Universidade Estadual de Campinas
Instituto de Computação

Yulle Glebbyo Felipe Borges

Exact and Heuristic Algorithms for Packing Problems: Color Alternation, Uncertainty, and in Smart Grids

Algoritmos Exatos e Heurísticos para Problemas de Empacotamento: Alternância de Cores, Incerteza e em Smart Grids

CAMPINAS
2023

Yulle Glebbyo Felipe Borges

Exact and Heuristic Algorithms for Packing Problems: Color Alternation, Uncertainty, and in Smart Grids

Algoritmos Exatos e Heurísticos para Problemas de Empacotamento: Alternância de Cores, Incerteza e em Smart Grids

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Rafael Crivellari Saliba Schouery
Co-supervisor/Coorientador: Prof. Dr. Flávio Keidi Miyazawa

Este exemplar corresponde à versão final da Tese defendida por Yulle Glebbyo Felipe Borges e orientada pelo Prof. Dr. Rafael Crivellari Saliba Schouery.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Silvania Renata de Jesus Ribeiro - CRB 8/6592

Borges, Yulle Glebbyo Felipe, 1992-
B644e Exact and heuristic algorithms for packing problems : color alternation, uncertainty, and in smart grids / Yulle Glebbyo Felipe Borges. – Campinas, SP : [s.n.], 2023.

Orientador: Rafael Crivellari Saliba Schouery.
Coorientador: Flávio Keidi Miyazawa.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Otimização combinatória. 2. Problemas de empacotamento. 3. Programação linear inteira. 4. Heurística (Computação). I. Schouery, Rafael Crivellari Saliba, 1986-. II. Miyazawa, Flávio Keidi, 1970-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações Complementares

Título em outro idioma: Algoritmos exatos e heurísticos para problemas de empacotamento : alternância de cores, incerteza e em smart grids
Palavras-chave em inglês:
Combinatorial optimization
Packing problems
Integer linear programming
Heuristic (Computer Science)
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Rafael Crivellari Saliba Schouery [Orientador]
Reinaldo Morabito Neto
Edilson Fernandes de Arruda
Kelly Cristina Poldi
Mário César San Felice
Data de defesa: 04-07-2023
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-1865-4104
- Currículo Lattes do autor: http://lattes.cnpq.br/5053937017978886

Universidade Estadual de Campinas
Instituto de Computação

Yulle Glebbyo Felipe Borges

Exact and Heuristic Algorithms for Packing Problems: Color Alternation, Uncertainty, and in Smart Grids

Algoritmos Exatos e Heurísticos para Problemas de Empacotamento: Alternância de Cores, Incerteza e em Smart Grids

Banca Examinadora:

- Prof. Dr. Rafael Crivellari Saliba Schouery
  IC/UNICAMP
- Prof. Dr. Reinaldo Morabito Neto
  CCET/UFSCar
- Prof. Dr. Edilson Fernandes de Arruda
  University of Southampton
- Profa. Dra. Kelly Cristina Poldi
  IMECC/UNICAMP
- Prof. Dr. Mário César San Felice
  DC/UFSCar

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 04 de julho de 2023

Acknowledgments

In moments of introspection while composing this thesis, I've realized that this journey involves more than research—it's about growth, resilience, and discovery. None of this could have been achieved without the support of those closest to me.

I've been fortunate to be born into a family that values education. Their commitment allowed me to dedicate myself fully to my studies from the beginning. They've stood by me through every decision, even when clarity eluded us. Their presence has been my bedrock, especially during challenging times. For this, and much more, my heartfelt gratitude extends to my mother, Flávia, my father, José, and my brothers José Jr. and Mathews.

To Prof. Rafael and Prof. Flávio, your role goes beyond that of mentors; you've served as guiding beacons through the research labyrinth. Your belief in me has empowered me to find belief in myself. You represent what I aspire to become, both as a researcher and a person. My sincerest gratitude is directed towards you both.

A nod of appreciation goes out to my colleagues at the Institute of Computing for the stimulating discussions and collaborative atmosphere. To my peers within the Laboratory of Optimization and Combinatorics, you've transformed these years into an enjoyable journey, free from any sense of intimidation. Those coffee room debates and seemingly endless lunch breaks at the university restaurant are etched in my memory. Working beside you has been an uplifting and gratifying experience.

Not to be forgotten, my friends have been steady companions, each step of the way. Your companionship infused my journey with motivation during its most difficult moments. Special mention goes to my girlfriend, Ingrid, whose understanding and unwavering support have been invaluable, particularly during the final months of writing this thesis.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001 and the Brazilian National Council for Scientific and Technological Development (CNPq), grant #144257/2019–0.

While words may fall short of expressing the extent of my gratitude, I hope this acknowledgment conveys the depth of my appreciation.

Resumo

O problema de empacotamento unidimensional (BPP, do inglês *Bin Packing Problem*) consiste em alocar um conjunto de items unidimensionais, cada um com um dado comprimento, na menor quantidade possível de recipientes unidimensionais de uma dada capacidade. Nesta tese, desenvolvemos algoritmos exatos e heurísticas para as seguintes variantes e generalizações do BPP: o problema de empacotamento colorido, no qual cada item também possui uma cor e, em cada recipiente, os itens são empacotados em uma ordem de forma que as cores são alternadas; o problema de empacotamento com cenários, no qual consideramos a presença de cenários incertos e queremos minimizar o número de recipientes no pior cenário; e problemas de alocação de energia restritos a uma limitação de consumo para um esquema de leilão de energia renovável em *smart grids*. Embora estes problemas sejam NP-difíceis, eles possuem motivações práticas que nos encorajam a abordá-los com modelos matemáticos e heurísticas. Propomos quatro modelos matemáticos para o problema de empacotamento colorido. Dois deles são baseados na formulação pseudo-polinomial de fluxo em arcos de Valério de Carvalho, e os outros dois são modelos exponenciais de particionamento de conjuntos, resolvidos por um algoritmo de *branch-cut-and-price* usando o *VRPSolver*. Propomos vários algoritmos para o problema de empacotamento com cenários: dois algoritmos de aproximação, um algoritmo exato de *branch-and-price* com *dual feasible functions*, e uma meta-heurística de busca em vizinhança variável. Por fim, propomos um leilão de energia renovável em *smart grids* no qual os sub-problemas de alocação de energia são modelados como diferentes variantes de um problema de empacotamento multidimensional. Especificamente, consideramos dois problemas. No primeiro, um consumidor deve calcular seu lance para um dado plano de uso de energia de acordo com suas preferências. No segundo, o provedor deve acomodar planos de uso de energia dos consumidores sem ultrapassar a capacidade de geração de energia da rede. Ambos os problemas são resolvidos com um modelo de programação linear inteira mista.

Abstract

The bin packing problem (BPP) consists of assigning a set of one-dimensional items, each with a given length, to the smallest possible number of one-dimensional bins of a given capacity. In this thesis, we develop exact algorithms and heuristics for the following variants and generalizations of the BPP: the colored bin packing problem, in which each item also has a color and, in each bin, the items are packed in an order where the colors alternate; the bin packing problem with scenarios, in which we consider the presence of uncertain scenarios and want to minimize the number of bins in the worst scenario; and energy allocation problems constrained by consumption limiting in an auction scheme for renewable energy in smart grids. Although these problems are NP-hard, they have practical motivations that encourage us to approach them with mathematical models and heuristics. We propose four mathematical models for the colored bin packing problem. Two of them are based on the pseudo-polynomial arc flow formulation of Valério de Carvalho and the other two are exponential set-partition models, solved by a branch-cut-and-price algorithm using the VRPSolver framework. We propose several algorithms for the bin packing problem with scenarios: two approximation algorithms, one exact branch-and-price algorithm using dual feasible functions, and a Variable Neighborhood Search metaheuristic. Finally, we propose an auction for renewable energy in smart grids in which energy allocation sub-problems are modeled as different variants of a multidimensional packing problem. Specifically, we consider two problems. In the first one, a consumer must compute their bid for a given energy usage plan according to their preferences. In the second one, the provider must accommodate consumers' energy usage plans without exceeding the energy generation capacity. Both problems are solved with a mixed-integer linear programming model.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Operations Research (OR) is an interdisciplinary field that aims to achieve better decision-making by applying scientific principles to a variety of problems, providing a quantitative basis for complex decisions [34]. Such problems are often characterized as Combinatorial Optimization (CO) problems, that is, one wants to find a solution that either minimizes or maximizes a given function over a discrete, but usually very large, set of possible solutions. Although simple in their descriptions, these problems often require sophisticated computational approaches in order to find good solutions in a reasonable time [35].

Among these problems, consider the situation in which a producer must decide how to cut stocks from a certain material into the smaller pieces necessary to manufacture their final product while using the least amount of stock possible. Or a transporter who must find the smallest possible amount of trucks needed to accommodate all boxes that they must ship. These are some examples of cutting and packing (C&P) problems, a class that includes some of the oldest CO problems investigated under an OR perspective [39].

C&P problems usually have the following common structure: a set (or subset) of small objects has to be assigned to a set (or subset) of large objects such that geometric conditions, such as non-overlapping items and large objects' boundaries, are satisfied. Such situations arise in numerous real-life applications, e.g. flat-glass [37], plastic [45], and paper [25] industries, cloud computing [1], and logistics and transportation [38, 42, 36, 21]. Due to timing constraints and scalability concerns, it is important to develop tailor-made algorithms that find optimal, or near optimal, solutions for these problems in a reasonable computing time.

The main C&P problems of interest in this thesis are the one-dimensional bin packing problem (BPP) and the one-dimensional cutting stock problem (CSP). In the BPP, we are given a set of items, each with their respective lengths, and we have to pack them inside the least possible amount of bins of a given maximum length. In the CSP, we are given a set of material pieces of a given length and have to cut them into smaller pieces to satisfy the demand for each smaller piece, while minimizing the material used. The main differences between these problems come from the practical applications of where they arise. In their typology of C&P problems, Wäscher et al. [47] state that bin packing problems often present items that are strongly heterogeneous, that is there are only a few items of the same length. On the other hand, cutting stock problems are usually associated with a weakly heterogeneous assortment of items. Both problems are NP-hard, that is, it

is unlikely to exist an efficient algorithm that is guaranteed to find an optimal solution to them. Hence, we have to find more creative ways to tackle such problems, compromising some solution quality in exchange for computational efficiency.

There exist many variants of the BPP and CSP. Any algorithmic results for these problems have the potential to be further developed to the specificities of other variants. Since some variants often share characteristics, results for a specific variant may unlock new research opportunities for seemingly unrelated problems. However, C&P problems are diverse in terms of their difficulty, so deciding which algorithmic approach works best for each problem is a part of the research challenge [47].

This area has been used as a training ground for novel techniques in mixed-integer linear programming and approximation algorithms. New developments in algorithmic design have been introduced in other areas, but just recently have started making their way to these problems, such as branch-cut-and-price algorithms and adaptive hybrid metaheuristics. All these reasons motivate the development of effective algorithms for C&P problems, as explored in this thesis.

Each of the subsequent chapters corresponds to a fairly self-contained article, either published or submitted to international journals in the field. All chapters present the concepts necessary to contextualize their results. However, to help guide the unfamiliar reader, we briefly introduce some common concepts throughout the thesis in Section 1.1. In Section 1.2 we state the contributions of this work. Finally, in Section 1.3 we describe the organization for the remainder of the text.

1.1 Background

In this section, we provide some useful concepts that are common throughout the thesis. These may help guide the unfamiliar reader into the later chapters. We also provide references that cover each theme in detail for the interested reader.

1.1.1 Cutting and Packing Problems

Cutting and Packing is a class of combinatorial optimization problems that contains all problems included in this thesis. It is mainly composed of two types of problems: *cutting*, in which some material resource must be divided into smaller demanded pieces; and *packing*, in which demanded items must be allocated into space-limited containers. Interestingly enough, these types of problems are strongly correlated in a way that solution methods for one type can also be applied to the other. The book by Scheithauer [39] offers a good introduction to the problems, modeling, and solution methods for C&P problems.

In this section, we formally introduce the main C&P problems, contextualizing them with the content of this thesis. We focus on one-dimensional C&P problems, so all problems discussed throughout the thesis are assumed to be one-dimensional unless explicitly stated otherwise. Because of this, we omit the 1D prefix that is often used in the literature do specify that a problem is one-dimensional.

Knapsack Problem (KP) In the KP, we are given a knapsack of capacity $L \in \mathbb{Q}_+$, and a set of items $I = \{1, \ldots, n\}$, each $i \in I$ having a length $l_i \in \mathbb{Q}_+$ and a value $v_i \in \mathbb{Q}_+$. A solution S is a subset of I such that $\sum_{i \in S} l_i \leq L$. The objective is to find a solution that maximizes the sum of the values of the items in it.

This is a classic CO problem that often appears as a substructure in larger problems. Although the KP is a NP-hard problem, there exists fairly efficient, although super-polynomial, algorithms for it. The book by Kellerer et al. [27] provides details on the main algorithms for the KP as well as some of its variants.

Cutting Stock Problem (CSP) In the CSP, we are given an infinite number of identical stock rolls of length $L \in \mathbb{Q}_+$, and a set of demanded items $I = \{1, \ldots, n\}$, each $i \in I$ with a different length $l_i \in \mathbb{Q}_+$ and demand $d_i \in \mathbb{Z}_+$. A solution $S = (S_1, \ldots, S_m)$ is an attribution of at least d_i copies of each $i \in I$ into m stock rolls, such that $\sum_{(i,x_{ij}) \in S_j} x_{ij} l_i \leq L$ for all $j \in \{1, \ldots, m\}$, where x_{ij} corresponds to the amount of copies of item i that are allocated to subset j. The objective is to find a solution that uses the least possible amount m of stock rolls.

Bin Packing Problem (BPP) In the BPP, we are given an infinite number of identical bins with capacity $L \in \mathbb{Q}_+$, and a set of items $I = \{1, \ldots, n\}$, each $i \in I$ with length $l_i \in \mathbb{Q}_+$. A solution $S = \{S_1, \ldots, S_m\}$ is a partition of I such that $\sum_{i \in S_j} l_i \leq L$ for all $j \in \{1, \ldots, m\}$. The objective is to find a solution with the least possible amount m of parts.

Although they focus on different applications, the one-dimensional CSP can be viewed as a generalization of the BPP. Given an instance of the BPP, we can create an instance of the CSP by grouping together all items of the same length and turning them into a single item of the CSP with demand equal to the cardinality of the set of grouped items. This way, algorithms for one problem can usually be adapted to the other. In Figure 1.1 we provide an example of an instance of the CSP with 3 items and bin capacity 7. Item lengths are given as $\{5, 3, 2\}$ and demands $\{2, 1, 3\}$. The figure also shows an optimal solution for the proposed instance.

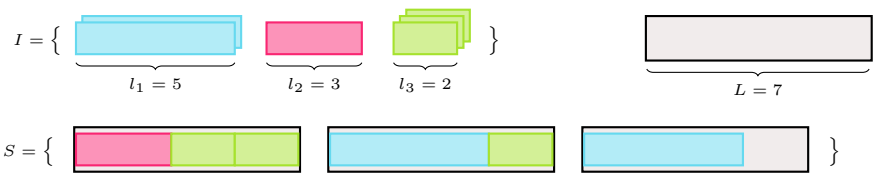

Figure 1.1: Simple instance for the CSP with 3 items and one optimal solution

1.1.2 Combinatorial Optimization Approaches

In this section, we give a brief introduction to all the CO approaches considered throughout the thesis.

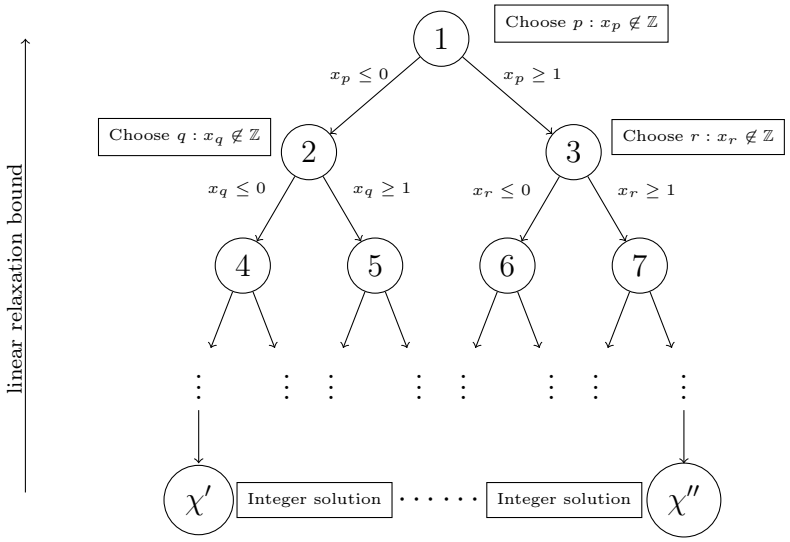

Figure 1.2: Branch-and-bound tree example.

