2.6
0.4	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.7	0.23	0.02	108	4	82.7
0.5	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.3	0.23	0.00	19	5	7.0
0.6	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.1	0.23	0.00	15	5	7.2
0.7	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.1	0.23	0.00	15	5	6.7
0.8	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.2	0.23	0.00	15	5	6.0
0.9	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.2	0.23	0.00	15	5	6.0
1	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.1	0.23	0.00	15	5	6.0

	400 zones					Utrecht				
MT	i.gap	f.gap	#cols	#solved	time	i.gap	f.gap	#cols	#solved	time
0.1	0.57	0.57	248	1	2.7	0.73	0.73	786	0	-
0.2	0.57	0.47	246	1	2.7	0.73	0.73	803	0	-
0.3	0.57	0.43	238	1	2.7	0.73	0.47	757	0	-
0.4	0.57	0.17	174	3	147.4	0.73	0.00	353	1	315.6
0.5	0.57	0.00	93	5	273.6	0.73	0.00	252	1	122.3
0.6	0.57	0.00	65	5	63.6	0.73	0.00	96	1	49.6
0.7	0.57	0.00	60	5	49.6	0.73	0.00	96	1	50.3
0.8	0.57	0.00	60	5	47.1	0.73	0.00	96	1	48.4
0.9	0.57	0.00	60	5	47.4	0.73	0.00	96	1	49.8
1	0.57	0.00	60	5	49.1	0.73	0.00	96	1	54.7

The results in Table 5.1 show that for $MT > 0.7$ constraints (5.9k) are not binding, i.e., they do not cut off the obtained optimal solution to the rest of the model, and Algorithm 1 behaves exactly like for $MT = 0.7$. The average time for the instances solved to optimality is often low, thus indicating that if we manage to prove optimality by completely closing the gap, then this is achieved quickly, generally with a few calls to ConstraintProgramming. In contrast, if we do not succeed in closing the gap early, then it is unlikely to be closed at all in the end. In the cases where optimality is not proven, improvements are still made during the solving process mostly because of improvements on the upper bound value as ConstraintProgramming tests more and more combinations of configurations. This suggests a few things. First, with a high enough MT, the initial lower

[9] The entry “#solved" takes an integer value between 0 and 5 for synthetic instances and 0/1 for the Utrecht instance.

bound is generally optimal, and feasible solutions whose objective values coincide with this bound can readily be found. Second, when the MT value is low, either feasible solutions whose objective values coincide with the initial lower bound are very rare, or they are nonexistent. Third, the computational power of Algorithm 1 is associated with the effectiveness in searching for primal solutions (upper bound improvement) versus the progress on the dual side (lower bound improvement), which appears to be very difficult.

Finally, we are also interested in identifying the time spent in the column generation part as opposed to the CONSTRAINTPROGRAMMING routine. To this end, we track the number of calls made to the function CONSTRAINTPROGRAMMING and also measure the total time spent in calls to the function. The number of generated columns and the number of calls to CONSTRAINTPROGRAMMING are detailed in Table 5.2.

Table 5.2 Number of columns and calls to CONSTRAINTPROGRAMMING associated with the maximum transition (MT) constraints of the various instance sizes.

	50 zones		**100 zones**		**200 zones**		**400 zones**		**Utrecht**	
MT	#cols	#calls	#cols	#calls	#cols	#calls	#cols	#calls	#cols	#calls
0.1	738	736	1120	1108	714	608	248	72	786	210
0.2	3	1	674	662	710	605	246	72	803	217
0.3	3	1	11	3	537	435	238	66	757	202
0.4	3	1	9	1	108	58	174	22	353	54
0.5	3	1	9	1	19	3	93	6	252	21
0.6	3	1	9	1	15	1	65	2	96	1
0.7	3	1	9	1	15	1	60	1	96	1

The results in Table 5.2 show that when enough freedom in the day-to-day movement of ambulances is allowed, optimality can generally be proven immediately, with just one call to the CONSTRAINTPROGRAMMING routine. We also notice that the ratio of columns to the number of calls to the routine increases with the instance size. Such a ratio for the Utrecht instance is disproportionately high, due again to the distribution of the population densities.

Finally, Table 5.3 shows the average amount of time spent generating columns (“CG"), that spent enforcing the transition constraints through CONSTRAINTPROGRAMMING (“CP") and the ratio between these two CPU times (“ratio") for the most difficult case, i.e., $MT = 0.1$.

The results in Table 5.3 show that generating columns for larger instances (more zones) takes naturally more time, which is why this effort increases with n. The enforcement of the transition constraints, however, requires a more stable effort, as the input of CON-

Table 5.3 CPU times in seconds spent in column generation and ConstraintProgramming, with $MT = 0.1$.

	time		
Instance	CG	CP	ratio
50 zones	86	154	0.56
100 zones	411	550	0.75
200 zones	331	151	2.19
400 zones	711	253	2.81
Utrecht	1071	129	7.96

straintProgramming is always around three to ten variables, regardless of the size of the instance.

5.7 Conclusion

We introduced an abstract framework for solving a sequence of fair allocation problems, such that fairness is achieved over time. For some relevant special cases, we have been able to give theoretical proofs for the time horizon required for perfect fairness. We described a general integer programming formulation for this problem, as well as a formulation based on column generation and constraint programming. This latter formulation can be used in a practical context, as shown by the ambulance location problem applied to the city of Utrecht. The largest synthetic instances suggest that this formulation would scale well to regions twice the size of Utrecht, given that the freedom of movement of the ambulances is not overly restricted.

Acknowledgements

The authors would like to thank the National Institute for Public Health and the Environment of the Netherlands (RIVM) for access to the data of their ambulance service.

CHAPTER 6 GENERAL DISCUSSION

The three articles which comprise this thesis came about rather organically. As the final project of a constraint programming class which I took early in my PhD, we were asked to solve a combinatorial problem of our choice, using CP. I opted for the *Wedding Seating Problem* (WSP). In this problem, groups of guests are to be seated at tables for a wedding reception. They have some preference to be seated with one group or another, and some groups cannot be seated together at all. The objective is to construct tables of people who will be happy together, while also balancing the occupancy of the tables. This project was later expanded, and the resulting paper was published in the proceedings of the 2018 CPAIOR conference [58]. This work is the precursor of Chapter 3.

The reader will recall that this thesis is titled *Fairness in Combinatorial Optimization*, and that its main objectives are to show how one might go about including different ways of achieving balance, in various types of optimization frameworks. The first step toward this goal was to find a suitable combinatorial problem which would allow for the study of fairness. This was the motivation for solving the WSP (and the *Quadratic Multiknapsack Problem* with conflicts and balance constraints of Chapter 3, which is essentially the same problem) with various CP and IP models. This problem is a fitting one as it has an optimization objective separate from the balance requirements, which is the kind of dynamic that is often found in practice. The problem also exhibits a structure allowing it to be modeled by column generation, thus expanding the study of balance in IP.