Exact Algorithms

With exact algorithms, we aim to develop algorithms that find proven optimal solutions by exploring the problem structure and (implicitly) enumerating its search space.

The main strategy considered for exact algorithms is *Mixed-Integer Linear Programming* (MILP). In MILP, we represent the problem as *Linear Program* (LP) in which some, or even all, of the decision variables, must take an integer value. To solve it, a *branch-and-bound* algorithm is used to implicitly explore the whole solution space, pruning non-improving solutions with the use of model-based bounds. This is achieved by starting, at the root node, with the linear program (with all integrality constraints relaxed), which is solved with any linear-programming solver (for instance the Simplex algorithm [7]). After solving the LP, if none of the originally-integer variables has a non-integer value, then that tree is solved. Otherwise, one of these variables is chosen to be branched on by creating new nodes on which linear constraints are imposed to make the current solution infeasible. To illustrate, if the chosen variable x_p has a fractional value between 0 and 1, then we may create two nodes: one forcing that variable to assume a value smaller than or equal to zero, $x_p \leq 0$; and the other forcing that variable to assume a value greater than or equal to one, $x_p \geq 1$. This branching process is repeated recursively on every node. Figure 1.2 illustrates this process. If any particular node results in an infeasible LP, or if it can only result in worse solutions than the current incumbent, the whole subtree under that node is pruned-off.

The MILP strategy takes advantage of the fact that LP can be solved efficiently, and since the nodes are very similar, the structure of previous nodes can be used to speed up the search. Furthermore, lower and upper bounds can be computed at different points of the exploration process in order to increase the pruning capabilities of the algorithms. This strategy also makes it possible to integrate large linear programs through column generation, resulting in *branch-and-price* methods, or even cut generation, resulting in *branch-and-cut* methods. In fact, it is possible to combine these two techniques in order to achieve very large models with strong bounds, the so-called *branch-cut-and-price* (BCP)

methods.

In the following, we briefly discuss the two main models for the BPP and CSP. Although these models are very different in concept and in regard to the number of variables, they both present the same, very strong, linear relaxation upper bound [43].

The first MILP we discuss is a **set-cover-based branch-and-price** model based on the column generation work of Gilmore and Gomory [17] and later implemented using branch-and-price by Vance et al. [44]. We call a subset P of items such that the sum of its lengths is at most L a *packing pattern*. Now, let $\mathcal{P}$ be the family of all packing patterns, and $\mathcal{P}_i$ be the set of patterns that include any item $i \in I$. Notice that a feasible solution for the BPP can be represented as a set of packing patterns such that each item in I appears in at least one of the patterns. Thus, we can formulate the BPP as the following MILP in which variable x_P indicates whether a packing pattern $P \in \mathcal{P}$ is in the solution or not:

$$\begin{aligned}
\text{(P)} \quad & \text{minimize} \sum_{P \in \mathcal{S}} x_P \\
& \text{subject to} \sum_{P \in \mathcal{P}_i} x_P \geq 1 \qquad \forall\, i \in I \qquad (1.1) \\
& \qquad\qquad x_P \in \{0,1\} \qquad \forall\, P \in \mathcal{P}, \qquad (1.2)
\end{aligned}$$

where Constraint (1.1) makes sure that each item appears in at least one packing pattern, and Constraints (1.2) defines the domain of the integer variables.

We can also write the dual (D) of the linear relaxation of (P) as follows:

$$\begin{aligned}
\text{(D)} \quad & \text{maximize} \sum_{i \in I} \nu_i \\
& \text{subject to} \sum_{i \in P} \nu_i \leq 1 \qquad \forall\, P \in \mathcal{P} \\
& \qquad\qquad \nu_i \geq 0 \qquad \forall\, i \in I.
\end{aligned}$$

Since the number of packing patterns can be exponential, then so can the number of variables in (P) and the number of constraints in (D). However, we can solve (P) using a branch-and-bound method using only a few initial packing patterns, then generating new patterns through the process of *pricing* in each node until an optimal solution for the linear relaxation of (P) is found. In order to generate a new variable to be added to (P), we must find a constraint that is violated in (D), that is, given a vector ν find a packing pattern P such that $\sum_{i \in P} \nu_i > 1$. This is equivalent to finding a solution to the Knapsack Problem (in which each item $i \in I$ has length l_i, value ν_i, and the size of the knapsack is L) with a total value greater than 1. If no such pattern exists, then the current set of packing patterns is sufficient to find an optimal solution for the relaxation of (P). Unfortunately though, depending on how branching is implemented, pricing problems in lower nodes of the tree might differ from the classical KP and, instead, include new additional constraints, which might compromise the performance of the algorithm. For instance, if a branching constraint imposes that a variable $x_P \leq 0$ for some pattern $P \in \mathcal{P}$, then we must guarantee that another pattern identical to P will not be generated during

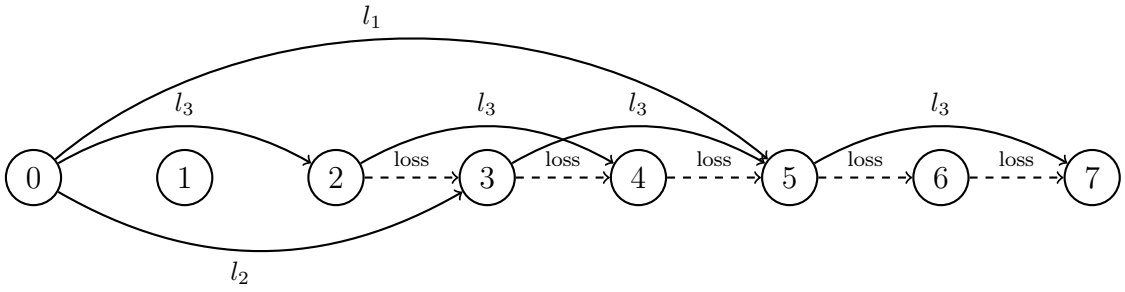

Figure 1.3: Acyclic digraph corresponding to a simple instance of the CSP with 3 items and stock roll length 7, adapted from [5].

pricing, resulting in a KP with forbidden patterns as pricing subproblem.

The other MILP discussed in this section is the CSP **arc flow formulation** of Valério De Carvalho [43]. In this model, we assume that the lengths of the demanded items and stock rolls are integer values, for simplicity, since in practice it is always possible to scale rational lengths into large integers. This way, we can represent the set of items that can be cut from a single stock roll as the problem of finding a path in a specific acyclic digraph with $L + 1$ vertices. Let this graph be $G(V, A)$ with $V = \{0, 1, \ldots, L\}$ and $A_{item} = \{(i, j) : 0 < j \leq L, \exists u \in I : j - i = l_u\}$, that is, there is an arc connecting two vertices (i, j) if there exists an item in I of size $j - i$. For completeness' sake, we must also consider loss arcs $A_{loss} = \{(i, i + 1) : l_{min} < i \leq L - 1\}$, where l_{min} is the length of the smallest item in the instance, to represent wastes. This way, we consider the set of arcs $A = A_{items} \cup A_{loss}$. Notice that the number of arcs in this graph is $O(nL)$. Considering this graph, the cutting of any single stock roll can be represented by a path from 0 to L, with each item arc used in this path representing the cutting of an item in the position designated by the endpoints of the arc, and each loss arc used represents wastes in those positions. It is also possible to reduce symmetries in this model by using the following rule proposed by Valério De Carvalho [43]: Consider only solutions in which items appear in a fixed non-decreasing order of lengths, consequently reducing the number of arcs necessary to represent such solutions.

To illustrate, consider the following example: Given an instance of the CSP with $L = 7$ and 3 items with lengths $5, 3, 2$ and demands $3, 1, 2$, respectively. We construct the acyclic digraph shown in Figure 1.3. All cutting patterns necessary to represent an optimal solution to the instance in the example can be represented as paths from 0 to L in the graph of Figure 1.3.

This way, we can formulate the CSP as the problem of determining the minimum flow departing from 0 and arriving at L, added of side constraints that enforce that the amount of flow passing through the arcs that correspond to an item $u \in I$ is at least d_u. We can use a feedback arc connecting vertices L and 0 to count the number of paths used in the solution. Thus, considering one decision variable x_{ij} for each arc $(i, j) \in A$ and one variable z for the feedback arc, we can model the CSP as the following MILP:

$$
\begin{aligned}
&\text{minimize } z \\
&\text{subject to} \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = \begin{cases} -z & \text{for } j = 0, \\ 0 & \text{for } j = 1, \ldots, L-1, \\ z & \text{for } j = L, \end{cases} && (1.3) \\
&\qquad \sum_{(i,j)\in A: j = i + l_u} x_{ij} \geq d_u \qquad \forall u \in I && (1.4) \\
&\qquad x_{ij} \in \mathbb{Z}_+ \qquad \forall\, (i,j) \in A, \\
&\qquad z \in \mathbb{Z}_+.
\end{aligned}
$$

in which Constraints (1.3) guarantee flow conservation, that is the flow entering in a vertex must be equal to flow emanating from the same vertex for all intermediate vertices. Constraints (1.4) are the demand constraints.

Delorme et al. [11] provide a survey detailing several mathematical models for bin packing and cutting stock problems as well as computational experiments comparing them. For further information on mixed linear integer programming, we refer to the books of Schrijver [40] and Wolsey [49].

Approximation Algorithms

An approximation algorithm for an optimization problem is an algorithm that, for any given instance, has its execution time growing polynomially with the size of the input, and produces a feasible solution with objective value within some guaranteed factor of the optimal value. Although exact algorithms find guaranteed optimal solutions, when it comes to NP-hard problems, those algorithms usually take exponential time, making it difficult to use them for solving practical large instances of optimization problems. In contrast, approximation algorithms are more versatile, compromising some solution quality in exchange for guaranteed polynomial run-time.

Given a minimization problem, an algorithm A has an approximation factor of α if, given an instance I, it produces a solution of objective $A(I)$ that is at most α times greater than the objective value of an optimal solution of I, $OPT(I)$. Such algorithms are also referred to as α-approximations. In other words, an α-approximation algorithm A produces a solution such that

$$A(I) \leq \alpha OPT(I).$$

A *Polynomial-Time Approximation Scheme* (PTAS) is a family of approximation algorithms $\{A_\varepsilon\}$ that, for any fixed $\varepsilon > 0$, produces a solution such that

$$A_\varepsilon(I) \leq (1 + \varepsilon) OPT(I).$$

In this case, the execution time of each algorithm is required to grow polynomially with the size of the input instance, but may grow exponentially with $1/\varepsilon$. When a PTAS is

also guaranteed to run in polynomial time with $1/\varepsilon$, it is called a *Fully Polynomial-Time Approximation Scheme* (FPTAS).

Many C&P problems, such as the BPP, are *APX-hard*, that is, there cannot exist a PTAS for them, unless $P = NP$. In these cases, it is possible to consider algorithms with asymptotic approximation factors, that is an algorithm A and a constant β such that $A(I) \leq \alpha OPT(I) + \beta$, where α is the asymptotic approximation factor and constant β is greater than 0. Similarly, we can define an *Asymptotic Polynomial-Time Approximation Scheme* as a family of approximation algorithms $\{A_\varepsilon\}$ with a constant β, that for any fixed $\varepsilon > 0$, produces a solution such that

$$A(I) \leq (1 + \varepsilon)OPT(I) + \beta.$$

Perhaps the most influential approximation algorithms for packing problems are the ones by Fernandez de La Vega and Lueker [14] and Karmarkar and Karp [26] for the classic BPP. The algorithm of Fernandez de La Vega and Lueker [14] is an APTAS based on the linear grouping of the items according to their lengths. The one of Karmarkar and Karp [26] is an approximation algorithm A such that $A(I) \leq OPT(I) + O(\log^2 OPT(I))$, based on the formulation of Gilmore and Gomory [17] and the ellipsoid method for solving linear programming.

Approximation algorithms are used to solve many different types of packing problems. Coffman et al. [8] provide a survey covering approximation algorithms for several bin packing problem variants. For further information on approximation algorithms, we refer to the books by Vazirani [46] and Williamson and Shmoys [48].

Heuristics e Metaheuristics

Heuristics for an optimization problem are straightforward problem-specific procedures that look for good solutions without concerns with guarantees in terms of objective value. Several simple heuristics have been proposed for packing problems, focusing on quickly finding good solutions. Among these, we have the *next fit* (NF) for the BPP, in which items are considered in order and the first one is packed in the first bin. From then on, we check if the following item can be packed in the same bin as the previous one, closing the bin and opening a new one for the following item in case it does not. The *first fit* (FF) heuristic packs subsequent items in the first open bin in which the item fits, meaning it has to check, for each item, possibly all previously created bins. Variations of these heuristics can be obtained by simply re-ordering items in non-increasing order of lengths, obtaining respectively *next fit decreasing* (NFD) and *first fit decreasing* (FFD). In his seminal Ph.D. thesis, Johnson [24] showed that these algorithms (and others similar to them) guarantee asymptotical approximation rates, making them approximation algorithms.

Metaheuristics are high-level methods, not necessarily designed for a particular problem or with well-defined steps, but instead focus on providing a framework to be applied to a wide range of problems. More specifically, according to Gendreau and Potvin [16], metaheuristics are "solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space".

The combination of heuristics and metaheuristics is an attractive approach in the context of solving difficult problems under strict computational time restrictions. For instance, a heuristic can be used to quickly find an initial solution to be improved by a more sophisticated metaheuristic that will return the best solution it can find, given a specific time limit. This same time limit approach cannot be achieved by exact or approximation algorithms, since they cannot be prematurely interrupted in order to keep their solution quality guarantees.

There are many metaheuristics for CO problems in the literature. They can usually be divided into three main groups: neighborhood search procedures such as *greedy randomized adaptive search procedures* (GRASP) [13], *tabu search* (TS) [18], *variable neighborhood search* (VNS) [33], *large neighborhood search* (LNS) [41] and others; population-based methods such as *genetic algorithms* (GA), *particle swarm optimization* (PSO) [28], *biased random key genetic algorithms* (BRKGA) [19] and others; and hybrid metaheuristics, which in addition to combinations of different types of metaheuristics, also include hybridization with other types techniques, such as *matheuristics*, that combine metaheuristics with mathematical programming [32].

The literature on metaheuristics for C&P problems is vast. Fleszar and Hindi [15] proposed a VNS algorithm for the BPP based on an initial solution given by a minimal bin slack algorithm [20]. Loh et al. [31] proposed a *weight annealing* metaheuristic for the BPP that was able to solve all the instances proposed by Fleszar and Hindi [15]. Later on, Buljubašić and Vasquez [6] presented a *consistent neighborhood search* and Kucukyilmaz and Kiziloz [29] provided an *island-parallel grouping genetic algorithm* for the BPP. Both algorithms performed similarly well in the experiments reported. However, most metaheuristic-related works on C&P focus on variants with a larger solution space, such as higher-dimension bin packing problems. Lodi et al. [30] review algorithms and bounds for the two-dimensional bin packing problem and variants, including many metaheuristics. Hopper and Turton [22] compare different metaheuristic approaches to a two-dimensional strip packing problem.

For an overview of the use of metaheuristics, we refer to the book of Gendreau and Potvin [16].

1.2 Contributions

The objective of this work is to develop techniques in exact, approximation, and heuristic approaches to different C&P problems, while carefully considering what approach fits best to what problems. To this extent, we tackle three different projects, each resulting in its own chapter of this thesis. Each project consists of a different problem approached with multiple techniques. In this section, we describe the main contributions of each project as well as some additional products obtained during their development.

1.2.1 Colored Bin Packing Problem

The colored bin packing problem (CBPP) is a generalization of the classical BPP in which each item also has a color, and no two items of the same color are allowed to be

packed consecutively in the same bin. For this problem, we focus mainly on exact models. We provide a characterization of any feasible packing of the CBPP in a way that does not depend on its ordering. We also propose four exact algorithms for the CBPP. The first is an adaptation of Valério de Carvalho's arc flow formulation using a graph with multiple layers, each representing a color. The second is an improved arc flow formulation that uses a more compact graph and has the same strong linear relaxation bound as the first one. Finally, we design two exponential set-partition models based on reductions to a generalized vehicle routing problem, which are solved by a branch-cut-and-price algorithm through VRPSolver. To evaluate the proposed models, we present a varied benchmark set with 574 instances.

The findings of this research were compiled in a journal article that was recently submitted for publication in an international journal of the field. This article is fully presented in Chapter 2.

Other Products

Some preliminary results of this project were also published as a four-page extended abstract at a Brazilian conference in 2020. The paper was titled, in Portuguese, "Modelos Pseudo Polinomiais para o Problema do Empacotamento Colorido" and appeared in the annals of the V Encontro da Teoria da Computação [3].

Concurrently with this work, we also collaborated with the undergraduate student Renan Fernando Franco da Silva to develop heuristic and metaheuristic approaches to the CBPP. The preliminary results of these efforts also appear as extended-abstracts in Brazilian conferences. The paper titled "Heurísticas para o Problema do Empacotamento Colorido" appears in the annals of the VI Encontro da Teoria da Computação in 2021 [9]. A second, and more detailed version of this work, also titled "Heurísticas para o Problema Do Empacotamento Colorido" appears in the annals of 54^o Simpósio Brasileiro de Pesquisa Operacional in 2022 [10]. These results will be compiled in a full article to be submitted to an international journal in the field soon, but are outside the scope of this thesis.

1.2.2 Bin Packing Problem with Scenarios

The bin packing problem with scenarios (BPPS) is a generalization of the classical BPP which considers the presence of uncertain scenarios, of which only one is realized. For this problem, we propose an absolute approximation algorithm whose ratio is bounded by the square root of the number of scenarios times the approximation ratio for an algorithm for the vector bin packing problem. We also show how an APTAS is derived when the number of scenarios is constant. As a practical study of the problem, we present an exact branch-and-price algorithm to solve an exponential model and a variable neighborhood search (VNS) heuristic. To speed up the convergence of the exact algorithm, we also consider lower bounds based on dual feasible functions. We also consider a version of the exact algorithm that uses the best solution found by the VNS heuristic as a warm start. To evaluate the practical study, we present a set of 120 instances adapted from the literature.

This project was a collaboration with the then Ph.D. student Vinícius Loti de Lima, and Professor Lehilton Lelis Chaves Pedrosa from the Institute of Computing at the University of Campinas, as well as Professor Thiago Alves de Queiroz from the Institute of Mathematics and Technology at the Federal University of Catalão. The findings were compiled in a journal article that was recently submitted for publication in an international journal of the field. This article is fully presented in Chapter 3.

Other Products

A preliminary version of these results, including a sketch of the APTAS was published in Portuguese as a four-page extended abstract at a Brazilian conference in 2019. The paper titled "Um esquema de aproximação para um problema de empacotamento com cenários" appears in the annals of the IV Encontro da Teoria da Computação [2].

1.2.3 Demand-side Management for Smart Grids

We consider energy allocations problems with consumption limiting in the context of an auction of renewable energy in smart grids. To contextualize, smart grids are electricity networks that can intelligently integrate the actions of all users connected to it [12]. These networks are expected to provide various benefits to society by integrating advances in power engineering with recent developments in the field of Information and Communications Technology. One of the advantages is the support of efficient demand-side management (DSM), e.g. changes in consumer demands for energy based on using incentives. Indeed, DSM is expected to help grid operators balance time-varying generation by wind and solar units, and the optimization of their usage. In this context, we propose an auction for DSM, in which consumers submit bids to renewable energy usage plans. The main problem consists of allocating these usage plans to consumers according to their bids while respecting the capacity of generating renewable energy of the grid. An additional model is introduced to allow consumers to compute their bid for a given usage plan according to their preferences. Both the energy allocation and usage plan bidding computation problems are modeled as variants of the vector knapsack problem, and solved using a mixed-integer linear programming formulation. The models are also extended to include energy storage devices. The impact of the use of energy storage in households and in the energy provider is also considered.

This project was a collaboration with the Ph.D. student Lucas Prado Melo, and Professor Nelson Luis Saldanha da Fonseca from the Institute of Computing at the University of Campinas, as well as Professor Fabrizio Granelli from the Department of Information Engineering and Computer Science at the University of Trento (Italy). The findings were compiled in a journal article published in International Transactions in Operational Research in 2020 [4]. This article is fully presented in Chapter 4, with the authorization for its reproduction in the thesis in Appendix A.

Other Products

A preliminary version of these results was presented in Portuguese as a workshop talk titled "Precificação Inteligente de Energia para Gerenciamento pelo Lado-da-Demanda em smart grids de Energia Renovável" at the XX Oficina Nacional de Problemas de Corte, Empacotamento, Planejamento e Programação da Produção e Correlatos, and the XIV Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica do Instituto de Computação da Universidade Estadual de Campinas in 2019.

1.3 Text Organization

This thesis is presented in an alternative format proposed by the University of Campinas in 2021 [23]. As defined in this format, the text is composed of this introductory chapter, followed by one chapter per published or submitted article and ending with discussions and conclusions. In Chapter 2, we present our results for the colored bin packing problem. In Chapter 3, we present the article on the bin packing problem with scenarios. We present our published article on demand-side management for smart grids in Chapter 4. At last, in Chapter 5, we present some general discussions as well as our concluding remarks. The authorization for reproducing the published article in Chapter 4 can be found in Appendix A.

Chapter 2

Mathematical Models and Exact Algorithms for the Colored Bin Packing Problem

Abstract This paper focuses on exact approaches for the Colored Bin Packing Problem (CBPP), a generalization of the classical one-dimensional Bin Packing Problem in which each item has, in addition to its length, a color, and no two items of the same color can appear consecutively in the same bin. To simplify modeling, we present a characterization of any feasible packing of this problem in a way that does not depend on its ordering. Furthermore, we present four exact algorithms for the CBPP. First, we propose a generalization of Valério de Carvalho's arc flow formulation for the CBPP using a graph with multiple layers, each representing a color. Second, we present an improved arc flow formulation that uses a more compact graph and has the same linear relaxation bound as the first formulation. And finally, we design two exponential set-partition models based on reductions to a generalized vehicle routing problem, which are solved by a branch-cut-and-price algorithm through VRPSolver. To compare the proposed algorithms, a varied benchmark set with 574 instances of the CBPP is presented. Results show that the best model, our improved arc flow formulation, was able to solve over 62% of the proposed instances to optimality, the largest of which with 500 items and 37 colors. While being able to solve fewer instances in total, the set-partition models exceeded their arc flow counterparts in instances with a very small number of colors.

Keywords bin packing problem; arc flow; VRPSolver; mixed integer linear programming;

2.1 Introduction

The *Bin Packing Problem* (BPP) consists of packing a set of items, each with a positive length, into the smallest possible number of bins such that the sum of the lengths of the items in a bin does not exceed its given capacity. A popular generalization of the BPP is the *Cutting Stock Problem* (CSP), in which items have a positive demand value that

indicates the number of times they must be included in a solution. For generality, we use the CSP notation for some of the models presented in this work.

Applications of cutting and packing problems have been studied for decades. In the packing context, there are many problems in production planning and load balancing, in which one wants to fit a set of boxes into the smallest possible number of containers. On the other hand, the term cutting has been adopted in the industrial process of cutting rolls of stock material into smaller pieces in order to satisfy a given demand while minimizing the number of rolls used, which is often encountered in industries such as paper, metal, and fabric. These problems can also be encountered as part of bigger optimization problems found in real-world applications such as capacitated vehicle routing, load balancing scheduling, and capacitated facility location, among others.

The *Colored Bin Packing Problem* (CBPP) is a generalization of the BPP in which each item, in addition to a length and demand, also has a color. The objective is to find a packing of all items in the smallest number of bins possible, in which no two items from the same color appear consecutively in the same bin. That is, for every bin in the solution there must exist at least one permutation of the items in it (considering all their respective copies in the case with demands) such that adjacent items always have different colors.

One possible application for the CBPP, in the particular case in which there are only two colors, is when there must be an intercalation of two types of items in a schedule. Balogh et al. [1] describe the example in which white items can represent blocks of a radio show and black items represent commercial breaks, with the items' lengths indicating their duration. These blocks of content must be scheduled in an alternating form and respect the shows' duration, usually a time window of one hour.

When considering multiple colors, the same idea applies: CBPP can model the need to alternate items of different types. For example, Dósa and Epstein [14] describe an application where one wants to print batches of flyers on papers with different colors, and must alternate the colors to make it easy to split the print into the correct batches afterward. As another example, one can consider alternating different kinds of information and advertisement on the screen of a smartphone on sites like YouTube and other social media platforms, as described by Balogh et al. [1].

Notice also that the development of good algorithms for CBPP could lead to the development of good algorithms for similar problems such as knapsack or scheduling problems where the items must alternate in some form. In fact, one can consider the CBPP as part of a larger class of packing problems where there are constraints on pairs of items, defining how or if they can be packed together.

Probably, the most famous problem in this class is the Bin Packing Problem with Conflicts, considered by Jansen and Öhring [18] and Jansen [17], where a list of conflicts between pairs of items is given, and two items can be packed in the same bin only if there is no conflict between them. This problem appears, for example, as a sub-problem when solving BPP with a branch-and-price algorithm using a branching rule that either forces two items to be packed together or creates a conflict between them, as considered by Vance et al. [28].

Another problem in this class is the Class Constrained Bin Packing Problem [26], where every item belongs to some class, and there is a limit on the number of different

classes that can be used in the same bin. One application for this problem is when there is a limited number of scissors in a machine for cutting metal coils, and the pieces can be grouped into classes of treatments to be received after the cut (for example, reducing thickness). Thus, the coils are cut into classes, each class receives its corresponding treatment, and then the final pieces are cut. This problem also has several applications in data load balance, such as Video-on-Demand servers [30]. One main difference between the Class Constrained Bin Packing Problem, when compared with Bin Packing Problem with Conflicts, is that in the former, there is no direct conflict between items but, instead, what prevents us from adding a new item to a bin is the whole set of items already packed. Nonetheless, the ordering of the packing does not matter.

Peeters and Degraeve [22] presented the Co-Printing Problem, a variant of the Class Constrained Bin Packing Problem where items have the same length, but can be of multiple classes at the same time. This is done to model a problem when printing Tetra Pak packages in a printer that can use a limited number of colors at once.

There are other packing problems where the order of the packing matters, such as Bin Packing with Largest in the Bottom Constraint proposed by Manyem et al. [21], where the largest items must be packed before the smaller ones in the bin. Notice that the difficulty of these problems usually arises when the items must be packed in an online fashion. The CBPP problem lies in a frontier between these kinds of problems. While the order of each packing matters for the online setting, in the offline setting, which is the focus of this paper, what matters is only the number of items of each color.

2.1.1 Previous Works

Previous works on CBPP are either from the perspective of approximation or online algorithms. These works consider three different versions of the problem: the *offline* version, where the whole input is known at the beginning of the algorithm; the *restricted offline* version, where the input is known at the beginning of the algorithm, but the items must be packed in the given order; and the *online* version, where items arrive in an online fashion and must be packed definitively in a selected bin, without knowledge of what items will come next. Recall that an online algorithm A for a minimization problem is called an *α-competitive algorithm* if, given an instance I, it produces a solution of objective value $A(I)$ that is at most α times greater than value of an offline (or in our case restricted offline) optimal solution for instance of I, $OPT(I)$. We also say that α is the *competitive ratio* of A.

In 2012, Balogh et al. [1] introduced the Black and White Bin Packing Problem (BWBPP) as a precursor of the CBPP in which only two colors are considered. They presented a lower bound of 1.7213 on any online algorithm for the BWBPP, as well as a 3-competitive algorithm. They also give a 2.5-approximation algorithm and an APTAS for the offline (unrestricted) version. These results were later published as two full articles in journals [3, 2].

In 2014, Dósa and Epstein [14] introduced the CBPP. They proved that there is no optimal online algorithm for the zero-sized items variant of the CBPP, contrasting with BWBPP where such an algorithm exists. Their main results are a 4-competitive online

algorithm and a lower bound of 2 on the asymptotic competitive ratio of any online algorithm. This last result also holds for the BWBPP, thus improving the previous result by Balogh et al. [1]. Chen et al. [8] studied the special case of the BWBPP where items have lengths of at most half of the capacity of the bin, and presented a 8/3-competitive algorithm for it, improving on the result of Balogh et al. [1] which had a competitive ratio of 3 even for this particular case.

Finally, Böhm et al. [6] considered the case of the CBPP where every item has length zero. They proved that the restricted offline optimum is always equal to the *color discrepancy*, a measure of how unbalanced the colors in the instance are. Then, they provided an asymptotically 1.5-competitive algorithm, which is, in fact, optimal as any deterministic online algorithm is shown to have at least this competitive ratio. The algorithm also has an absolute competitive ratio of 5/3. For the general case, that is, items with arbitrary lengths and at least three colors, they designed a 3.5-competitive algorithm and presented a lower bound of 2.5 on the competitive ratio of any deterministic online algorithm for this problem. Interestingly, they also show that classical BPP algorithms First Fit, Best Fit, and Worst Fit do not present a constant competitive ratio for the CBPP. For the BWBPP problem on the other hand, they prove that any fit algorithms (such as First Fit) have an absolute competitive ratio of 3 and that any deterministic online algorithm has an asymptotic competitive ratio of 2.

More recently, Bilò et al. [4, 5] consider a game-theory version of the problem, where players control the items and decide where to pack them.

2.1.2 Related Works

In this section, we discuss some previous works that are directly connected to the contributions of this paper. The BPP and CSP have been the birthplace of some crucial development in mixed integer linear programming in the past. Gilmore and Gomory [15] modeled the CSP with a set-covering model, in which columns represent all possible cutting configurations of a stock roll. To solve this model, they used the *column generation* technique, in which cutting configurations are considered implicitly, that is, columns are only generated on an on-demand basis in an iterative process. This set-covering model was shown to have a very strong linear relaxation. Let $LP(I)$ be its linear relaxation value and $OPT(I)$ be the optimal value for instance I, the instance is said to have the *Integer Round Up Property* (IRUP) if $OPT(I) = \lceil LP(I) \rceil$. Similarly, instance I is said to have the *Modified Integer Round Up Property* (MIRUP) if $OPT(I) - \lceil LP(I) \rceil \leq 1$. It is conjectured that MIRUP holds for all instances of the CSP and BPP [25].

Later on, Valério De Carvalho [27] proposed the *arc flow* model for the CSP based on a previous work of Wolsey [29] that connected integer programming with network flows. Given a directed graph where each vertex represents a cutting level of a stock roll, and arcs represent items, with its weight equal to the corresponding item length, a feasible cutting configuration can be found by computing a shortest path in such a graph. Valério De Carvalho [27] used this idea as the basis for a branch-and-price algorithm for the CSP and showed that the arc flow formulation is equivalent to Gilmore and Gomory's set-covering model in terms of their linear relaxation.

Kramer et al. [20] approached a problem of job scheduling on parallel machines with family setup times, in which an additional setup time was required between two consecutive jobs of different families. They proposed an original arc flow formulation that uses one layer per family of jobs, modeling setup times as arcs crossing between layers. The authors also proposed another arc flow model that uses a single layer, but divides arcs into job-arcs, dummy-arcs which are zero-sized and allow for the transition between jobs from the same family, and setup-arcs which represent the setup time between two different families. The model requires that any path must alternate job-arcs with either a setup-arc or a dummy arc.

Recently, Pessoa et al. [24] proposed a generic branch-and-cut-and-price solver that included several computational optimizations and improvements that have been proposed in the vehicle routing literature in the past few years. Their model is general enough to be applied to many combinatorial optimization problems. This was achieved by designing a generic Vehicle Routing Problem (VRP) variant that can be modeled as a set-cover or set-partition problem in which columns can be generated by solving a Resource-Constrained Shortest Path Problem (RCSP). Additionally, several critical algorithmic improvements were generalized for this variant. Even though the model's priority is generality, the authors presented experimental results for the BPP that are very competitive with state-of-the-art problem-specific algorithms and models at the time.

2.1.3 Our Contributions

First, we provide a characterization of any feasible packing for the CBPP in a way that does not depend on its ordering, making algorithmic modeling much simpler. This characterization, along with other preliminary concepts and definitions are presented in Section 2.2. Afterward, we propose the pseudo-polynomial multilayered arc flow formulation, which generalizes Valério de Carvalho's classic CSP model for the CBPP. Furthermore, we propose the original color-alternating arc flow formulation which works with a much more compact graph while retaining the same strong linear relaxation. These pseudo-polynomial formulations are discussed in Section 2.3. Later on, we propose two exponential set-partition models based on reductions to a generalized vehicle routing problem which are solved by a branch-cut-and-price algorithm through VRPSolver. These models are detailed in Section 2.4. Finally, we propose a new benchmark set that includes uniformly randomly generated instances, instances adapted from the CSP/BPP literature, as well as randomly generated instances with a Zipf distribution of colors. This benchmark set, as well as the numerical experiments evaluating all the models introduced in this work, are presented in Section 2.5. In Section 2.6, we recapitulate the results and provide our final remarks.

2.2 Preliminaries

In this section, we provide a few definitions to contextualize our contributions. Along the text, for any $k \in \mathbb{Z}^+$, we denote $\{1, 2, \ldots, k\}$ by $[k]$.