We started with the implementation of three models to solve this problem: one CP model, one natural IP model, and one IP model using column generation along with branch and price. These models varied in the complexity of their implementation, in their flexibility to solve various types of instances, and in their performance.

The core of the CP model, as is often the case with this paradigm, is compact and was fairly quick to implement. Of the three approaches, the CP model is the most straightforward one with respect to balancing, as this is taken care of by a single `dispersion` constraint, which conceals all the associated complexity and greatly simplifies the task of the modeler. We take note, however, of two particularities directly related to balance, which affect one's mindset during modeling. Since we're trying to optimize the objective while enforcing strict balance constraints, this leads us to consider two different avenues (objective-based or balance-based) when deciding on branching strategies and symmetry breaking. So, while the balance constraint itself is simple to add to the model, it also creates new opportunities for customized branching and symmetry breaking. This is somewhat of a double-edged sword, as these options are tempting, but time-consuming, to explore. We spent a considerable amount of time trying to improve the performance of the model by customizing the branching strategy, both via strategies related to the objective, and to balance. Unfortunately, we found that despite exploiting what we knew of the structure of the problem, we were unable to achieve a better performance than the default search of CP Optimizer.[1] We also tested many combinations of various symmetry-breaking constraints, some based on the objective and some based on balance. In the end, we settled on two, related to balance, which could be used concurrently.

For the natural IP model, implementation was more complicated, especially for the L_0- and L_2-norms. The L_0-norm required several auxiliary variables and extra constraints, making the overall model convoluted, while the L_2-norm could only be approximated by a relaxed model. We have found that the performance of the natural IP model is close to that of the CP model. Considering the difficulties and limitations of the former, and the simplicity of the latter, we would opt for CP over a natural IP model for most balancing norms.

The IP model based on column generation and branch and price was the most complicated and lengthy to implement, but provided the best results. Most of the difficulty for this model lies in the pricing problem and in the branch-and-price algorithm. First, the constraints of the pricing problem closely resemble those of the natural IP model, and even the CP model that we settled on for the pricing problem remained complicated as it was modeled with 0-1 variables, and used as many constraints as the natural IP model. The second difficulty was with the branch-and-price algorithm. Unlike other solvers such as SCIP, which include a branch and price mechanism, the solver that we used, CPLEX, did not—we thus had to implement branch and price from scratch.

Solving this problem with different models that use various balancing norms, however, did not paint the whole picture and left some questions still open. All norms achieve balance in different ways, but what can we expect from one norm or another? Are some norms more suited than others to solve a particular problem? And if so, how does one go about determining this? Understanding what the use of a particular norm entails would be a step toward answering these questions.

In the course of reviewing the literature, [81] became the starting point for this investigation. In this paper, the authors compare how the balanced solutions given by four norms

[1] Note that this default search is quite sophisticated, and is designed such that users are only bothered with the modeling part.

are able to approximate each other. They solve the *Balanced Academic Curriculum Problem* (BACP), and found that the balance criterion generally used for this problem could be replaced with a superior one. Perhaps, we thought, are there other problems where the usual way of achieving balance could be reevaluated, and where lessons could be learned. Indeed, similarly to the previous problem, [84] found that the *Nurse-Patient Assignment Problem*, as it is generally defined, could also have its balance component improved.

After some more research, we came across the *Balancing Bike Sharing Systems* problem, in which we believed balance could also be enhanced. We experimentally showed that balance could in fact be improved in this problem, which resulted in a more robust system. With the experience gained through the study of these three problems, we compiled a list of guidelines which should help a modeler choose an appropriate way of achieving balance for a given problem. In contrast with the first paper, which included the definition of multiple original models, this was one of methodology, where the focus was on the characteristics of the solutions, rather than on the path leading to them.

The third paper explored fairness from another angle, by modeling a special type of problem where fairness is considered over a time horizon, and presenting some related theoretical proofs. Many of these proofs are of particular importance for this type of problem as they show, for example, that it does not suffice to prove optimal fairness for each individual period of time in order to prove optimality for the time horizon. The model we introduce constructs optimally fair allocations with an IP model based on column generation, and uses a CP model to check if such allocations are feasible with respect to specific constraints which are ill-suited to be directly included in the IP model. This modeling choice creates a nice dynamic between the two optimization methods used in this thesis.

It is easy to make the case that, for several applications, there is little difference between various balancing schemes, as they all offer solutions that, in practice, work well enough for the situation at hand. This paper, in a way, also developed as a contrasting illustration of the necessity of optimal fairness for some problems. Were a man to file a complaint against his local emergency services, stating that his father has died as a result of a poor ambulance response time, answering his plea by declaring that fairness in the system is "good enough" is likely not going to cut it. Optimal fairness provides a formal proof that, truly, the best that could be done with the available limited resources has been done.

CHAPTER 7 CONCLUSION

In this thesis, we studied fairness in the context of combinatorial optimization, in particular within the CP and IP frameworks. In what follows, we highlight our contributions, discuss the limitations of our work, and consider future research directions.

7.1 Contributions

We will recall the objectives of this thesis, as presented in Section 1.4.1:

1. Study the intrinsic balance characteristics of various balancing methods;
2. Present ways of integrating these balancing methods into CP and IP models;
3. Compare the implementation effort and the performance of these models.

In the first article, we studied a generalization of the *quadratic multiknapsack problem*, which includes conflicts and balance constraints. We presented various models based on CP and IP, comparing both the complexity of their implementation, as well as their performance. These models implemented several different measures of fairness. This article fulfills the second and third research objectives.

The focus of the second article was on methodology. We showed several problems for which balance could be improved, and presented some characteristics of balancing methods. We discussed a list of guidelines that one should follow when adding a fairness component to a model. This article fulfills the first research objective.

The third article tackled a special case where fairness was considered over a period of time. We presented some theoretical results for special cases, then provided an IP formulation for more complex cases. We have shown how this can be used in the problem of locating ambulance, for which we introduced a model based on column generation, with IP and CP components. This article puts the lessons learned in the two previous articles into practice by finding a fair and efficient method to solve a problem of practical interest.

7.2 Limitations and Future Work

It is inevitable that the deeper one digs into a subject, and the stronger one feels that one is standing in a shallow hole. This thesis is no exception.

In the whole of this research, we consider but a small number of measures of balance. Some more can be found in the literature (e.g., minimizing the sum of all pairwise distances in a set of values), and even more can still be imagined. It would be conceivable to expand the study toward a greater number of measures of balance. In turn, this would open up additional avenues of research. We may be able to devise proofs showing that some measures are equivalent to others, or even dominate others.

We only considered CP and IP in this thesis. Yet, other methods exist. It would be interesting to compare the CP and IP approaches studied in the first article to other exact optimization methods, such as decision diagrams. We also have not touched upon the subject of balancing in the context of metaheuristics, such as genetic algorithms. These could prove useful to mitigate the limitations of exact methods when the problems scale to very large sizes.

In the second article, we introduced a few properties which can characterize the measures of balance, namely dispersion, smoothness, and the number of outliers. This set of properties could be expanded, which would lead to suggesting additional guidelines to a potential modeler. The literature could also be further surveyed to identify more problems where balance could be improved.

REFERENCES

[1] S. J. Brams, M. A. Jones, and C. Klamler, "Better ways to cut a cake," *Notices of the American Mathematical Society*, vol. 53, no. 11, Dec. 2006.

[2] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*, 1st ed. Athena Scientific, 1997.

[3] G. Belov and G. Scheithauer, "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting," *European Journal of Operational Research*, vol. 171, no. 1, pp. 85–106, May 2006.

[4] J. Desrosiers and M. E. Lübbecke, "A primer in column generation," in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds. Boston, MA: Springer US, 2005, pp. 1–32.

[5] E. C. Freuder and A. K. Mackworth, "Constraint satisfaction: An emerging paradigm," in *Handbook of Constraint Programming*, 2006.

[6] K. W. Whiteacre, "Testing the level of service inventory–revised (LSI-R) for racial/ethnic bias," *Criminal Justice Policy Review*, vol. 17, no. 3, pp. 330–342, 2006.

[7] S. Corbett-Davies *et al.*, "Algorithmic decision making and the cost of fairness," *CoRR*, vol. abs/1701.08230, 2017.

[8] 2nd United States Congress, "An act for apportioning representatives among the several states, according in the first enumeration," April 1792.

[9] W. F. Willcox, "The apportionment of representatives: Annual address of the President," *The American Economic Review*, vol. 6, no. 1, pp. 3–16, 1916.

[10] C. W. Seaton, "Report of the superintendent of census," in *Annual Report of the Secretary of the Interior on the Operations of the Department for the Year Ended in June 30, 1881*. Washington: Government Printing Office, 1882, vol. 2, p. 677.

[11] E. V. Huntington, "A new method of apportionment of representatives," *Quarterly Publications of the American Statistical Association*, vol. 17, no. 135, pp. 859–870, 1921.

[12] D. Desilver. (2019, May) Despite global concerns about democracy, more than half of countries are democratic. [Online]. Available: https://www.pewresearch.org/fact-tank/2019/05/14/more-than-half-of-countries-are-democratic

[13] S. L. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Operations Research*, vol. 12, no. 3, pp. 450–459, Jun. 1964.

[14] M. T. Marsh and D. A. Schilling, "Equity measurement in facility location analysis: A review and framework," *European Journal of Operational Research*, vol. 74, no. 1, pp. 1–17, 1994.

[15] N. Kumar and K. Shanker, "Comparing the effectiveness of workload balancing objectives in FMS loading," *International Journal of Production Research*, vol. 39, no. 5, pp. 843–871, 2001.

[16] T. Bektaş and A. N. Letchford, "Using ℓ^p-norms for fairness in combinatorial optimisation," *Computers & Operations Research*, vol. 120, 2020.

[17] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 3323–3331.

[18] C. Dwork *et al.*, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 214–226.

[19] D. Bertsimas, V. F. Farias, and N. Trichakis, "Fairness, efficiency, and flexibility in organ allocation for kidney transplantation," *Operations Research*, vol. 61, no. 1, pp. 73–87, 2013.

[20] G. Pesant and J.-C. Régin, "Spread: A balancing constraint based on statistics," in *Principles and Practice of Constraint Programming – CP 2005*, P. van Beek, Ed. Springer Berlin, 2005, pp. 460–474.

[21] P. Schaus *et al.*, "Simplification and extension of the spread constraint," in *Third International Workshop on Constraint Propagation and Implementation*, Jan. 2006, pp. 77–91.

[22] P. Schaus and J.-C. Régin, "Bound-consistent spread constraint," *EURO Journal on Computational Optimization*, vol. 2, no. 3, pp. 123–146, Aug. 2014.

[23] P. Schaus *et al.*, "The deviation constraint," *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 260–274, 2007.

[24] P. Schaus, Y. Deville, and P. Dupont, “Bound-consistent deviation constraint,” in *Principles and Practice of Constraint Programming – CP 2007*, C. Bessière, Ed. Springer Berlin, 2007, pp. 620–634.

[25] J.-N. Monette *et al.*, “A parametric propagator for discretely convex pairs of sum constraints,” in *Principles and Practice of Constraint Programming*, C. Schulte, Ed. Springer Berlin, 2013, pp. 529–544.

[26] ——, “A parametric propagator for pairs of sum constraints with a discrete convexity property,” *Artificial Intelligence*, vol. 241, pp. 170–190, 2016.

[27] G. Pesant, “Achieving domain consistency and counting solutions for dispersion constraints,” *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 690–703, 2015.

[28] C. Bessiere *et al.*, “The balance constraint family,” in *Principles and Practice of Constraint Programming*, B. O’Sullivan, Ed. Cham: Springer International Publishing, 2014, pp. 174–189.

[29] M. Gaudioso and P. Legato, “Linear programming models for load balancing,” *Computers & Operations Research*, vol. 18, no. 1, pp. 59–64, 1991.

[30] T. Ichimori, H. Ishii, and T. Nishida, “Optimal sharing,” *Mathematical Programming*, vol. 23, no. 1, pp. 341–348, Dec. 1982.

[31] J. Larusic and A. Punnen, “The balanced traveling salesman problem,” *Computers & OR*, vol. 38, pp. 868–875, May 2011.

[32] S. Kaplan, “Application of programs with maximin objective functions to problems of optimal resource allocation,” *Operations Research*, vol. 22, no. 4, pp. 802–807, August 1974.

[33] K. Ransikarbum and S. J. Mason, “Multiple-objective analysis of integrated relief supply and network restoration in humanitarian logistics operations,” *International Journal of Production Research*, vol. 54, no. 1, pp. 49–68, 2016.

[34] I. S. Orgut *et al.*, “Modeling for the equitable and effective distribution of donated food under capacity constraints,” *IIE Transactions*, vol. 48, no. 3, pp. 252–266, 2016.

[35] H. Luss, “On equitable resource allocation problems: A lexicographic minimax approach,” *Operations Research*, vol. 47, no. 3, pp. 361–378, 1999.