Colored Bin Packing Problem (CBPP) An instance of the problem is composed of an integer bin capacity $L > 0$, an integer number of colors $Q \geq 2$ and a set $\mathcal{I} = [m]$ of items, each with an integer length $l_u > 0$, integer demand value $d_u > 0$ and color $c_u \in [Q]$ for all $u \in \mathcal{I}$. Since each item has a demand, we consider that all pairs of length and color are unique in $\mathcal{I}$. We also consider an auxiliary set $\mathcal{I}'$ composed of d_u copies of each item $u \in \mathcal{I}$. This way, a solution for CBPP is a partition $\mathcal{B}$ of $\mathcal{I}'$, such that for each part $B \in \mathcal{B}$, the total length of the items in B is at most L, and there exists at least one permutation of B in which no two items of the same color appear consecutively. The objective is to find a solution that minimizes the number of parts. We consider that for any $S \subseteq \mathcal{I}$, $\{S_1, S_2, \ldots, S_Q\}$ is a partition of S by color, that is $u \in S_q$ if and only if $u \in S$ and $c_u = q$.

Before proceeding, we introduce the following example to better illustrate the problem. Consider an instance with bin capacity equal to 8 and items as described in Figure 2.1.

items			
u	l_u	d_u	c_u
1	4	2	1
2	3	1	2
3	3	1	3
4	2	1	3

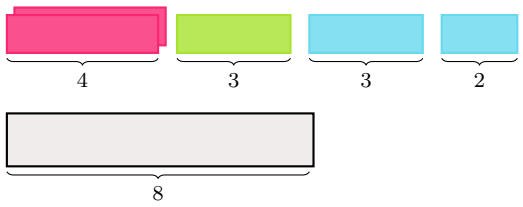

Figure 2.1: Example of an instance of the CBPP with 4 items, 3 colors and bin capacity 8.

An optimal solution for the instance in the example would be to pack the first copy of item 1 with item 2 in a single bin, and the second one with item 3 in another bin. This would make it necessary to use a third bin to pack item 4 by itself. This solution is illustrated in Figure 2.2.

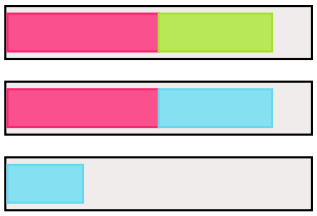

Figure 2.2: An optimal solution for the previous example.

We call a *packing pattern* any subsequence P of items in $\mathcal{I}'$ such that the sum of the lengths of the items in P is smaller than or equal to L and, for $1 \leq k < |P|$, the color of the k-th item of P is different from that of the $(k+1)$-th one. With this, we may also describe a solution for the CBPP as the smallest set of packing patterns that satisfy the demands exactly for every item $u \in \mathcal{I}$. Note that, unlike in the CSP, a solution for the CBPP cannot contain more than d_u copies of an item $u \in \mathcal{I}$, as otherwise two items of the same color could be intercalated with an additional differently colored item, possibly reducing the necessary number of bins.

In the following, we recall the important concept of *color discrepancy* defined by Balogh et al. [2] and generalized by Böhm et al. [6]. Let S be a fixed subset of $\mathcal{I}'$ and let $s_{q,u} = 1$

if item $u \in S$ is from color $q \in [Q]$ and $s_{q,u} = -1$ if item u is from any color other than q. The *discrepancy* between q-items and non-q-items in S is $\delta_q^S = \sum_{u \in S} s_{q,u}$ for all $q \in [Q]$, and the *color discrepancy* of S is $\delta^S = \max_{q \in [Q]} \delta_q^S$. Finally, we define the *critical color* of S as $q_S^* = \operatorname{argmax}_{q \in [Q]} \delta_q^S$ (choosing arbitrarily in case of a tie). We omit S from the notation in δ and q^* unless it is necessary since it is usually clear by context. Theorem 1 gives us simple conditions to determine whether any subsequence of items admits a *color-alternating permutation*, i.e. the items in it can be rearranged in such a way that no two items of the same color appear side-by-side. This allows us to verify, in linear time, if there exists a permutation of a given subset of items S with $\sum_{u \in S} l_u \leq L$ that is a valid packing pattern.

Theorem 1. *Let $S \subseteq \mathcal{I}'$ and $\sum_{u \in S} l_u \leq L$. The following are equivalent:*

1. *S admits a color-alternating permutation;*
2. *$|S_{q^*}| \leq \lceil |S|/2 \rceil$;*
3. *$\delta \leq 1$.*

The proof of this theorem follows from the lemmas below.

Lemma 1. *If S admits a color-alternating permutation, then $|S_{q^*}| \leq \lceil |S|/2 \rceil$.*

Proof. We prove that if $|S_{q^*}| \geq \lceil |S|/2 \rceil + 1$, then S does not admit a color-alternating permutation. By the pigeonhole principle, since there are $|S_{q^*}|$ q^*-items, there must be at least $|S_{q^*}| - 1$ non-q^*-items for S to admit a color-alternating permutation. Thus, there must be at least $2|S_{q^*}| - 1 \geq 2(\lceil |S|/2 \rceil + 1) - 1 \geq |S| + 1$ items in total, a contradiction. □

Lemma 2. *If $|S_{q^*}| \leq \lceil |S|/2 \rceil$, then $\delta \leq 1$.*

Proof. Since δ is the difference of the number of q^*-items and non-q^*-items in S, we can write it as $\delta = |S_{q^*}| - t$, where t is the amount of non-q^*-items in S. Also, notice that if $|S_{q^*}| \leq \lceil |S|/2 \rceil$ holds, then $t \geq \lfloor |S|/2 \rfloor$. Thus, we have

$$\delta = |S_{q^*}| - t \leq |S_{q^*}| - \lfloor |S|/2 \rfloor \leq \lceil |S|/2 \rceil - \lfloor |S|/2 \rfloor \leq 1. \qquad \square$$

Lemma 3. *If $\delta \leq 1$, then S admits a color-alternating permutation.*

Proof. For this, we give the algorithm ALTERNATE that receives $S \subseteq \mathcal{I}'$ such that $\delta \leq 1$ and returns a color-alternating permutation of S. We assume that the input set is partitioned by colors and can be accessed by color index. Also, we assume, without loss of generality, the items are non-increasing ordered by color frequency.

ALTERNATE(S, Q)

```
1  Let 𝓛 be a list of |S| initially empty sequences
2  k = 0
3  for u ∈ S_1
4        k = k + 1                  // Counts amount of open sequences
5        𝓛_k = 𝓛_k ⌢ (u)            // Operator ⌢ means concatenation at the end of the sequence
6  j = 1
7  for q ∈ {2, ..., Q}
8        for u ∈ S_q
9              𝓛_j = 𝓛_j ⌢ (u)
10             j = (j + 1) mod k    // Places one item in each of the k sequences before returning to the first
11 return 𝓛_1 ⌢ ··· ⌢ 𝓛_k
```

First, we argue that the color which has the most items in S is color q^*. For this, consider a color q with δ_q and color q' with $\delta_{q'}$. If $\delta_q > \delta_{q'}$ then

$$|S_q| - (|S| - |S_q|) > |S_{q'}| - (|S| - |S_{q'}|) \implies 2|S_q| - |S| > 2|S_{q'}| - |S| \implies |S_q| > |S_{q'}|.$$

The first inequality follows from the fact that δ_q can be computed as the number of items from color q minus the number of items from the other colors, which can be stated as $|S| - |S_q|$. With this, we have that the colors are ordered by their discrepancies in the input.

Now we show that algorithm ALTERNATE is correct. In the first loop, lines 3-5, each item from the first color is placed in a separate sequence, and variable k counts the number of sequences used. Notice that, since $\delta \leq 1$, there are enough remaining items to put in each of the k sequences except, possibly, the last one, which in turn would make the concatenation of these sequences a valid color-alternating permutation.

However, it would still be possible that there are more items from the following colors, so we alternate the items into k sequences and return to the first sequence after an item is placed in the k-th one. The main loop, in lines 7-10, uses this method to place all items in one of the sequences. Now suppose that, with this method, one item is followed by another one from the same color q in a single sequence, then there must be more than k items from color q, which means $\delta_q > \delta$ resulting in a contradiction. Thus, with this method, there will be no item followed by another of the same color in a single sequence.

Since each sequence only has one item from each color, and all sequences start with an item of color 1 but end in an item with a different color (except, possibly the last one), the algorithm returns a concatenation of the sequences in order, which must be a valid color-alternating permutation of S. □

2.3 Pseudo Polynomial Models

In this section, we propose two new pseudo-polynomial formulations for the CBPP. Before describing our contributions, we remind the reader about the arc flow formulation of Valério De Carvalho [27] for the BPP. The concepts of this formulation are crucial to

the understanding of the models we propose in this section.

In the arc flow model, we consider the directed graph $G = (V, A)$ as follows: V consists of all possible packing points of a bin, i.e. $V = \{0, 1, \ldots, L\}$; A_{item} is the set of item-arcs, which connect every i and j such that there exists an item of length $j - i$, i.e. $A_{item} = \{(i, j) \colon j - i = l_u \text{ for } 0 \leq i < j \leq L, \text{ and } u \in \mathcal{I}\}$; A_{loss} is the set of loss-arcs that connect every pair of adjacent vertices, i.e. $A_{loss} = \{(k, k+1) \text{ for } 0 \leq k \leq L-1\}$; and $A = A_{item} \cup A_{loss}$. A path from 0 to L in G corresponds to a valid packing pattern for an instance of the BPP, in which the presence of an item-arc (i, j) in the path represents the packing of an item of length $j - i$, and the presence of loss-arcs represent empty spaces left in the bin. With this in mind, the BPP can be modeled as the problem of finding a set of paths in G with the smallest cardinality that covers the demands for all items in the instance, which is achieved with the arc flow integer programming formulation of Valério De Carvalho [27].

2.3.1 Multilayered Arc Flow

A straightforward adaptation of the arc flow model for the CBPP is presented in this section. The idea is based on the construction of a graph in which the vertices are all the packing points repeated for each color. We can then add arcs that only connect vertices of different colors, thus enforcing that any paths must alternate colors.

Consider the digraph $G(V, A)$ as follows:

- The set of vertices V is partitioned as $\{V_0, V_1, V_2, \ldots, V_Q\}$:
  - V_0 is the set with a single source vertex $(0, 0)$,
  - $V_q = \{(1, q), (2, q) \ldots, (L, q)\}$ is the set of packing points for color q;
- The set of arcs A is partitioned as $\{A_{source}, A_{item}, A_{loss}\}$:
  - $A_{source} = \{\big((0, 0), (j, q)\big) : (j, q) \in V \text{ and } \exists u \in \mathcal{I} : l_u = j, c_u = q\}$,
  - $A_{item} = \{\big((i, q), (j, q')\big) : (i, q) \in V, (j, q') \in V_{q'},\ q \neq q' \text{ and } \exists u \in \mathcal{I} : l_u = j - i, c_u = q'\}$,
  - $A_{loss} = \{\big((i, q), (L, q)\big) : (i, q), (L, q) \in V \text{ for any } q \in [Q] \text{ and } 1 \leq i \leq L - 1\}$.

With this, each color has L vertices, resulting in $|V| = LQ + 1$ and $|A| = O(LQ|\mathcal{I}|)$. An item-arc connects a vertex $(i, q) \in V$ to $(j, q') \in V$ where $q \neq q'$ and there is an item in $\mathcal{I}$ with color q' and length $j - i$, while a loss-arc connects any vertex in V_q to the sink vertex of that color (L, q) for any $q \in [Q]$. Notice that loss-arcs in this model are different from those of the original model for the BPP, as they enforce all empty spaces to be placed at the end of a bin, thus avoiding alternating items with empty spaces. Hence, departing from any vertex, all arcs emanating from it either arrive at a vertex of a different color from the one it emanates from or arrive at the sink of that color.

Like with the arc flow formulation, we can model the CBPP as the problem of finding the smallest number of paths from 0 to L in G such that the demand for each item is attended to equality. Formulation AF_{ml} described below is an adaptation of the arc flow

model of Valério De Carvalho [27] to work with the graph and notation introduced in this section. In it, x is a vector of integer variables that indicates the amount of flow passing through each arc, and z is the integer variable indicating the cumulative amount of flow passing through all arcs that emanate from the source vertex V_0.

$$\mathrm{AF}_{\mathrm{ml}} = \min z \tag{2.1}$$

$$\text{s.t.} \sum_{((i,q'),(j,q))\in A} x_{iq'jq} - \sum_{((j,q),(k,q''))\in A} x_{jqkq''} = \begin{cases} -z & \text{for } j = 0, q = 0 \\ 0 & \text{for } j \in [L-1], q \in [Q] \\ z & \text{for } j = L, q \in [Q] \end{cases} \tag{2.2}$$

$$\sum_{\substack{((i,q'),(j,q))\in A \\ j=i+l_u, q=c_u, q'\neq q}} x_{iq'jq} = d_u \qquad \text{for } u \in \mathcal{I} \tag{2.3}$$

$$z, x_{iq'jq} \in \mathbb{Z}^+ \qquad \forall\, ((i,q'),(j,q)) \in A, \tag{2.4}$$

The objective (2.1) is to minimize the number of paths (or patterns) used. Constraints (2.2) guarantee the conservation of flow in all vertices of G, except in the source $(0,0)$ and sinks (L,q) for any $q \in [Q]$. Constraints (2.3) ensure that the demands be satisfied to the equality for every item in $\mathcal{I}$. Finally, Constraints (2.4) are the integrality constraints.

Since there are no arcs in A that connect vertices of the same color, any path must intercalate vertices of different colors by using arcs representing items of different colors. The model will find the smallest set of paths that satisfy the demands to equality, which will also be an optimal solution for the CBPP since any valid pattern can be represented as a path in G. Figure 2.3 shows a graph representing an optimal solution of $\mathrm{AF}_{\mathrm{ml}}$ for the example introduced in Figure 2.1.

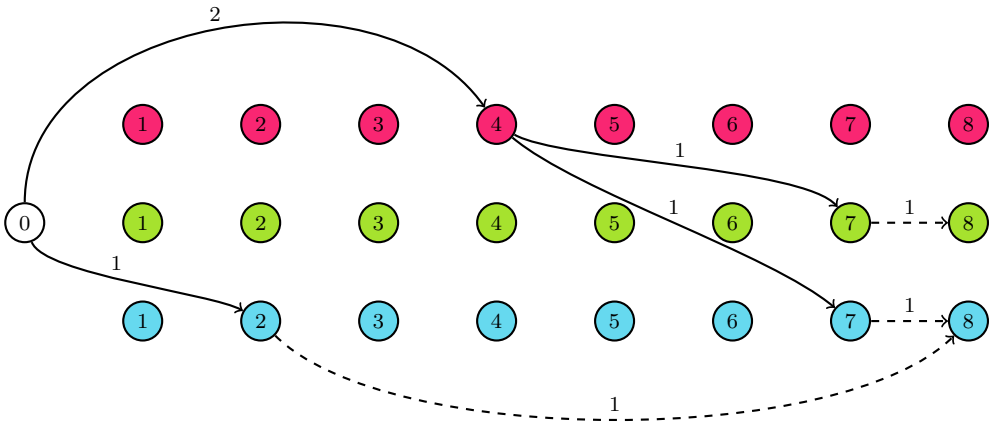

Figure 2.3: Optimal solution of model $\mathrm{AF}_{\mathrm{ml}}$ for the example of Figure 2.1. Loss-arcs are represented with dashed lines, and arc labels represent the amount of flow passing through it in the solution.

2.3.2 Color-Alternating Arc Flow

Although formulation AF_{ml} correctly models the CBPP, it requires a large graph with up to $LQ+1$ vertices and $O(LQ|\mathcal{I}|)$ arcs. Because of this, we propose an alternative model that requires at most $L+1$ vertices and $O(L|\mathcal{I}|)$ arcs, by using a multi-graph instead. We describe this model in this section.

Consider a directed multi-graph $G(V, A)$ in which arcs are labeled with a color, that is, an arc is a tuple (i, j, q) with i and j being its start and end points, and q its color. We consider the additional color $Q+1$ to indicate the loss-arcs. The graph is defined as follows:

- The set of vertices V is the packing points $\{0, 1, \ldots, L\}$;
- The set of arcs A is partitioned as $\{A_1, A_2, \ldots, A_Q, A_{Q+1}\}$:
  - $A_q = \{(i, j, q) : 0 \leq i \leq j \leq L, \exists u \in \mathcal{I} : l_u = j - i, c_u = q\}$ for $q \in [Q]$, and
  - $A_{Q+1} = \{(i, L, Q+1) : 1 \leq i \leq L-1\}$.