[36] R. M. Salles and J. A. Barria, “Lexicographic maximin optimisation for fair bandwidth allocation in computer networks,” *European Journal of Operational Research*, vol. 185, no. 2, pp. 778–794, 2008.

[37] A. Kumar and J. Kleinberg, “Fairness measures for resource allocation,” *SIAM Journal on Computing*, vol. 36, no. 3, pp. 657–680, Sep. 2006.

[38] D. R. Shier, “A min-max theorem for p-center problems on a tree,” *Transportation Science*, vol. 11, no. 3, pp. 243–252, 1977.

[39] S. Sayin, “A mixed integer programming formulation for the 1-maximin problem,” *Journal of the Operational Research Society*, vol. 51, no. 3, pp. 371–375, 2000.

[40] G. Y. Handler, “Medi-centers of a tree,” *Transportation Science*, vol. 19, no. 3, pp. 246–260, 1985.

[41] E. Erkut and S. Neuman, “Comparison of four models for dispersing facilities,” *INFOR: Information Systems and Operational Research*, vol. 29, no. 2, pp. 68–86, 1991.

[42] F. G. Zhang and E. Melachrinoudis, “The maximin-maxisum network location problem,” *Computational Optimization and Applications*, vol. 19, no. 2, pp. 209–234, Jul. 2001.

[43] M. J. Kuby, “Programming models for facility dispersion: the p-dispersion and maxisum dispersion problems,” *Mathematical and Computer Modelling*, vol. 10, no. 10, p. 792, 1988.

[44] C. Duin and A. Volgenant, “Minimum deviation and balanced optimization: A unified approach,” *Operations Research Letters*, vol. 10, no. 1, pp. 43–48, 1991.

[45] R. E. Burkard and E. Çela, *Linear Assignment Problems and Extensions*. Boston, MA: Springer US, 1999, pp. 75–149.

[46] R. Burkard, M. Dell’Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.

[47] A. K. Chandra and C. K. Wong, “Worst-case analysis of a placement algorithm related to storage allocation,” *SIAM Journal on Computing*, vol. 4, no. 3, pp. 249–263, 1975.

[48] C.-S. Lin *et al.*, “Efficient workload balancing on heterogeneous GPUs using mixed-integer non-linear programming,” *Journal of Applied Research and Technology*, vol. 12, no. 6, pp. 1176–1186, 2014.

[49] R. Lewis, “Constructing wedding seating plans: A tabu subject,” in *Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM’13)*. CSREA Press, 2013, pp. 24–32.

[50] D. Bergman, “An exact algorithm for the quadratic multiknapsack problem with an application to event seating,” *INFORMS Journal on Computing*, vol. 31, no. 3, pp. 477–492, 2019.

[51] A. Hiley and B. A. Julstrom, “The quadratic multiple knapsack problem and three heuristic approaches to it,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’06. New York, NY, USA: ACM, 2006, pp. 547–552.

[52] M. L. Bellows and J. D. L. Peterson. (2012, Feb.) Finding an optimal seating chart. [Online]. Available: https://www.improbable.com/news/2012/Optimal-seating-chart.pdf

[53] R. Lewis and F. Carroll, “Creating seating plans: a practical application,” *Journal of the Operational Research Society*, vol. 67, no. 11, pp. 1353–1362, Nov. 2016.

[54] P. Schaus, “Solving balancing and bin-packing problems with constraint programming,” Ph.D. dissertation, Université catholique de Louvain, 2009.

[55] R. Sadykov and F. Vanderbeck, “Bin packing with conflicts: a generic branch-and-price algorithm,” *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 244–255, 2013.

[56] C. Mullinax and M. Lawley, “Assigning patients to nurses in neonatal intensive care,” *Journal of the Operational Research Society*, vol. 53, no. 1, pp. 25–35, Jan 2002.

[57] C. Castro and S. Manzano, “Variable and value ordering when solving balanced academic curriculum problems,” *Proceedings of 6th Workshop of the ERCIM WG on Constraints (Prague, June 2001)*, Nov. 2001.

[58] P. Olivier, A. Lodi, and G. Pesant, “A comparison of optimization methods for multi-objective constrained bin packing problems,” in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, W.-J. van Hoeve, Ed. Cham: Springer International Publishing, 2018, pp. 462–476.

[59] P. Shaw, “A constraint for bin packing,” in *Principles and Practice of Constraint Programming – CP 2004*, M. Wallace, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 648–662.

[60] S. Gualandi and M. Lombardi, "A simple and effective decomposition for the multidimensional binpacking constraint," in *Principles and Practice of Constraint Programming: 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 356–364.

[61] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, Sep. 1973.

[62] P. Vilím, P. Laborie, and P. Shaw, "Failure-directed search for constraint-based scheduling," in *Integration of AI and OR Techniques in Constraint Programming*, L. Michel, Ed. Cham: Springer International Publishing, 2015, pp. 437–453.

[63] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *Principles and Practice of Constraint Programming — CP98*, M. Maher and J.-F. Puget, Eds. Springer Berlin, 1998, pp. 417–431.

[64] A. Mitsos, B. Chachuat, and P. I. Barton, "McCormick-based relaxations of algorithms," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 573–601, 2009.

[65] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer US, 1998.

[66] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective integer programming: The balanced box method," *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 735–754, 2015.

[67] P. K. Pattanaik, "Social welfare function," in *The New Palgrave Dictionary of Economics*. London: Palgrave Macmillan UK, 2017, pp. 1–7.

[68] A. Sen, "Rawls versus Bentham: An axiomatic examination of the pure distribution problem," *Theory and Decision*, vol. 4, no. 3, pp. 301–309, Feb. 1974.

[69] J. Nash, "Two-person cooperative games," *Econometrica*, vol. 21, no. 1, pp. 128–140, 1953.

[70] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence. Elsevier, 2006, vol. 2.

[71] A. Lodi, "Mixed integer programming computation," in *50 Years of Integer Programming 1958-2008*, M. Jünger *et al.*, Eds. Springer, Berlin, Heidelberg, 2010, pp. 619–645.

[72] C. D'Ambrosio and A. Lodi, "Mixed integer nonlinear programming tools: an updated practical overview," *Annals of Operations Research*, vol. 204, no. 1, pp. 301–320, 2013.

[73] W. Ogryczak, "Inequality measures and equitable approaches to location problems," *European Journal of Operational Research*, vol. 122, no. 2, pp. 374 – 391, 2000.

[74] X. Dong and Y. Cai, "A novel genetic algorithm for large scale colored balanced traveling salesman problem," *Future Generation Computer Systems*, vol. 95, pp. 727–742, 2019.

[75] B. S. Everitt and A. Skrondal, *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010, vol. 4th Edition.

[76] J. Walla, M. Ruthmair, and G. R. Raidl, "Solving a video-server load re-balancing problem by mixed integer programming and hybrid variable neighborhood search," in *Hybrid Metaheuristics*, M. J. Blesa *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 84–99.

[77] M. X. Weng and J. A. Ventura, "A quadratic integer programming method for minimizing the mean squared deviation of completion times," *Operations Research Letters*, vol. 15, no. 4, pp. 205 – 211, 1994.

[78] P. Olivier, A. Lodi, and G. Pesant, "The quadratic multiknapsack problem with conflicts and balance constraints," *INFORMS Journal on Computing*, to appear.

[79] B. Hnich, Z. Kiziltan, and T. Walsh, "Modelling a balanced academic curriculum problem," in *Proceedings of the Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR'02)*, N. Jussien and F. Laburthe, Eds., Le Croisic, France, 2002, pp. 121–131.

[80] B. Hnich *et al.*, "Hybrid modelling for robust solving," *Annals of Operations Research*, vol. 130, no. 1, pp. 19–39, Aug. 2004.

[81] J.-N. Monette *et al.*, "A CP approach to the balanced academic curriculum problem," *Symcon'07, The Seventh International Workshop on Symmetry and Constraint Satisfaction Problems*, Jul. 2007.

[82] M. Chiarandini *et al.*, "The balanced academic curriculum problem revisited," *Journal of Heuristics*, vol. 18, no. 1, pp. 119–148, Feb. 2012.

[83] S. Ceschia, L. Di Gaspero, and A. Schaerf, "The generalized balanced academic curriculum problem with heterogeneous classes," *Annals of Operations Research*, vol. 218, no. 1, pp. 147–163, Jul. 2014.

[84] P. Schaus, P. Van Hentenryck, and J.-C. Régin, "Scalable load balancing in nurse to patient assignment problems," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, W.-J. van Hoeve and J. N. Hooker, Eds. Berlin, Heidelberg: Springer, 2009, pp. 248–262.

[85] G. Pesant, *Balancing Nursing Workload by Constraint Programming*. Cham: Springer International Publishing, 2016, pp. 294–302.

[86] L. Di Gaspero, A. Rendl, and T. Urli, "A hybrid ACO+CP for balancing bicycle sharing systems," in *Hybrid Metaheuristics*, M. J. Blesa *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 198–212.

[87] ——, "Constraint-based approaches for balancing bike sharing systems," in *Principles and Practice of Constraint Programming*, C. Schulte, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 758–773.

[88] M. Rainer-Harbach *et al.*, "Balancing bicycle sharing systems: A variable neighborhood search approach," in *Evolutionary Computation in Combinatorial Optimization*, M. Middendorf and C. Blum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 121–132.

[89] ——, "PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems," *Journal of Global Optimization*, vol. 63, no. 3, pp. 597–629, Nov. 2015.

[90] L. Di Gaspero, A. Rendl, and T. Urli, "Balancing bike sharing systems with constraint programming," *Constraints*, vol. 21, no. 2, pp. 318–348, Apr. 2016.

[91] C. Contardo, C. Morency, and L.-M. Rousseau, "Balancing a dynamic public bike-sharing system," CIRRELT, Tech. Rep., 2012.

[92] D. Chemla, F. Meunier, and R. Wolfler Calvo, "Bike sharing systems: Solving the static rebalancing problem," *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.

[93] T. Raviv, M. Tzur, and I. A. Forma, "Static repositioning in a bike-sharing system: models and solution approaches," *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, Aug. 2013.

[94] J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve, “Inventory rebalancing and vehicle routing in bike sharing systems,” *European Journal of Operational Research*, vol. 257, no. 3, pp. 992–1004, 2017.

[95] H. Hemmati, A. Arcuri, and L. Briand, “Achieving scalable model-based testing through test case diversity,” *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 1, Mar. 2013.

[96] N. K. Toshihide Ibaraki, *Resource Allocation Problems: Algorithmic Approaches*, ser. Foundations of Computing. The MIT Press, 1988, p. 1.

[97] J. L. Heier Stamm *et al.*, “Quantifying and explaining accessibility with application to the 2009 H1N1 vaccination campaign,” *Health Care Management Science*, vol. 20, no. 1, pp. 76–93, Mar. 2017.

[98] B. O. Koopman, “The optimum distribution of effort,” *Journal of the Operations Research Society of America*, vol. 1, no. 2, pp. 52–63, 1953.

[99] O. R. Burt and C. C. Harris, “Letter to the editor—apportionment of the U.S. House of Representatives: A minimum range, integer solution, allocation problem,” *Operations Research*, vol. 11, no. 4, pp. 648–652, 1963.

[100] Z. Zeitlin, “Minimization of maximum absolute deviation in integers,” *Discrete Applied Mathematics*, vol. 3, no. 3, pp. 203 – 220, 1981.

[101] N. Katoh, T. Ibaraki, and H. Mine, “An algorithm for the equipollent resource allocation problem,” *Mathematics of Operations Research*, vol. 10, no. 1, pp. 44–53, 1985.

[102] S. Jacobsen, “On marginal allocation in single constraint min-max problems,” *Management Science*, vol. 17, no. 11, pp. 780–783, 1971.

[103] E. L. Porteus and J. S. Yormark, “More on min-max allocation,” *Management Science*, vol. 18, no. 9, pp. 502–507, 1972.

[104] H. Luss, *Equitable Resource Allocation: Models, Algorithms and Applications*, ser. Information and Communication Technology Series,. Wiley, 2012.

[105] E. Bampis, B. Escoffier, and S. Mladenovic, “Fair resource allocation over time,” in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS ’18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 766–773.

[106] Ogryczak, Wlodzimierz, "Fair optimization – methodological foundations of fairness in network resource allocation," in *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, 2014, pp. 43–48.

[107] W. Ogryczak *et al.*, "Fair optimization and networks: A survey," *Journal of Applied Mathematics*, vol. 2014, p. 612018, Sep. 2014.

[108] O. Eisenhandler and M. Tzur, "The humanitarian pickup and distribution problem," *Operations Research*, vol. 67, no. 1, pp. 10–32, 2019.

[109] L. Brotcorne, G. Laporte, and F. Semet, "Ambulance location and relocation models," *European Journal of Operational Research*, vol. 147, no. 3, pp. 451–463, 2003.

[110] M. Gendreau, G. Laporte, and F. Semet, "A dynamic model and parallel tabu search heuristic for real-time ambulance relocation," *Parallel Computing*, vol. 27, no. 12, pp. 1641–1653, 2001.

[111] F. Margot, "Symmetry in integer linear programming," in *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 647–686.