With this, the number of vertices is $L+1$ and there is an item-arc of color $q \in [Q]$ between vertices $i \in V$ and $j \in V$ if there is an item of color q and length $j-i$ in $\mathcal{I}$. There is also a loss-arc connecting each vertex $i \in V \setminus \{0, L\}$ to L. However, to guarantee the color constraints in this graph we must specify that two arcs from the same color cannot be used consecutively in any path. Additionally, since the loss-arcs can only arrive at vertex L, we guarantee that empty spaces are always placed at the end of the bin, which is necessary to avoid items from the same color being alternated with empty spaces. We propose formulation AF_{ca}, in which x is a vector of integer variables that indicate the amount of flow going through each arc and z is the integer variable that indicates cumulative the amount of flow passing through all arcs that emanate from the source vertex 0.

$$\text{AF}_{\text{ca}} = \min\ z \tag{2.5}$$

$$\text{s.t.} \quad \sum_{(i,j,q)\in A} x_{ijq} - \sum_{(j,k,q')\in A} x_{jkq'} = \begin{cases} -z & \text{for } j = 0, \\ 0 & \text{for } j = 1, \ldots, L-1, \\ z & \text{for } j = L, \end{cases} \tag{2.6}$$

$$\sum_{(i,j,q)\in A} x_{ijq} - \sum_{\substack{(j,k,q')\in A: \\ q'\in[Q+1]\setminus\{q\}}} x_{jkq'} \leq 0 \qquad \text{for } j = 1, \ldots, L-1, q \in [Q] \tag{2.7}$$

$$\sum_{\substack{(i,j,q)\in A \\ j=i+l_u, c_u=q}} x_{ijq} = d_u \qquad \text{for } u \in \mathcal{I}, q \in [Q] \tag{2.8}$$

$$z, x_{ijq} \in \mathbb{Z}^+ \qquad \forall\, (i, j, q) \in A, q \in [Q] \tag{2.9}$$

The objective (2.5) is to minimize the number of paths used. Constraints (2.6) guarantee the conservation of flow. Constraints (2.7) ensure that the amount of flow passing through arcs of color $q \in [Q]$ that arrive in a vertex $j \in V$ cannot be greater than the amount of flow emanating from j through arcs of any color other than q. Constraints (2.8)

ensure that the demands be satisfied to equality for every item in $\mathcal{I}$. Finally, Constraints (2.9) are the integrality constraints. Figure 2.4 shows a graph representing an optimal solution of AF_{ca} for the example introduced in Figure 2.1.

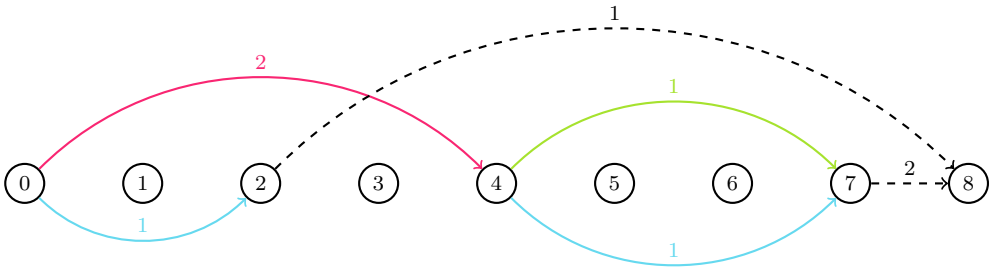

Figure 2.4: Optimal solution of model AF_{ca} for the example of Figure 2.1. Arcs are colored according to the item they represent, loss-arcs are represented with dashed lines, and arc labels represent the amount of flow passing through it in the solution.

We claim that Constraints (2.7) guarantee the color constraints of the CBPP. The idea is that if they are satisfied for a vertex $j \in V$, then there must exist a decomposition of the solution into paths such that any path that passes through j arrives from an arc of a different color than the one it leaves from. We show that, for each integer point in the polytope defined by AF_{ca}, there exists a feasible solution to the CBPP associated with it. We enunciate Lemmas 4 and 5, which help achieve this result with Lemma 6. Furthermore, we show that any CBPP solution can be mapped into an integer point of AF_{ca} with the same objective value, in Lemma 7.

Lemma 4. *If any feasible solution $(\bar{x}, \bar{z})$ of AF_{ca} uses arcs from at least two different colors, say a and b, and satisfy Constraints (2.7) to equality for both these colors at vertex $j \in V \setminus \{0, L\}$, then the only arcs $(i, j, q) \in A$ with $\bar{x}_{ijq} > 0$ have $q \in \{a, b\}$.*

Proof. Since, in vertex j, $\bar{x}$ satisfy Constraints (2.7) to equality for a and b we have that

$$\sum_{(i,j,a)\in A} \bar{x}_{ija} = \sum_{\substack{(j,k,q)\in A \\ q\in[Q+1]\setminus\{a\}}} \bar{x}_{jkq} = \sum_{(j,k,q)\in A} \bar{x}_{jkq} - \sum_{(j,k,a)\in A} \bar{x}_{jka}, \tag{2.10}$$

$$\sum_{(i,j,b)\in A} \bar{x}_{ijb} = \sum_{\substack{(j,k,q)\in A \\ q\in[Q+1]\setminus\{b\}}} \bar{x}_{jkq} = \sum_{(j,k,q)\in A} \bar{x}_{jkq} - \sum_{(j,k,b)\in A} \bar{x}_{jkb}. \tag{2.11}$$

Now, notice that

$$\sum_{(i,j,q)\in A} \bar{x}_{ijq} \geq \sum_{(i,j,a)\in A} \bar{x}_{ija} + \sum_{(i,j,b)\in A} \bar{x}_{ijb} \tag{2.12}$$

$$= \sum_{(j,k,q)\in A} \bar{x}_{jkq} - \sum_{(j,k,a)\in A} \bar{x}_{jka} + \sum_{(j,k,q)\in A} \bar{x}_{jkq} - \sum_{(j,k,b)\in A} \bar{x}_{jkb} \tag{2.13}$$

$$= 2 \sum_{(j,k,q)\in A} \bar{x}_{jkq} - \sum_{(j,k,a)\in A} \bar{x}_{jka} - \sum_{(j,k,b)\in A} \bar{x}_{jkb} \tag{2.14}$$

$$\geq 2 \sum_{(j,k,q)\in A} \bar{x}_{jkq} - \sum_{(j,k,q)\in A} \bar{x}_{jkq} \tag{2.15}$$

$$= \sum_{(j,k,q)\in A} \bar{x}_{jkq}. \tag{2.16}$$

Since any feasible solution must satisfy the flow conservation imposed by Constraints (2.6), both inequalities above are, in fact, equalities. Thus, there cannot exist arcs arriving at or emanating from j from any color other than a or b with positive value in $\bar{x}$. □

Corollary 1. *For any feasible solution $(\bar{x}, \bar{z})$ of $\mathrm{AF}_{\mathrm{ca}}$ and vertex $j \in V \setminus \{0, L\}$, the number of colors that satisfy Constraints (2.7) to equality at j and $\sum_{(i,j,q)\in A} \bar{x}_{ijq} > 0$ is at most 2.*

We show that if a solution $(\bar{x}, \bar{z})$ is feasible, then there must exist a way to extract one *color-alternating* path from 0 to L from $(\bar{x}, \bar{z})$ and keep the remaining solution feasible. Color-alternating paths are those in which no arcs of the same color appear consecutively. The following definition will help us understand when the removal of a specific pair of arcs from $(\bar{x}, \bar{z})$ will result in an infeasible solution.

Definition 1. *Given a feasible solution $(\bar{x}, \bar{z})$ of $\mathrm{AF}_{\mathrm{ca}}$, a vertex $j \in V \setminus \{0, L\}$, and an arc (i, j, a), an arc (j, k, b) is blocked for (i, j, a) if $a = b$ or there is a color $q \notin \{a, b\}$ such that*

$$\sum_{(i',j,q)\in A} \bar{x}_{i'jq} = \sum_{\substack{(j,k',q')\in A \\ q'\in[Q+1]\setminus\{q\}}} \bar{x}_{jk'q'}.$$

Lemma 5. *Given a feasible solution $(\bar{x}, \bar{z})$ of $\mathrm{AF}_{\mathrm{ca}}$, a vertex $j \in V \setminus \{0, L\}$, and an arc (i, j, a) such that $\bar{x}_{ija} > 0$, there exists at least one unblocked arc (j, k, b) with $\bar{x}_{jkb} > 0$.*

Proof. Since $\bar{x}_{ija} > 0$, then, by Corollary 1, the number of colors that satisfy Constraints (2.7) to equality for vertex j is either 0, 1, or 2. Thus, we have the following cases.

If there are no colors that satisfy Constraints (2.7) to equality, then, by Constraint(2.7), there is an arc (j, k, b) with $b \neq a$ with $\bar{x}_{jkb} > 0$, which is unblocked.

In case there is exactly one color c that satisfies Constraint (2.7) to equality, if $c = a$, then any arc (j, k, b) with $b \neq a$ is unblocked and, by Constraint (2.7), there must be such an arc with $\bar{x}_{jkb} > 0$. And, if $c \neq a$, then, by Constraint (2.6),

$$\sum_{(i',j,a)\in A} \bar{x}_{i'ja} + \sum_{(i',j,c)\in A} \bar{x}_{i'jc} \leq \sum_{(i',j,q)\in A} \bar{x}_{i'jq} = \sum_{\substack{(j,k',q')\in A \\ q'\in[Q+1]\setminus\{c\}}} \bar{x}_{jk'q'} + \sum_{(j,k',c)\in A} \bar{x}_{jk'c},$$

thus as c satisfies Constraint (2.7) to equality and $\bar{x}_{ija} > 0$, there is an unblocked arc (j, k, c) such that $\bar{x}_{jkc} > 0$.

Lastly, if there are two colors, c and c', that satisfy Constraints (2.7) to equality, then by Lemma 4 these are the only colors with arcs of positive flow in j. Therefore, without loss of generality, $c = a$ and, by Constraint (2.6),

$$\sum_{(i',j,a)\in A} \bar{x}_{i'ja} + \sum_{(i',j,c')\in A} \bar{x}_{i'jc'} = \sum_{(i',j,q)\in A} \bar{x}_{i'jq} = \sum_{\substack{(j,k',q')\in A \\ q'\in[Q+1]\setminus\{c\}}} \bar{x}_{jk'q'} + \sum_{(j,k',c')\in A} \bar{x}_{jk'c'},$$

from which we conclude that there is an unblocked arc (j, k, c') such that $\bar{x}_{jkc'} > 0$. □

With Lemmas 4 and 5 in place, we can describe an algorithm to decompose a feasible solution into color-alternating paths. Algorithm DecomposeAF takes an integer feasible solution $\bar{x}$, and the instance graph $G(V, A)$ as input and produces a set of z color-alternating paths.

DecomposeAF$(\bar{x}, \bar{z}, V, A)$

```
1  if z̄ = 0
2       return ∅
3  in = arbitrary arc (0, j, a) ∈ A with x̄_{0ja} > 0
4  P = {in}
5  while j ≠ L
6       for each arc out = (j, k, b) ∈ A with x̄_{jkb} > 0 and b ≠ a
7            for q = 1 to Q and q ≠ b
8                 if Σ_{(i,j,q)∈A} x̄_{ijq} = Σ_{(j,k,q')∈A:q'≠q} x̄_{jkq'}
9                      block arc out                    // Arc out is blocked by color q
10           if arc out is not blocked
11                P = P ∪ {out}
12                in = out, a = b, j = k
13                break
14 ď = min_{(i,j,q)∈P} x̄_{ijq}
15 for each arc (i, j, q) ∈ P
16      x̄_{ijq} = x̄_{ijq} − ď
17      z̄ = z̄ − ď
18 return {(ď, P)} ∪ DecomposeAF(x̄, z̄, V, A)
```

The idea is that with each call of DecomposeAF for a solution $(\bar{x}, \bar{z})$ of $\mathrm{AF}_{\mathrm{ca}}$, one color-alternating path is removed from $\bar{x}$ and added to the output while also maintaining feasibility of the remaining solution. Then, by induction, the same algorithm can be repeated to the remaining solution until there are no arcs with positive flow left.

Lemma 6. *Algorithm* DecomposeAF *decomposes a solution $\bar{x}$ of the model $\mathrm{AF}_{\mathrm{ca}}$ into color-alternating paths from 0 to L in $G(V, A)$ with flow equal to $\bar{z}$.*

Proof. Condition in Line 1 is the base case, for when there is no positive flow in the solution.

The main loop, in Lines 5-13, iterates over V starting from the endpoint of the arc chosen in the previous iteration until it reaches L. Loop in Lines 6-13 iterates over all arcs with positive flow and color different from a to find an unblocked arc *out* with positive flow. From Lemma 5, we know that this loop will always find such an arc, which is then added to the current path and the loop continues from its endpoint. Line 14 computes the smallest amount of flow $\check{d}$, passing through the arcs in the recently formed path. This is the amount of flow that can be removed from all arcs in the path while maintaining the remaining solution feasible and setting at least one variable of the new solution $(\bar{x}, \bar{z})$ to zero. The loop in Lines 15-17 decrements exactly $\check{d}$ from all variables representing arcs in the recently formed path. At last, in Line 18, the current path and its multiplicity are concatenated with the result of a recursive call of the algorithm on the remaining solution.

Notice that before the recursive call, the same amount of flow is removed from a set of arcs connecting 0 to L, which in turn keeps Constraints (2.6) satisfied. Since none of the removed pairs of arcs are blocked by any color, the remaining solution before any recursive call to DecomposeAF is always feasible. Furthermore, since the amount of flow removed from the arcs in each iteration is guaranteed to set at least one variable to zero, this procedure is guaranteed to eventually stop (in at most $O(|A|)$ steps).

Notice that, adding $\check{d}$ for every path P the value obtained is precisely $\bar{z}$. □

Lemma 7. *Any feasible solution to the CBPP that uses $\bar{z}$ bins can be represented as a feasible solution of model* $\mathrm{AF}_{\mathrm{ca}}$ *with objective value $\bar{z}$.*

Proof. Consider a feasible solution to the CBPP with objective value $\bar{z}$. This solution can be described as $\bar{z}$ packing patterns. We must show that any packing pattern P can be represented as a color-alternating path from 0 to L. For this, we select the first item u of P (respecting the given order) and add one unit of flow to the arc corresponding to that item, i.e. the arc of length l_u and color c_u emanating from 0, and repeat the process starting from vertex $0 + l_u$ with sequence $P \setminus \{u\}$. This procedure stops either when the end of the sequence P or vertex L are reached, with the former being completed with a loss-arc emanating from the current vertex and arriving in L.

Since P is a packing pattern, at any vertex $j \in V$ there is exactly one unit of flow arriving in j from an arc of color q and one unit of flow emanating from j from an arc of color $q' \in [Q] \setminus \{q\}$, thus Constraints (2.7) are satisfied to equality for color q in j. This procedure is done for each packing pattern in the solution, and the resulting paths are overlapped to form the solution to model $\mathrm{AF}_{\mathrm{ca}}$. Note that when all paths are overlapped, although Constraints (2.7) might not be satisfied to equality because the same color q' can be used for different incoming colors, they still hold. Lastly, for each feasible pattern, one unit of flow is added only to a set of arcs that connect 0 to L, so Constraints (2.6) are always satisfied. □

With the results of Lemmas 6 and 7, we can finally state the following Theorem.

Theorem 2. *The arc flow formulation* $\mathrm{AF}_{\mathrm{ca}}$ *models the CBPP.*

2.3.3 Comparing AF_{ml} and AF_{ca}

Although they differ in graph representation, both models proposed in this section use the same general idea of arc flow. These types of models typically present very strong lower bounds for bin packing problems. In this section, we develop a correlation between the linear programming relaxation of both proposed models.

First, we remark that model AF_{ml} is similar to the formulation proposed by Valério De Carvalho [27] for the BPP, although modeled on a different graph to incorporate the color constraints into the flow paths. The latter formulation has been shown to provide the same linear programming lower bound as the set-covering model of Gilmore and Gomory [15] for the BPP, which is the basis for the MIRUP conjecture, that states that the optimal value of any instance is within one plus this lower bound rounded up [25].

In our case, the lower bound provided by AF_{ml} linear relaxation is the same as the lower bound provided by the linear relaxation of the set-partition model (analogous to Gilmore and Gomory [15]) in which all color constraints are handled exclusively in the pricing subproblem.

Lemma 8. *Given a feasible solution $(\bar{x}, \bar{z})$ of the linear relaxation of AF_{ml}, it is possible to compute a corresponding solution $(\bar{x}', \bar{z})$ that is feasible for the linear relaxation of the AF_{ca}.*

Proof. In this proof, we denote the set of arcs of the graph corresponding to formulation AF_{ca} by A'.

For all $(i, j, q) \in A'$, let us define

$$\bar{x}'_{ijq} = \begin{cases} \sum\limits_{\substack{((i,q'),(j,q))\in A \\ q'\neq q}} \bar{x}_{iq'jq}, & \text{if } q \leq Q, \\ \sum\limits_{q'\in[Q]} \sum\limits_{((i,q'),(j,q'))\in A} \bar{x}_{iq'jq'}, & \text{otherwise,} \end{cases}$$

where the second case corresponds to the loss-arcs, which only exist for $j = L$. We prove that $(\bar{x}', \bar{z})$ is a feasible solution for AF_{ca}.

Since (2.3) is feasible in $\bar{x}$, for all $u \in \mathcal{I}$ and considering $q = c_u$, we have that

$$\sum_{\substack{(i,j,q')\in A \\ j=i+l_u, q'=c_u}} \bar{x}'_{ijq'} = \sum_{\substack{(i,j,q)\in A \\ j=i+l_u}} \sum_{\substack{q'\neq q: \\ ((i,q'),(j,q))\in A}} \bar{x}_{iq'jq} = \sum_{\substack{((i,q'),(j,q))\in A: \\ j=i+l_u, q'\neq q}} \bar{x}_{iq'jq} = d_u,$$

and, thus, we conclude that Constraints (2.8) are valid for $\bar{x}'$.

Now, we can show that Constraints (2.6) hold, as $\bar{x}$ satisfies Constraints (2.2). In fact,

for all $j \in \{0, \ldots, L\}$, we have that

$$\sum_{(i,j,q)\in A'} \bar{x}'_{ijq} - \sum_{(j,k,q')\in A'} \bar{x}'_{jkq'} = \sum_{((i,q'),(j,q))\in A} \bar{x}_{iq'jq} - \sum_{((j,q),(k,q''))\in A} \bar{x}_{jqkq''}$$
$$= \begin{cases} -z & \text{for } j = 0, q = 0 \\ 0 & j \in \{1, \ldots, L-1\}, q \in [Q] \\ z & j = L, q \in [Q]. \end{cases}$$

Finally, we can derive the validity of Constraints (2.7) as follows. For fixed $j \in \{0, \ldots, L-1\}$ and $q \in [Q]$, we have that

$$\sum_{(i,j,q)\in A'} \bar{x}'_{ijq} - \sum_{\substack{(j,k,q')\in A' \\ q'\in[Q+1]\setminus\{q\}}} \bar{x}'_{jkq'} = \sum_{((i,q'),(j,q))\in A} \bar{x}_{iq'jq} - \sum_{((j,q''),(k,q'))\in A} \bar{x}_{jq''kq'} \tag{2.17}$$
$$\leq \sum_{((i,q'),(j,q))\in A} \bar{x}_{iq'jq} - \sum_{((j,q),(k,q'))\in A} \bar{x}_{jqkq'} \tag{2.18}$$
$$= 0, \tag{2.19}$$

where Equation (2.17) follows from the fact that there are no loss-arcs with $j < L$, Inequality (2.18) holds as $\bar{x} \geq 0$ and Equation (2.19) follows from the flow conservation imposed by Constraint (2.2). □

Lemma 9. *Given a solution $(\bar{x}, \bar{z})$ to the linear programming relaxation of AF_{ca}, it is possible to compute a corresponding solution $(\bar{x}', \bar{z})$ that is feasible for the linear relaxation of AF_{ml}.*

Proof. Consider that we run algorithm DecomposeAF with solution $(\bar{x}, \bar{z})$ and obtain a set of color-alternating patterns $\mathcal{P}$.

Now, for each path $P = ((i_0, i_1, q_1), (i_1, i_2, q_2), \ldots, (i_{k-1}, i_k, q_k))$ with $i_0 = 0$ generated from $\hat{x}$, we define a path P' for AF_{ml} such that

$$P' = (((i_0, q_0), (i_1, q_1)), ((i_1, q_1), (i_2, q_2)), \ldots, ((i_{k-1}, q_{k-1}), (i_k, q')))$$

where $q_0 = 0$ and $q' = q_{k-1}$ if (i_{k-1}, i_k, q_k) is a loss-arc and $q' = q_k$ otherwise. Finally, we add $\check{d}$ units of flow to each arc in P', where $\check{d}$ is the amount of flow passing through the path P in $\hat{x}$, and we obtain a solution $(\bar{x}', \bar{z})$ of AF_{ml} by combining the flow of every path in $\mathcal{P}$.

It is easy to see that, by construction, $\bar{x}'$ is non-negative and satisfies the flow conservation constraints and the demand constraints. □

As a consequence of Lemmas 8 and 9, we can state the following theorem about the strength of the linear relaxation of the proposed models.

Theorem 3. *The lower bounds provided by the linear relaxation of formulations AF_{ml} and AF_{ca} are the same.*

2.3.4 Graph Reduction

Both models presented in this section represent packing patterns as paths in digraphs with at least $L + 1$ vertices. This makes the number of variables (and constraints) of the models grow pseudo-polinomially with the bin capacity. To mitigate this, we use a reduction technique known as *normal patterns* (also known as *canonical dissections*). This technique was proposed independently by Herz [16] and Christofides and Whitlock [9], and it reduces the number of packing points of a bin by showing that there will always exist an optimal solution in which all items are packed in the leftmost position available. The points of a bin in which it is possible to pack an item are defined as $\{x = \sum_{i \in \mathcal{I}} l_i \pi_i : 0 \leq x \leq L, \pi_i \in \{0, 1\}, \text{ for } i \in \mathcal{I}\}$. This set is computed through a dynamic programming algorithm [10] before building the graph. The graph is then created using only vertices that represent points in the normal patterns, thus achieving a more compact graph while still preserving optimality.

Normal patterns is one of the most generic reduction techniques for packing points in a bin. Although there exist other more aggressive reductions, such as meet-in-the-middle proposed by Côté and Iori [10] and reflect proposed by Delorme and Iori [11] (specifically for arc flow), those require extra investigative work to be adapted for the CBPP since the order of the packing matters.

2.4 Exponential Models

In this section, we propose set-partition exponential models for the CBPP that represent all packing patterns implicitly. These models can be solved through branch-and-price or branch-cut-and-price algorithms, and usually present very strong upper bounds [12]. We use the VRPSolver generic scheme, proposed by Pessoa et al. [23, 24] to design such models for the CBPP. With this scheme, we reduce the target problem into a variant of the Vehicle Routing Problem (VRP), which is then solved by a black-box branch-and-cut-and-price algorithm.

In the remainder of this section, we describe how the VRPSolver works in the context of the CBPP, and then describe the two models proposed.

2.4.1 VRPSolver

VRPSolver is a generic branch-cut-and-price solver that incorporates several modern algorithmic improvements such as rank-1 cuts with limited memory, path enumeration, and rounded capacity cuts. It solves a variant of the VRP that is sufficiently general to model many other variants of the VRP or even other problems such as the BPP and the Generalized Assignment Problem. A model for the BPP using this framework was proposed by Pessoa et al. [24], and showed remarkable results, overperforming other modern algorithms designed specifically for the BPP. While we do not go into detail on how the framework manages to generalize the key elements of the branch-cut-and-price algorithm, we strongly recommend the paper by Pessoa et al. [24] to the interested reader.

The main concept of VRPSolver is to provide a formulation that can be solved by a

branch-cut-and-price algorithm in which the pricing problem is modeled as a Resource Constrained Shortest Path problem (RCSP). The variables of the resulting paths are linked to specifically designed objective function and constraints through *mappings*. These constraints and objective function are then added to the original master formulation to be solved by the branch-and-cut-and-price algorithm.

Resource Constrained Shortest Path Problem

An instance of the RCSP is composed of a directed multi-graph $G = (V, A)$, a source vertex $v_{source} \in V$, a sink vertex $v_{sink} \in V$, and a set of resources R. For each arc $a \in A$, there is a cost c_a and, for each resource $r \in R$, there is a consumption value $q_{ar} \in \mathbb{R}$. There is a finitely accumulated resource consumption interval $[l_{ar}, u_{ar}]$ for each resource $r \in R$ and arc $a \in A$. A path $p = (v_{source}, a_1, v_1, \ldots, a_{n-1}, v_{n-1}, a_n, v_{sink})$ with $n \geq 1$ is called a resource constrained path of G if $v_i \in V$, $a_i = (v_{i-1}, v_i) \in A$ (considering $v_0 = v_{source}$ and $v_n = v_{sink}$) and, for each $r \in R$, the accumulated resource consumption at the j-th vertex of p, $S_{j,r} = \max\{l_{a_j,r}, S_{j-1,r} + q_{a_j,r}\}$ and is at most $u_{a_j,r}$ for all $1 \leq j \leq n$, with $S_{0,r} = 0$. Let P be the set of all resource constrained paths of G. The objective is to find a resource constrained path $p \in P$ that minimizes the sum of the costs of the arcs used.

Let p be any resource constrained path of G, h^p is a vector of integer variables that describes which arcs are used in p, so for any arc $a \in A$, h^p_a represents the number of times arc a is used in path p. This concept will be useful for mapping resource constrained paths to variables in the master formulation.

Modeling and Mappings

The VRPsolver's restricted master problem formulation has three sets of variables. The first set of n_1 integer variables are mapped to the arcs of the RCSP graphs. The second set of n_2 integer variables are generic and allow modeling different constraints. Finally, there is a non-negative integer variable λ_p for each path in the set of resource constrained paths P considered in the restricted master problem.

For each variable of the first set, x_j for $j \in [n_1]$, there must exist a *mapping* $M(x_j)$ into a non-empty subset of arcs of the RCSP graph G. While not all arcs of G need to be mapped into a variable of the model, it is still useful to define $M^{-1}(a) = \{j | a \in M(x_j)\}$.

Although the framework allows for multiple graphs to be used together for the same model, we omit it from the formulation for simplicity, since this fact is not relevant in our

case. With this in mind, we can describe the master formulation as follows.

$$GMF = \min \sum_{j=1}^{n_1} c_j x_j + \sum_{s=1}^{n_2} f_s y_s \tag{2.20}$$

$$\text{s.t.} \quad \sum_{j=1}^{n_1} \alpha_{ij} x_j + \sum_{s=1}^{n_2} \beta_{is} y_s \geq d_i \qquad \forall\, 1 \leq i \leq m, \tag{2.21}$$

$$x_j = \sum_{p \in P} \left(\sum_{a \in M(x_j)} h_a^p \right) \lambda_p \qquad \forall\, 1 \leq j \leq n_1, \tag{2.22}$$

$$\mathcal{L} \leq \sum_{p \in P} \lambda_p \leq \mathcal{U}, \tag{2.23}$$

$$\lambda_p \in \mathbb{Z}^+ \qquad \forall p \in P, \tag{2.24}$$

$$x_j, y_s \in \mathbb{Z} \qquad \forall\, 1 \leq j \leq n_1, 1 \leq s \leq n_2, \tag{2.25}$$

in which Equation (2.20) is a general linear objective function with $c \in \mathbb{R}^{n_1}$ and $f \in \mathbb{R}^{n_2}$ being cost vectors, Constraints (2.21) represents m general linear constraints over the first two sets of variables, with $\alpha \in \mathbb{R}^{m \times n_1}$ and $\beta \in \mathbb{R}^{m \times n_2}$ being the coefficient matrices for these variables, and $d \in \mathbb{R}^m$ the right-hand side vector. Constraints (2.22) are the mapping constraints, and ties variables of the first group to arcs in the RCSP graph. Constraints (2.23) define lower bounds $\mathcal{L}$ and upper bounds $\mathcal{U}$ on the number of paths used in the solution. Constraints (2.24)–(2.25) define the domains of variables.

With a specifically designed RCSP graph and the generic master formulation above, we can define specific models for several problems in the literature.

Packing Sets

The VRPSolver framework generalizes many concepts used in state-of-the-art branch-and-cut-and-price algorithms. Such concepts include ng-ranks, path enumeration, limited memory rank-1 cuts, Ryan-Foster branching rule, branching over accumulated resource consumption, and others. This is achieved with the introduction of *Packing Sets*.

Packing Sets of the VRPSolver model are defined as a collection $\mathcal{P}$ of mutually disjoint subsets of arcs A, such that the arcs in each $S \in \mathcal{P}$ appear at most once in all paths that are part of some optimal solution (x^*, y^*, λ^*) of the master problem, that is, for all $S \in \mathcal{P}$, $\sum_{p \in P} \left(\sum_{a \in S} h_a^p \right) \lambda_p^* \leq 1$. We refer to each element in $\mathcal{P}$ as a packing set.

The definition of the Packing Sets does not derive directly from the generic model, thus it is a modeling task to properly define it and show its validity. Furthermore, some features of the algorithm depend on the definition of the packing sets, and since some of these features such as the Ryan-Foster branching rule are necessary for the correctness of the model, the definition of packing sets becomes mandatory. We refer to Pessoa et al. [24] for more information on Packing Sets and how they are used to generalize important algorithmic concepts of this framework.

2.4.2 Color-Resources Partition Model for the CBPP

Consider a graph similar to the one presented by Pessoa et al. [24] for the BPP, that is $G = (V, A)$, with $V = \{v_0, v_1, \ldots, v_{|\mathcal{I}'|}\}$ with $v_{source} = v_0$ and $v_{sink} = v_{|\mathcal{I}'|}$, and $A =$

$\{a_i^+, a_i^- : i \in \mathcal{I}'\}$ where $a_i^+ = (v_{i-1}, v_i)$ and $a_i^- = (v_{i-1}, v_i)$ for $i \in \mathcal{I}'$. Figure 2.5 shows a generic representation of this graph.

As in the model for the BPP consider a resource $\bar{r} \in R$ for the bin capacity L, with $q_{a_i^+,\bar{r}} = l_i$ and $q_{a_i^-,\bar{r}} = 0$ for $i \in \mathcal{I}$ and $l_{v,\bar{r}} = 0$ and $u_{v,\bar{r}} = L$ for $v \in V$. Consider also, for each color $c \in [Q]$, a resource $r_c \in R$ with $q_{a_i^-,r_c} = 0$ and, $q_{a_i^+,r_c} = 1$ if $c_i = c$ or $q_{a_i^+,r_c} = -1$ otherwise. Finally, for each color $c \in [Q]$ and vertex $v \in V$, the bounds on the accumulated resources are $l_{v,r_c} = -|\mathcal{I}'|$, and $u_{v,r_c} = 1$ if $v = v_{sink}$ and $u_{v,r_c} = |\mathcal{I}'|$ otherwise. Like in the BPP model, the packing sets are the singletons composed of the positive arc of each item, that is $\mathcal{P} = \cup_{i \in \mathcal{I}'}\{\{a_i^+\}\}$.

In this model, all color constraints are enforced in the pricing problem through the RCSP graph. Whenever an item of color c is included in a resource constrained path of G, one unit of resource r_c is collected and one unit of resource is dropped for every other $r_{c'}$ such that $c' \neq c$. Through this procedure, we have the color discrepancy for the path at each vertex, so we enforce that such color discrepancy be smaller than or equal to 1 at the sink vertex for all colors. From Theorem 1, we know that this condition is necessary and sufficient to guarantee the existence of a color-alternating permutation of the items in the path. Since all color constraints are modeled in the RCSP, the master formulation can be described with integer variables $x_0, x_1, \ldots, x_{|\mathcal{I}'|}$, where x_0 represents the number of paths used, while x_i indicates whether item i is covered in a path, and mappings $M(x_0) = \{a_1^+, a_1^-\}$, $M(x_i) = \{a_i^+\}$ for $i \in \mathcal{I}'$. The master formulation is given by

$$\min x_0 \tag{2.26}$$

$$\text{s.t.} \quad x_i = 1 \qquad \forall\, i \in \mathcal{I}', \tag{2.27}$$

$$x_i = \sum_{p \in P} \left(\sum_{a \in M(x_i)} h_a^p \right) \lambda_p \qquad \forall\, i \in \mathcal{I}', \tag{2.28}$$

$$\mathcal{L} \leq \sum_{p \in P} \lambda_p \leq \mathcal{U}, \tag{2.29}$$

$$\lambda_p, x_i \in \mathbb{Z}^+ \qquad \forall\, p \in P, \forall i \in \mathcal{I}' \cup \{0\}, \tag{2.30}$$

in which Equation (2.26) takes the place of Equation (2.20), and represents the objective of minimizing the number of paths or packing patterns used. Constraints (2.27) takes the place of the general Constraint (2.21), and enforce that each item appear in exactly one path used in the solution. The remaining constraints, (2.28)–(2.30) are derived directly from general master formulation. For the BPP, we can use $\mathcal{L} = 1$ and $\mathcal{U} = |\mathcal{I}'|$ as trivial bounds on the number of paths.

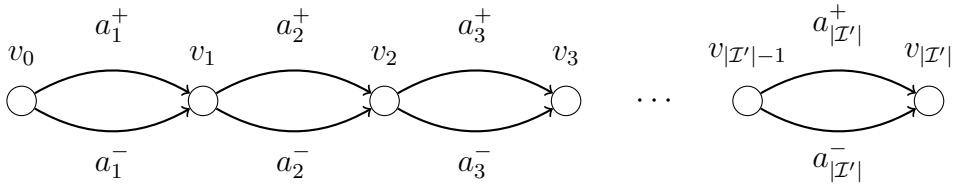

Figure 2.5: RCSP graph for Color-Resources Partition Model.