[112] C. D'Ambrosio *et al.*, "On interval-subgradient and no-good cuts," *Operations Research Letters*, vol. 38, no. 5, pp. 341–345, 2010.

[113] E. Balas and R. Jeroslow, "Canonical cuts on the unit hypercube," *SIAM Journal on Applied Mathematics*, vol. 23, no. 1, pp. 61–69, 1972.

[114] G. Gamrath *et al.*, "The SCIP Optimization Suite 7.0," Zuse Institute Berlin, ZIB-Report 20-10, 2020. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:0297-zib-78023

[115] ——, "The SCIP Optimization Suite 7.0," Optimization Online, Technical Report, 2020. [Online]. Available: https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/7802

[116] S. Demassey, G. Pesant, and L.-M. Rousseau, "A cost-regular based hybrid column generation approach," *Constraints*, vol. 11, no. 4, pp. 315–333, Dec. 2006.

[117] B. Matérn, "Spatial variation – stochastic models and their applications to some problems in forest survey sampling investigations," *Report of the Forest Research Institute of Sweden*, vol. 49, no. 5, pp. 1–144, 1960.

[118] E. Levina and P. Bickel, “The Earth Mover’s distance is the Mallows distance: some insights from statistics,” in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, July 2001, pp. 251–256.

APPENDIX A CHARACTERISTICS OF BALANCE

In this appendix, we provide some formal statistical definitions and basic results [75] of the concepts used in the paper in the attempt of making the reading more self contained.

Definitions

Let

- n be the number of balancing variables $x_1, x_2, \ldots, x_n$,
- $v = \sum_{i=1}^{n} x_i$, the sum of values to be distributed to the balancing variables,
- $\mu = v/n$ be the (fixed) mean,
- d be the deviation allowed for the problem,
- ℓ and u be the (nonnegative) lower and upper bounds of variables $x_1, x_2, \ldots, x_n$.

All of the previously-defined variables are integers, except for the mean which could happen to be fractional. The values of d, ℓ, and u are assumed to be feasible, i.e.,

- $0 \leq \ell \leq \lceil \mu \rceil$ and $u \geq \lceil \mu \rceil$,
- $d \geq \lceil \mu \rceil$ for MINMAX,
- $d \geq 0$ if the mean is integral, or $d \geq 1$ if the mean is fractional, for L_∞-DEVIATION,
- The feasible lower bound of d for L_1- and L_2-DEVIATION needs some explanation due to the complexity introduced by the possibility of having a fractional mean. In such a case, the lowest deviation can be achieved by assigning $\overline{f} = v \bmod n$ variables to a value of $\lceil \mu \rceil$, and $\underline{f} = n - \overline{f}$ variables to a value of $\lfloor \mu \rfloor$. The absolute deviation of a variable which is assigned a value of $\lceil \mu \rceil$ is $\overline{g} = \underline{f}/n$, while for a variable which is assigned a value of $\lfloor \mu \rfloor$ it is $\underline{g} = 1 - \overline{g}$. The feasible lower bound d of L_p-DEVIATION is thus $d = \overline{f} \times \overline{g}^p + \underline{f} \times \underline{g}^p$.

Dispersion

The dispersion characteristic represents the interval within which values of the variables can be found.

For MINMAX, the worst-case dispersion interval is

$$\left[\max\left\{\ell, \lceil \mu - (n-1) \times (\min\{u, d\} - \mu) \rceil\right\}, \min\left\{u, d, \lfloor \mu + (n-1) \times (\mu - \ell) \rfloor\right\}\right].$$

We set all variables except one to the maximum allowed deviation. The remaining variable determines the lower bound of the dispersion interval. That bound cannot be negative, nor lower than ℓ. A similar reasoning applies to the upper bound.

For L_1-DEVIATION, the worst-case dispersion interval is

$$\left[\max\left\{\ell, \left\lceil \mu - \frac{d}{2} \right\rceil\right\}, \min\left\{u, \left\lfloor \mu + \frac{d}{2} \right\rfloor\right\}\right].$$

In the worst case, one variable can account for at most half of the deviation.

For L_2-DEVIATION, the worst-case dispersion interval is

$$\left[\max\left\{\ell, \left\lceil \mu - \sqrt{d \times \frac{n-1}{n}} \right\rceil\right\}, \min\left\{u, \left\lfloor \mu + \sqrt{d \times \frac{n-1}{n}} \right\rfloor\right\}\right].$$

In contrast with L_1-DEVIATION, for L_2-DEVIATION we need to take into account the number of variables in order to tightly bound the dispersion interval.

For L_∞-DEVIATION, the worst-case dispersion interval is

$$\begin{aligned}&\left[\max\left\{\ell, \lceil \mu - d \rceil, \lceil \mu - (\min\left\{u, \lfloor \mu + d \rfloor\right\} - \mu) \times (n-1) \rceil\right\},\right.\\ &\left.\min\left\{u, \lfloor \mu + d \rfloor, \lfloor \mu + (\mu - \max\left\{\ell, \lceil \mu - d \rceil\right\}) \times (n-1) \rfloor\right\}\right].\end{aligned}$$

Here, we have to take into account three cases. First, a simple case of ℓ or u providing the bound. Second, another simple case of the deviation d bounding the interval. Third, a more complicated case with a similar reasoning as for MINMAX (explained previously).

Outliers

As stated, in this paper, we call an outlier any value equal to one of the two extremal values of the dispersion interval (see previous section). In practice, the understanding of the nature of an outlier is more subtle, as it could be any value *far enough* from the mean, for some definition of *far enough*. Let i_{min} and i_{max} be, respectively, the minimum and maximum values of the intervals defined in the previous section.

For MINMAX and L_∞-DEVIATION, the worst-case number of outliers is (let $i_{\text{low}} = \min\{\mu - i_{\text{min}}, i_{\text{max}} - \mu\}$ and $i_{\text{high}} = \max\{\mu - i_{\text{min}}, i_{\text{max}} - \mu\}$)

$$\left\lfloor \frac{n}{1 + \dfrac{i_{\text{high}}}{i_{\text{low}}}} \right\rfloor + \left\lfloor \frac{n}{1 + \dfrac{i_{\text{low}}}{i_{\text{high}}}} \right\rfloor .$$

By looking at the ratio of the deviations between the extremes of the dispersion interval and the mean, we can infer the fractions of variables which will be equal to i_{min} and i_{max}. On the left is the number of outliers which are below the mean, and on the right the number of outliers which are above the mean.