2.4.3 Colored-Clusters Partition Model for the CBPP

For our second model, we propose a denser graph, but with only one resource constraint. Let $G = (V, A)$ be our colored-clusters graph. The set of vertices is $V = \{v_{source}, v_{sink}\} \cup \{v_i : i \in \mathcal{I}'\} \cup \{v_c^{in}, v_c^{out} : c \in [Q]\}$. And the set of arcs is $A = \{a_i^{source}, a_i^{sink}, a_i^{in}, a_i^{out} : i \in \mathcal{I}'\} \cup \{a_c^{c'} = (v_c^{out}, v_{c'}^{in}) : c, c' \in [Q], c \neq c'\}$ where, for $i \in \mathcal{I}'$, $a_i^{source} = (v_{source}, v_i)$, $a_i^{sink} = (v_i, v_{sink})$, $a_i^{in} = (v_{c_i}^{in}, v_i)$, and $a_i^{out} = (v_i, v_{c_i}^{out})$.

In this graph, there exists one vertex per item as well as one inbound and one outbound vertex for each color. Arcs connect items to their respective inbound colors and outbound colors to all their items. Furthermore, there are arcs connecting each inbound color vertex to every other outbound color vertex. At last, there are arcs connecting the source to each item vertex and from each item vertex to the sink. Hence, all paths in G will alternate between item vertices and color vertices and, since there are only arcs connecting inbound vertices to outbound vertices of different colors, there can be no consecutive items from the same color in a path. Thus, all color constraints are modeled in the RCSP. Figure 2.6 shows a representation of this graph for a simple instance with 2 colors and 3 items of each color.

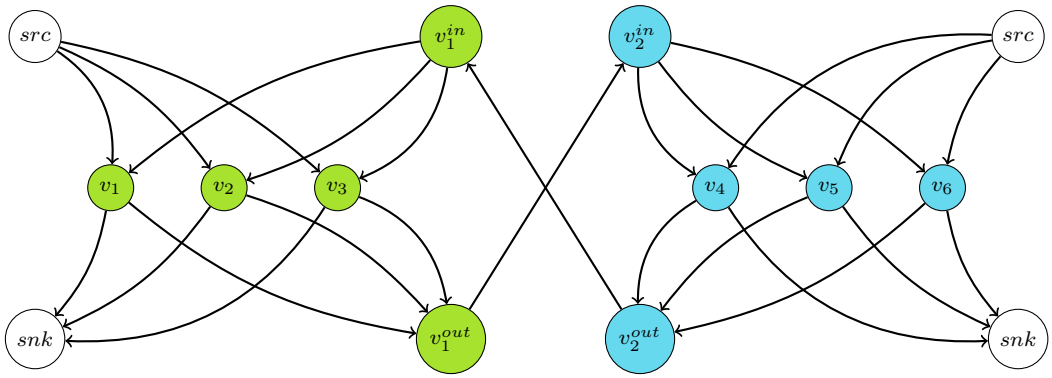

Figure 2.6: RCSP graph for the Colored-Clusters Partition Model for an instance with 2 colors and 3 items of each color. Vertices v_{source} and v_{sink} appear on both sides of the graph as *src* and *snk* respectively to simplify the drawing.

Since there is only one resource for the capacity L, so we omit the resource index in the consumption notation. Thus, we have, $q_{a_i^{source}} = q_{a_i^{in}} = l_i$, and $q_{a_i^{sink}} = q_{a_i^{out}} = 0$, for $i \in \mathcal{I}'$, and $q_{a_c^{c'}} = 0$ for $c, c' \in [Q]$ with $c \neq c'$.

Finally, the bounds on the accumulated resources are $l_v = 0$ and $u_v = L$ for all $v \in V$ and the packing sets are defined as $\mathcal{P} = \cup_{i \in \mathcal{I}'} \{\{a_i^{source}, a_i^{in}\}\}$. Since each item must be included exactly once in an optimal solution, we know that such a solution path will either include an arc from the source to the item vertex v_i or an arc from the inbound color vertex $v_{c_i}^{in}$ to the item vertex v_i for $i \in \mathcal{I}'$.

Since all color constraints are modeled in RCSP, the master formulation can, once again, be described with integer variables $x_0, x_1, \ldots, x_{|\mathcal{I}'|}$, and mappings $M(x_0) = \{a_1^{source}, a_2^{source}, \ldots, a_{|\mathcal{I}'|}^{source}\}$, and $M(x_i) = \{a_i^{source}, a_i^{in}\}$ for $i \in \mathcal{I}'$. With this graph and mappings, we can use Formulation (2.26)–(2.30) to solve the CBPP.

2.5 Numerical Experiments

In this section, we discuss numerical experiments used to evaluate all the proposed models. Since, to the best of our knowledge, there are no benchmarks for the CBPP in the literature, we propose our own set of instances. In Section 2.5.1, we describe our proposed benchmark set, and in Section 2.5.2 we discuss the experimental results.

2.5.1 Benchmark Instances

We propose three different sets of instances, with different levels of complexity. Each set is described below.

Uniform Randomly Generated Instances It is composed of instances with varied item lengths and bin capacities, similar to those proposed by Borges et al. [7] for a class constrained version of the BPP. This group of instances is represented by a tuple (M, L, Q, W). The value of M corresponds to the number of items in the instance, and we consider $M \in \{300, 500\}$. The value of L corresponds to the capacity of bins, and we consider $L \in \{500, 750, 1000\}$. The value of Q corresponds to the number of different colors of items, and we consider $Q \in \{2, 7, 15\}$. The value of W corresponds to the range for the length of items relative to the bin capacity, and we consider $W \in \{[0.1, 0.8], [0.01, 0.25]\}$. We generate 10 instances for each tuple, resulting in 360 total instances.

Randomly Generated Instances with Zipf Distribution of Colors It is composed of randomly generated instances in which the number of items of each color is uneven. In this set, we have instances in which there are few colors with many items and many colors with only a few items. This characteristic is useful to model, for instance, the application of packing content in pages on a social media platform, in which we must alternate advertisements and other types of content while having most adverts come from only a few advertisers. The Zipf distribution [19] has been used before to model data placement applications with such locality properties [30]. Instances in this set are represented by a tuple (M, L). The value of M represents the number of items, and $M \in \{300, 500\}$. The value of L represents the capacity of the bins, and $L \in \{300, 500, 750\}$. The lengths of items are generated randomly with a uniform distribution in the interval of $[0.01, 0.25]$, relative to the bin capacity. All colors are drawn according to a Zipf distribution with $\alpha = 2$. We generate 10 instances for each tuple, totaling 60 instances in this set.

Instances Adapted from BPP Literature It is composed of hard instances adapted from BPPLIB [13]. This group is divided into 94 instances in which the integer round-up property holds (AI), and 60 instances in which the integer round-up property does not hold (ANI). We used only instances from which an optimal solution was known. All optimal solutions used were taken from Delorme and Iori [11]. Given an AI or ANI instance, for each bin in its optimal solution, the items are sorted in a non-increasing order and colored the first half with one color and the second half with a different color. This would

guarantee that the same configuration of items in that optimal solution would be feasible in the CBPP (by Theorem 1), thus maintaining the same optimal solution value.

2.5.2 Computational Results

We have implemented all the proposed models to evaluate their performance. The pseudo-polynomial models of Section 2.3 were implemented in C++11, using Gurobi version 9.03 for solving MILP models. The exponential models of Section 2.4 were implemented in Julia version 1.2, using VRPSolver beta v0.3, and CPLEX version 12.8 for solving the underlying LP models. Experiments were run on Intel (R) Xeon (R) CPU E5–2630 v4 with 10 cores at 2.20GHz, 64 GB of RAM, and running on Ubuntu 18.04.2 LTS with Linux 4.15.0–45–generic.

For the pseudo-polynomial models, we have implemented a simple FF-like heuristic to obtain a simple initial feasible solution. For the exponential models, VRPSolver has a diving heuristic that is run periodically during the search in order to find feasible solutions. None of the models were provided with cutoff values to reduce the solution space. All models were solved with the single-thread mode of their respective solvers. We consider a time limit of 1800 seconds for each instance.

Results on Uniform Randomly Generated Instances

We start by evaluating the proposed models on the set of Uniform Randomly Generated Instances. This is the largest of the proposed sets and covers a wide variety of cases. Since, for the CBPP, the color constraints are enforced on the adjacency of items, it makes sense to investigate instances in which many items can be packed in the same bin. For this reason, half of the instances in this set contain only items with lengths of at most one-quarter of the bin capacity, thus fitting at least four items in any maximal packing pattern. This set also tests the relevance of having many different colors, since the number of colors ranges from 2 to 15.

Tables 2.1 and 2.2 show the results for all models on the first instance set. In Table 2.1 we see the results for instances with item length relative range $W = [0.1, 0.8]$, while Table 2.2 has the results of all instances with item length relative range $W = [0.01, 0.25]$.

The color-alternating arc flow model of Section 2.3.2 (hereafter referred to as CA-AF) shows good results for this instance set, optimally solving 231 out of the 360 instances (65%) with a relatively low average gap on the classes it does not solve completely. All instances with $W = [0.1, 0.8]$ are solved to optimality in under a minute on average. However, the results are not as impressive when it comes to instances with $W = [0.01, 0.25]$. Only 51 out of 180 (28%) of these instances are solved to optimality, with average times coming closer to the time limit even on classes where half of the instances are solved optimally. This discrepancy can be explained by the fact that the number of arcs in the graph increases dramatically with the number of items that can be packed in a single bin, resulting in a model with many more variables in cases with $W = [0.01, 0.25]$. The largest instance in this subset to be solved to optimality by this model has 500 items, 500 of bin capacity, and 15 colors.

			CA-AF			ML-AF			CC-partition			CR-partition		
M	L	Q	opt	time	gap	opt	time	gap	opt	time	gap	opt	time	gap
300	500	2	**10**	3	0	**10**	4	0	**10**	95	0	**10**	42	0
300	500	7	**10**	9	0	**10**	30	0	**10**	146	0	**10**	761	0
300	500	15	**10**	8	0	**10**	105	0	9	341	–	**10**	1118	0
300	750	2	**10**	9	0	**10**	6	0	**10**	85	0	**10**	82	0
300	750	7	**10**	9	0	**10**	60	0	8	589	–	**10**	712	0
300	750	15	**10**	16	0	**10**	235	0	9	419	–	**10**	1176	0
300	1000	2	**10**	8	0	**10**	8	0	**10**	205	0	**10**	73	0
300	1000	7	**10**	16	0	**10**	81	0	**10**	251	0	**10**	982	0
300	1000	15	**10**	20	0	**10**	319	0	9	427	0.001	**10**	1243	0
500	500	2	**10**	16	0	**10**	10	0	**10**	323	0	**10**	193	0
500	500	7	**10**	13	0	**10**	99	0	6	1053	0.002	**10**	1204	0
500	500	15	**10**	20	0	**10**	234	0	6	1167	–	8	1460	0.001
500	750	2	**10**	23	0	**10**	18	0	**10**	593	0	**10**	262	0
500	750	7	**10**	45	0	**10**	140	0	8	938	–	9	1400	0
500	750	15	**10**	44	0	**10**	552	0	7	1166	–	7	1721	–
500	1000	2	**10**	25	0	**10**	19	0	**10**	573	0	**10**	418	0
500	1000	7	**10**	43	0	**10**	207	0	7	1060	–	9	1530	–
500	1000	15	**10**	54	0	9	947	0.1	9	555	–	3	1770	–
	Average		**10**	21.1	0	9.9	170.7	0.006	8.7	554.7	–	9.2	897.1	–

Table 2.1: Results for all models on Uniform Randomly Generated Instances set with $W = [0.1, 0.8]$. Columns M, L, Q represent the number of items in the instance, the bin capacity, and the number of colors respectively. For each of the proposed models, we have a column *opt* showing the number of instances of that class (out of 10) that were solved to proven optimality, *time* which represents the average time in seconds, and *gap* which represents the average of the optimality gap computed as $(UB - LB)/UB$.

Comparatively, the multilayered arc flow model of Section 2.3.1 (hereafter referred to as ML-AF) shows milder results. Only 218 out of the 360 instances (60%) are solved optimally while failing to even find a lower bound before the time limit in several of the instances it does not solve. When considering only instances with $W = [0.1, 0.8]$, all but one are solved to optimality in a few hundred seconds on average. On the other hand, for instances with $W = [0.01, 0.25]$, the model was able to solve only 39 out of 180 (21%). Also, for this subset, the model is only able to reliably find lower bounds for instances with 2 colors. This can be explained by the fact that the number of vertices in the graph, for this model, grows by an order of M for every color considered, making it difficult to even solve the root node of the MILP for instances with a large enough number of vertices and colors. The largest instance in this subset to be solved to optimality by this model has 500 items, 750 of bin capacity, and 2 colors. Since this model has fewer constraints than the previous one, it may outperform CA-AF in specific cases where the number of colors is small enough, which can be observed in instances with $Q = 2$.

For the colored-clusters partition model of Section 2.4.3 (hereafter referred to as CC-partition) we have the worst results out of the models tested in this set. A total of 158 out of 360 instances (43%) were optimally solved, with none of them being from the subset with $W = [0.01, 0.25]$. Even for the subset with $W = [0.1, 0.8]$, it was not able to find a feasible solution for many cases, resulting in the impossibility of even computing an optimality gap (shown as "–" in the tables). Since this is a pattern-based model, the inability to solve instances with $W = [0.01, 0.25]$ can be explained by the fact that the number of possible patterns considered is comparatively higher for these cases. Furthermore, each pattern is generated as a resource constrained path, which also grows

			CA-AF			ML-AF			CC-partition			CR-partition		
M	L	Q	opt	time	gap	opt	time	gap	opt	time	gap	opt	time	gap
300	500	2	9	576	0.053	**10**	733	0	0	1982	–	**10**	130	0
300	500	7	**4**	1373	0.045	0	1801	0.508	0	1999	–	0	1023	–
300	500	15	**7**	1117	0.039	0	1803	1	0	1934	–	0	1024	–
300	750	2	8	1188	0.096	7	970	0.007	0	1991	–	**10**	195	0
300	750	7	**1**	1698	0.096	0	1802	1	0	2012	–	0	1034	–
300	750	15	**0**	1801	0.16	**0**	1804	1	**0**	1920	–	**0**	1037	–
300	1000	2	3	1557	0.304	9	964	0.005	0	1918	–	**10**	384	0
300	1000	7	**1**	1791	0.234	0	1803	1	0	2033	–	0	1045	–
300	1000	15	**2**	1732	0.223	0	1806	1	0	1901	–	0	1048	–
500	500	2	5	1387	0.219	8	926	0.003	0	1847	–	**10**	462	0
500	500	7	**6**	1457	0.056	0	1802	1	0	1862	–	0	1069	–
500	500	15	**5**	1535	0.042	0	1805	1	0	1835	–	0	1071	–
500	750	2	0	1801	0.409	5	1486	0.012	0	1826	–	**10**	523	0
500	750	7	**0**	1801	0.307	**0**	1803	1	**0**	1851	–	**0**	1094	–
500	750	15	**0**	1802	0.282	**0**	1808	1	**0**	1821	–	**0**	1101	–
500	1000	2	0	1801	0.46	0	1801	0.046	0	1829	–	**9**	834	–
500	1000	7	**0**	1802	0.316	**0**	1805	1	**0**	1835	–	**0**	1134	–
500	1000	15	**0**	1803	0.314	**0**	1810	1	**0**	1821	–	**0**	1138	–
	Average		2.8	1556.7	0.203	2.1	1585.1	0.643	0	1900.9	–	**3.27**	852.5	–

Table 2.2: Results for all models on Uniform Randomly Generated Instances set with $W = [0.01, 0.25]$. Captions in this table should be interpreted as in Table 2.1.

with the number of items that can be accommodated in the same bin, making the column generation phase more difficult in these cases.

On the other hand, the color-resources partition model of Section 2.4.2 (CR-partition) has shown to be very competitive. It was able to solve 225 out of 360 instances (62%) to optimality, although with a considerably higher average time. In the subset with $W = [0.1, 0.8]$, 166 out of 180 were optimally solved while failing to find upper bounds for some of the larger instances. Similarly to ML-AF, this model seems to perform its best in instances with only a few colors, as observed in cases with $Q = 2$. In fact, for the subset with $W = [0.01, 0.25]$, all but one of the instances with $Q = 2$ were solved to optimality. For instances with $Q > 2$ in this subset, the model was interrupted before the time limit because of memory constraints, so it was not able to produce an optimality gap. The reason for such problems may be because the color constraints are enforced as resource constraints in the column generation phase, instead of being represented as special vertices in the path like in CC-partition. Though this may result in a relative performance gain for a smaller number of colors, models with a higher number of resource constraints tend to be significantly more difficult. Furthermore, since branching on accumulated resource consumption is used, this may result in a very large branch-and-bound tree, possibly causing memory issues for large enough instances. The largest instance in this subset to be optimally solved has 500 items, 1000 of bin capacity, and 2 colors.