For L_1- and L_2-DEVIATION, things are much more difficult due to the added complexity of managing a (possibly) fractional mean.[1] In fact, we have not been able to devise a closed-form expression for the worst-case number of outliers for these two measures. Such a closed-form expression, *if it exists*, is likely to be overly complicated. We present instead an informal algorithm which achieves the same purpose. For L_1- and L_2-DEVIATION, then, the worst-case number of outliers is[2] computed as

1. Make the variables as balanced as possible. They will either be equal to the mean (if the mean is integral), or be equal to one of the two nearest integers of the mean (if the mean is fractional).

2. While the distribution of values is below the prescribed deviation threshold (also taking into account the potential effects of the actions below), and that at least two non-outlier values can still be found among the variables:

[1] The complexity stems from the fact that an under-mean value and an over-mean value may have distinct fractional parts, and that we do not know beforehand how many values will be under the mean, and how many will be over the mean.

[2] Here, we are assuming that $\mu - \ell \leq u - \mu$. If instead $\mu - \ell > u - \mu$, the logic would be reversed.

   (a) Pick the variable x_i with the lowest value (yet still greater than the value of an outlier), and lower its value by 1.
   (b) If it exists, pick the variable (not x_i and not already an outlier) with the value closest to and lower than $\lfloor \mu \rfloor$. If it does not exist, pick the variable (not x_i and not already an outlier) with the value closest to and greater than $\lceil \mu \rceil$. Increase the value of this variable by 1.

3. Count and return the number of outliers.

This algorithm starts with a balanced distribution of values, and maximizes the number of low-valued outliers (recall that in this particular case, we assume that $\mu - \ell \leq u - \mu$, and as such low-valued outliers are easier to reach than high-valued outliers). When the lowest non-outlier value goes down (step 2a), the highest non-outlier value goes up (step 2b), keeping the sum of values constant. This is done for as long as the deviation threshold allows it.

Smoothness

In order to assess the smoothness of a solution, we compare the distribution of its values to a perfectly smooth distribution, using the Wasserstein distance. This distance is equivalent to the so-called Earth Mover's distance [118]. Given two mounds of earth (in other words, two distributions), this metric represents the effort required to transform one mound of earth into the other. If two distributions are the same, their Wasserstein distance is zero. As the differences between two distributions increase, so does their Wasserstein distance.

APPENDIX B BACP MODEL

This BACP model, written in a logical form, uses a similar notation as [81]. Let

- $\mathcal{C} = \{1, \ldots, n\}$ be the index set of courses,
- $\mathcal{P} = \{1, \ldots, m\}$ be the index set of periods,
- w_i denote the load of course i with $w = \sum_{i \in \mathcal{C}} w_i$ representing the combined loads of all the courses,
- $\mathcal{Q} \subset \mathcal{C} \times \mathcal{C}$ denote the set of prerequisites, where an element (i, j) indicates that course i is a prerequisite to course j,
- $L = \{L_1, \ldots, L_m\}$ denote the loads of the periods for an assignment,
- $P_i \in \mathcal{P}$ denote the period course i is assigned to,
- B_{ik} denote if course i is assigned to period k.

The model is defined by

$$P_i < P_j, \qquad \forall (i, j) \in \mathcal{Q} \tag{B.1}$$

$$(P_i = k) \Leftrightarrow (B_{ik} = 1), \qquad \forall i \in \mathcal{C}, k \in \mathcal{P} \tag{B.2}$$

$$L_k = \sum_{i \in \mathcal{C}} B_{ik} w_i, \qquad \forall k \in \mathcal{P} \tag{B.3}$$

$$B_{ik} \in \{0, 1\}, \qquad \forall i \in \mathcal{C}, k \in \mathcal{P} \tag{B.4}$$

and its objectives are

$$\max_{k \in \mathcal{P}} L_k \tag{\textsc{minmax}}$$

$$\sum_{k \in \mathcal{P}} |L_k - w/m| \tag{L_1-\textsc{deviation}}$$

$$\sum_{k \in \mathcal{P}} (L_k - w/m)^2 \tag{L_2-\textsc{deviation}}$$

$$\max_{k \in \mathcal{P}} |L_k - w/m| \,. \qquad (L_\infty\text{-DEVIATION})$$

Constraints (B.1) ensure that course prerequisite requirements are met, and period loads L are tracked with the help of auxiliary variables B (B.2)–(B.4).

APPENDIX C NPAP MODEL

This NPAP model, written in a logical form, uses a similar notation as [85].[1] Let

- $\mathcal{N} = \{1, \ldots, n\}$ be the index set of nurses,
- $\mathcal{P} = \{1, \ldots, m\}$ be the index set of patients,
- a_i denote the acuity of patient i with $a = \sum_{i \in \mathcal{P}} a_i$ representing the combined acuities of all the patients,
- $p_{\min}$ and $p_{\max}$ denote the minimum and maximum number of patients that can be assigned to a nurse,
- n_i denote the nurse to which patient i is assigned,
- w_j denote the workload of nurse j,
- t_{ij} denote if patient i is assigned to nurse j.

The model is defined by

$$(n_i = j) \Leftrightarrow (t_{ij} = 1), \qquad \forall i \in \mathcal{P}, j \in \mathcal{N} \tag{C.1}$$

$$t_{ij} \in \{0, 1\}, \qquad \forall i \in \mathcal{P}, j \in \mathcal{N} \tag{C.2}$$

$$w_j = \sum_{i \in \mathcal{P}} a_i t_{ij}, \qquad \forall j \in \mathcal{N} \tag{C.3}$$

$$p_{\min} \leq \sum_{i \in \mathcal{P}} t_{ij} \leq p_{\max}, \qquad \forall j \in \mathcal{N} \tag{C.4}$$

and its objectives are

$$\max_{j \in \mathcal{N}} w_j \tag{MINMAX}$$

$$\sum_{j \in \mathcal{N}} \left| w_j - a/m \right| \tag{L_1-DEVIATION}$$

[1] In the interest of simplicity, we have left aside the staffing part of the NPAP as it is not particularly meaningful for our purposes. The reader may refer to the cited paper for details.

$$\sum_{j\in\mathcal{N}} \left(w_j - a/m\right)^2 \tag{L_2-DEVIATION}$$

$$\max_{j\in\mathcal{N}} \left|w_j - a/m\right|. \tag{L_∞-DEVIATION}$$

Auxiliary variables t (C.1)–(C.2) are used to track nurse workloads w (C.3), as well as to ensure the nurses are assigned a proper number of patients (C.4).

APPENDIX D BBSS MODEL

We present the CP-based BBSS model of [86], using that notation (an example of a non-CP formulation can be found in [89]). Let

- $\mathcal{S} = \{1, \ldots, S\}$ be the set of S stations,
- $\mathcal{D} = \{S+1, \ldots, S+D\}$ be the set of D depots,
- for a station $s \in \mathcal{S}$, $C_s > 0$ be its capacity, b_s its initial number of bikes, and t_s its target number of bikes,
- $\mathcal{V} = \{1, \ldots, V\}$ be the set of V vehicles,
- for a vehicle $v \in \mathcal{V}$, $c_v > 0$ be its capacity, $\widehat{b}_v \geq 0$ its initial load, and $\widehat{t}_v > 0$ its available time,
- travel time matrix tt_{uv} with $u, v \in \mathcal{S} \cup \mathcal{D}$ (which includes the processing time of serving a station).

This model requires the stations and depots to be grouped into an ordered set of nodes. The depots are duplicated as we need distinct starting and ending nodes, and a dummy vehicle (with an associated depot) is introduced. The nodes $\mathcal{U} = \mathcal{V}_s \cup \mathcal{S} \cup \mathcal{V}_e = \{0, \ldots, V, V+1, \ldots, V+S, V+S+1, \ldots, 2V+S+2\}$ begin with the starting depots $\mathcal{V}_s$ (including that of the dummy vehicle), then the stations $\mathcal{S}$, and finally the ending depots $\mathcal{V}_e$ (again including that of the dummy vehicle).

This formulation makes use of a successor-predecessor dynamic. Several auxiliary variables are required for this purpose

- $succ_i \in \mathcal{U}$ denotes the successor of node $i \in \mathcal{U}$,
- $pred_i \in \mathcal{U}$ denotes the predecessor of node $i \in \mathcal{U}$,
- $vehicle_i \in \mathcal{V}$ denotes the vehicle serving node $i \in \mathcal{U}$,
- $service_i \in \{-b_i, \ldots, C_i - b_i\}$ denotes the bike delta of node $i \in \mathcal{U}$ after being served,
- $load_i \in \{0, \ldots, c_v\}$ denotes the load of vehicle $v \in \mathcal{V}$ after serving node $i \in \mathcal{U}$,

- $time_i \in \{0, \ldots, \widehat{t}_v\}$ denotes the time at which vehicle $v \in \mathcal{V}$ arrives at node $i \in \mathcal{U}$.

The model is defined by

$$\texttt{alldifferent}(succ) \tag{D.1}$$

$$\texttt{alldifferent}(pred) \tag{D.2}$$

$$pred_{succ_s} = s, \qquad \forall s \in \mathcal{S} \tag{D.3}$$

$$succ_{pred_s} = s, \qquad \forall s \in \mathcal{S} \tag{D.4}$$

$$pred_v = V + S + v, \qquad \forall v \in \mathcal{V}_s \tag{D.5}$$

$$succ_{V+S+v} = v, \qquad \forall v \in \mathcal{V}_s \tag{D.6}$$

$$pred_i \neq i, \qquad \forall i \in \mathcal{U} \tag{D.7}$$

$$succ_i \neq i, \qquad \forall i \in \mathcal{U} \tag{D.8}$$

$$vehicle_v = v, \qquad \forall v \in \mathcal{V}_s \tag{D.9}$$

$$vehicle_{V+S+v} = v, \qquad \forall v \in \mathcal{V}_s \tag{D.10}$$

$$vehicle_{succ_i} = vehicle_i, \qquad \forall i \in \mathcal{U} \tag{D.11}$$

$$vehicle_{pred_i} = vehicle_i, \qquad \forall i \in \mathcal{U} \tag{D.12}$$

$$load_v = \widehat{b}_v, \qquad \forall v \in \mathcal{V}_s \setminus \{V\} \tag{D.13}$$

$$loadV = 0 \tag{D.14}$$

$$load_{succ_i} = load_i - service_i, \qquad \forall i \in \mathcal{U} \tag{D.15}$$

$$load_v = 0, \qquad \forall v \in \mathcal{V}_e \tag{D.16}$$

$$(vehicle_s \neq V) \Leftrightarrow (service_s \neq 0), \qquad \forall s \in \mathcal{S} \tag{D.17}$$

$$load_s \leq c_{vehicle_s}, \qquad \forall s \in \mathcal{S} \tag{D.18}$$

$$service_s \leq 0, \qquad \forall s \in \mathcal{S} : b_s > t_s \tag{D.19}$$

$$service_s \geq 0, \qquad \forall s \in \mathcal{S} : b_s < t_s \tag{D.20}$$

$$service_i = 0, \qquad \forall i \in \mathcal{V}_s \cup \mathcal{V}_e \tag{D.21}$$

$$b_s + service_s \leq C_s, \qquad \forall s \in \mathcal{S} \tag{D.22}$$

$$b_s + service_s \geq 0, \qquad \forall s \in \mathcal{S} \tag{D.23}$$

$$time_v = 0, \qquad \forall v \in \mathcal{V}_s \tag{D.24}$$

$$time_v = time_{pred_v} + tt_{pred_v,v}, \qquad \forall v \in S \cup \mathcal{V}_e \tag{D.25}$$

$$time_{succ_v} = time_v + tt_{v,succ_v}, \qquad \forall v \in \mathcal{V}_s \cup S \tag{D.26}$$

$$time_{V+S+v} \leq \widehat{t}_v, \qquad \forall v \in \mathcal{V}. \tag{D.27}$$

All values of the successors and predecessors are distinct (D.1)–(D.2); these constraints, coupled with the time constraints of the vehicles, ensure the absence of subtours. The successor-predecessor chain must be consistent (D.3)–(D.6), and loops are forbidden (D.7)–(D.8). Depots are assigned to the vehicles (D.9)–(D.10), and the vehicle chain must be consistent (D.11)–(D.12). The initial loads of the vehicles are set (D.13)–(D.14), the load chain must be consistent (D.15), and the vehicles must be empty at the end of their routes (D.16). A station receiving no service will be visited by the dummy vehicle (D.17), and the loads of the other vehicles must not exceed their capacities (D.18). Stations visited by a vehicle must see their bike counts altered in some way (D.19)–(D.20), while depots remain unserved (D.21). Stations cannot be served in excess of their capacities (D.22)–(D.23). The routes start at the depots with times of zero (D.24), the time chain must be consistent (D.25)–(D.26), and the durations of the routes must remain within the specified limits (D.27). The objectives are defined by

$$\max_{s\in\mathcal{S}}(b_s + service_s - t_s) \tag{MINMAX}$$

$$\sum_{s\in\mathcal{S}} |b_s + service_s - t_s| \tag{L_1-DEVIATION}$$

$$\sum_{s\in\mathcal{S}} (b_s + service_s - t_s)^2 \tag{L_2-DEVIATION}$$

$$\max_{s\in\mathcal{S}} |b_s + service_s - t_s| \,. \tag{L_∞-DEVIATION}$$