Results on Randomly Generated Instances with Zipf Distribution of Colors

We evaluate the proposed models on the set of Randomly Generated Instances with Zipf Distribution of Colors. In this relatively smaller instance set, we investigate the impact of the color constraints in the models by presenting cases with an unbalanced number of items for each color as well as a high number of total colors (up to 37). In this set, all instances have lengths in the interval of $[0.01, 0.25]$ relative to the bin capacity, so

maximal packings tend to contain several items. Since many items can be of the same color, it can be extremely difficult to find efficient packings.

		CA-AF			ML-AF		
M	L	opt	time	gap	opt	time	gap
300	300	**8**	509	0.049	3	1732	0.700
300	500	**9**	539	0.060	0	1805	1
300	750	**9**	897	0.002	0	1807	1
500	300	**10**	176	0	0	1806	0.941
500	500	**8**	1022	0.111	0	1810	1
500	750	**4**	1665	0.291	0	1816	1
Average		**8**	801.3	0.085	0.5	1796	0.94

Table 2.3: Results for arc flow models on Randomly Generated Instances with Zipf Distribution of Colors set. Column M represents the number of items in the instance, and L is the bin capacity considered. Each model has a set of columns *opt*, *time*, and *gap* meaning, respectively, amount of instances to be solved optimally (out of 10), average time in seconds, and average gap computed as $(UB - LB)/UB$.

Table 2.3 shows a summary of the results for this set. Following a trend observed in the previous instance set, model CC-partition failed to solve any of the instances in this set, so we decided to omit it from the reports. Due to the high number of colors considered in all instances of this set, model CR-partition failed to even load most of them due to memory constraints, so this model is also omitted from the reports.

Model CA-AF was able to optimally solve 48 out of 60 instances (80%) in this set, while ML-AF was only able to solve 3 to optimality (0.05%). This was expected since we knew from previous analysis that ML-AF could only surpass CA-AF in cases with a very small number of colors. We can also notice that despite the significantly larger number of colors, CA-AF was faster on average and solved more instances when compared to similar instances in the Uniform Randomly Generated Instances set. This shows that, unlike any of the other proposed models, CA-AF is reasonably efficient in dealing with high-degree color constraints.

Results on Instances Adapted from BPP Literature

We evaluate the proposed models on the set of Instances Adapted from BPP Literature. In this set, we adapt difficult instances from the BPP literature, namely the AI and ANI instances, to instances of the CBPP. These instances have specific properties to their lower bounds while solved by a classic set-cover formulation, so to maintain these properties, we guarantee that the optimal solution value is maintained when adapting them to the CBPP. Additionally, we consider only two colors in the adaptation, so all models can be competitive. For more details on the original instances and their properties, we refer to Delorme et al. [13]. Table 2.4 shows the results for all models in this instance set.

Model CA-AF was only able to solve the smaller instances of each set, being 42 out of 94 (44%) from AI and 39 out of 60 (65%) from ANI. The model was unable to even

		CA-AF			ML-AF			CC-partition			CR-partition		
instance	#	opt	time	gap	opt	time	gap	opt	time	gap	opt	time	gap
201_2500_AI	50	42	951	0.003	39	1099	0.023	45	392	0.002	**46**	386	0.001
402_10000_AI	36	0	1805	1	0	1804	1	25	994	–	**32**	1105	–
600_20000_AI	8	0	1816	1	0	1813	1	**2**	1759	–	0	2271	–
201_2500_ANI	49	39	846	0.005	44	791	0.042	41	784	–	**48**	471	0
402_10000_ANI	11	0	1804	1	0	1804	1	**1**	1772	–	0	1810	–
Average		16.2	1444.4	0.601	16.6	1462.2	0.613	22.8	1140.2	–	**25.2**	1208.6	–

Table 2.4: Results for all models on Instances Adapted from BPP Literature. Column *instance* describes the original instance class, with the first number being an approximation of the number of items, and the second number an approximation of the bin capacity. Column # shows the number of instances for each class. Each model has a set of columns *opt*, *time*, and *gap* meaning, respectively, amount of instances to be solved optimally, the average time in seconds, and the average gap computed as $(UB - LB)/UB$.

compute a lower bound for any instance of size greater than 200. This was expected since the AI and ANI sets are very difficult for simpler arc flow models. Advanced reduction techniques are often required to deal with such instances in the BPP, such as the reflect model of Delorme and Iori [11] or the meet-in-the-middle model of Côté and Iori [10]. Because of the nature of the color constraints, such techniques cannot be trivially adapted to the CBPP.

Model ML-AF does slightly better than CA-AF, solving 39 out of 94 instances (41%) from AI and 44 out of 60 (73%) from ANI. Similarly to the previous model, only instances of size 200 were solved. This result was expected, due to ML-AF outperforming CA-AF in cases with only a few colors in previous experiments. Since this set only has instances with 2 colors, ML-AF presents a smaller number of constraints when compared to CA-AF, which may explain the slightly better results.

We expected the exponential models to perform very well in this set, due to previous results of VRPSolver BPP models [24] even surpassing those of Delorme and Iori [11]. This was confirmed, with both partition models performing similarly well. CC-partition solved 72 out of 94 instances (76%) from AI and 42 out of 60 (70%) from ANI, while CR-partition solved 78 out of 94 instances (82%) from AI and 48 out of 60 (80%) from ANI. Interestingly, although model CR-partition was able to solve more instances in total, model CC-partition was the only one to solve 2 instances from AI with size 600 and one instance from ANI with size 400. This shows us that both these models have a high potential for solving difficult instances for cases when the number of colors is 2.

2.6 Concluding Remarks

In this work, we study the Colored Bin Packing Problem under an exact approach. We start by proposing a generalization of the arc flow model of Valério De Carvalho [27], using multiple layers to represent different colors (ML-AF). Then, we propose a novel arc flow formulation with color-alternating constraints (CA-AF) and demonstrate that the additional constraints are sufficient to model the CBPP without having to use one layer per color. Furthermore, we prove that the two arc flow models are equivalent in terms

of linear relaxation strength. We also present two set-partition models (CC-partition and CR-partition) to be solved by a branch-cut-and-price algorithm through a reduction to a general Vehicle Routing Problem with VRPSolver. A diverse benchmark set is also proposed to evaluate the models.

Experimental results comparing all the proposed models are presented. Among the ones considered, the CA-AF can solve the most instances, the largest of which has 500 items and 37 colors. However, when only two colors are considered, the set-partition models are shown to outperform the other models.

It would be interesting to investigate, in the future, the use of the characterization of feasible solutions given in Theorem 1 to design efficient heuristics and meta-heuristics for the CBPP. This characterization can also pave the way for new approximation algorithms for the CBPP or even approximation schemes for a colored variant of the knapsack problem. Good algorithms for the colored variant of the knapsack problem may lead to a good branch-and-price algorithm to solve the set-partition model based on the formulation of Gilmore and Gomory [15].

Acknowledgments

This project was supported by the São Paulo Research Foundation (FAPESP) grants #2015/11937–9 and #2022/05803–3, and the Brazilian National Council for Scientific and Technological Development (CNPq) grants #144257/2019–0, #311039/2020–0 and #313146/2022–5. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] J. Balogh, J. Békési, G. Dosa, H. Kellerer, and Z. Tuza. Black and white bin packing. In *Proceedings of the 10th International Workshop on Approximation and Online Algorithms*, pages 131–144, 2013. doi: 10.1007/978-3-642-38016-7_12.

[2] J. Balogh, J. Békési, G. Dósa, L. Epstein, H. Kellerer, and Z. Tuza. Online results for black and white bin packing. *Theory of Computing Systems*, 56(1):137–155, 2015. doi: 10.1007/s00224-014-9538-8.

[3] János Balogh, József Békési, György Dósa, Leah Epstein, Hans Kellerer, Asaf Levin, and Zsolt Tuza. Offline black and white bin packing. *Theoretical Computer Science*, 596:92–101, sep 2015. ISSN 03043975. doi: 10.1016/j.tcs.2015.06.045.

[4] Vittorio Bilò, Francesco Cellinese, Giovanna Melideo, and Gianpiero Monaco. On colorful bin packing games. In *Proceedings of the 24th International Computing and Combinatorics Conference, COCOON*, pages 280–292, 2018.

[5] Vittorio Bilò, Francesco Cellinese, Giovanna Melideo, and Gianpiero Monaco. Selfish colorful bin packing games. *Journal of Combinatorial Optimization*, 40(3):610–635, 2020.

[6] Martin Böhm, György Dósa, Leah Epstein, Jiří Sgall, and Pavel Veselý. Colored bin packing: Online algorithms and lower bounds. *Algorithmica*, 80(1):155–184, 2018.

[7] Yulle G. F. Borges, Flavio K. Miyazawa, Rafael C. S. Schouery, and Eduardo C. Xavier. Exact algorithms for class-constrained packing problems. *Computers & Industrial Engineering*, 144:106455, 2020.

[8] Jing Chen, Xin Han, Wolfgang Bein, and Hing-Fung Ting. Black and white bin packing revisited. In *Proceedings of the 9th Conference on Combinatorial Optimization and Applications, COCOA*, pages 45–59, 2015.

[9] Nicos Christofides and Charles Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44, 1977.

[10] Jean-François Côté and Manuel Iori. The meet-in-the-middle principle for cutting and packing problems. *INFORMS Journal on Computing*, 30(4):646–661, 2018.

[11] Maxence Delorme and Manuel Iori. Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS Journal on Computing*, 32(1): 101–119, 2020.

[12] Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

[13] Maxence Delorme, Manuel Iori, and Silvano Martello. BPPLIB: a library for bin packing and cutting stock problems. *Optimization Letters*, 12(2):235–250, 2018.

[14] G. Dósa and L. Epstein. Colorful bin packing. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory*, pages 170–181, 2014. doi: 10.1007/978-3-319-08404-6_15.

[15] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Operations research*, 9:849–859, 1961.

[16] J. C. Herz. Recursive computational procedure for two-dimensional stock cutting. *IBM Journal of Research and Development*, 16(5):462–469, 1972.

[17] Klaus Jansen. An approximation scheme for bin packing with conflicts. *Journal of combinatorial optimization*, 3(4):363–377, 1999.

[18] Klaus Jansen and Sabine Öhring. Approximation algorithms for time constrained scheduling. *Information and computation*, 132(2):85–108, 1997.

[19] Donald E. Knuth. *The art of computer programming: Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

[20] Arthur Kramer, Manuel Iori, and Philippe Lacomme. Mathematical formulations for scheduling jobs on identical parallel machines with family setup times and total weighted completion time minimization. *European Journal of Operational Research*, 289(3):825–840, 2021.

[21] Prabhu Manyem, Rhonda L Salt, and Marc Simon Visser. Approximation lower bounds in online LIB bin packing and covering. *Journal of Automata, Languages and Combinatorics*, 8(4):663–674, 2003.

[22] M. Peeters and Z. Degraeve. The co-printing problem: A packing problem with a color constraint. *Operations Research*, 52(4):623–638, 2004.

[23] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 354–369, 2019.

[24] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183:483–523, 2020. doi: 10.1007/s10107-020-01523-z.

[25] G. Scheithauer and J. Terno. The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, 84 (3):562–571, 1995. doi: 10.1016/0377-2217(95)00022-I.

[26] H. Shachnai and T. Tamir. Polynomial time approximation schemes for class-constrained packing problems. *Journal of Scheduling*, 4(6):313–338, 2001.

[27] J. M. Valério De Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659, 1999.

[28] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, 3(2):111–130, 1994. doi: 10.1007/BF01300970.

[29] Laurence A. Wolsey. Valid inequalities, covering problems and discrete dynamic programs. In *Annals of Discrete Mathematics*, volume 1, pages 527–538. 1977.

[30] E. C. Xavier and F. K. Miyazawa. The class constrained bin packing problem with applications to video-on-demand. *Theoretical Computer Science*, 393:240–259, 2008.

Chapter 3

Algorithms for the Bin Packing Problem with Scenarios

Abstract This paper presents theoretical and practical results for the bin packing problem with scenarios, a generalization of the classical bin packing problem which considers the presence of uncertain scenarios, of which only one is realized. For this problem, we propose an absolute approximation algorithm whose ratio is bounded by the square root of the number of scenarios times the approximation ratio for an algorithm for the vector bin packing problem. We also show how an asymptotic polynomial-time approximation scheme is derived when the number of scenarios is a constant. As a practical study of the problem, we present a branch-and-price algorithm to solve an exponential model and a variable neighborhood search heuristic. To speed up the convergence of the exact algorithm, we also consider lower bounds based on dual feasible functions. Results of these algorithms show the competence of the branch-and-price in obtaining optimal solutions for about 59% of the instances considered, while the combined heuristic and branch-and-price optimally solved 62% of the instances considered.

Keywords Bin Packing Problem; Scenarios; Approximation Algorithm; Variable Neighborhood Search; Branch-and-Price Algorithm;

3.1 Introduction

Cutting and packing problems have been widely studied in the context of Operations Research, mainly because of their properties and real-world applicability [51]. Among these problems, the *Bin Packing Problem* (BPP) asks to pack a set of one-dimensional items, each of a given size, into the least possible number of bins of given identical capacities, where the total size of items in a bin does not exceed the bin capacity. Several variants of this problem have been studied in the literature, either due to their theoretical interest, or their high applicability in different industries [48, 19].

The BPP and its variants have been extensively investigated since the thirties [40]. Such problems are among the most studied in approximation contexts. A recent survey by Coffman et al. [18] presented over 180 references related to approximation results

for the BPP and its variants. In particular, the BPP is shown to be APX-hard, and indeed no algorithm has an approximation factor smaller than 3/2, unless P = NP [50]. Regarding practical techniques for the BPP, a survey by Delorme et al. [25] reviewed models and solution methods, and experimentally compared the available software tools to solve the problem. Recent practical contributions for the BPP include the generalized arc-flow formulation by Brandão and Pedroso [11], the cooperative parallel genetic algorithm by Kucukyilmaz and Kiziloz [41], the reflect formulation by Delorme and Iori [24], the branch-and-cut-and-price algorithm by Wei et al. [52] and the framework of de Lima et al. [21].

Several variants of the BPP have also been extensively investigated in the literature. For example, in the variant with fragile objects, the capacity of a bin is directly influenced by its most fragile item [17]. In the BPP with precedence constraints, a precedent item cannot be packed in a bin later than its successors [43]. The variant with overlapping items assumes that some subsets of items, when packed in the same bin, occupy less capacity than the sum of their individual size [35]. In the BPP with item fragmentation, items can be split and fragmentally packed [7]. In the generalized BPP, bins may have different cost and capacity and items are associated to a profit and are divided by compulsory (i.e., mandatory to load) and non-compulsory. The aim is to minimize the overall cost based on the bins cost and items profit [3]. In the online BPP, items arrive sequentially and each must be packed before the arrival of the next one, with no information about the next items [4]. The temporal variant considers one additional dimension in the BPP, and the bin capacity should not be violated at any unit of a discretized time horizon [22].

One of the current challenges to solving practical logistic problems is to deal with the uncertainty arising from real-world applications [47, 55, 39, 2]. In particular, one popular way to deal with this issue is by describing scenarios. A scenario is defined as a possible outcome that may arise depending on the problem's uncertain variables. Some authors use scenarios to consider the problem description as combinatorial rather than stochastic, as, e.g., Feuerstein et al. [29]. Considering the scarcity of works that apply this concept of scenarios to deal with uncertainty in bin packing problems and the existence of practical applications, Bódis and Balogh [9] recently introduced the *Bin Packing Problem with Scenarios* (BPPS).

The BPPS is a generalization of the BPP where each item belongs to a subset of scenarios, and a packing must respect the capacity constraints of the bins for each scenario individually. Whilst the set of scenarios is known in advance, only one of the scenarios will be realized. This introduces the possibility of packing in a single bin items whose combined sizes surpasses the bin capacity, as long as the bin capacity is respected in each individual scenario. Although Bódis and Balogh [9] introduced three objective functions for the BPPS, we are concerned with the objective of minimizing the number of bins of the worst-case scenario, which is the most challenging one. In this way, the BPP is the particular case of the BPPS where there is a single scenario.

A generalization of the BPP that is similar to the BPPS is the *Vector Bin Packing Problem* (VBPP), where bins and items have multiple dimensions and the bin capacity must be respected in all of its dimensions. The objective is the same as in the BPP, to minimize the number of bins. Caprara and Toth [13] presented lower bounds, heuristics,

and a branch-and-bound (B&B) algorithm for the VBPP with two dimensions. Alves et al. [1] investigated dual feasible functions for this problem. Buljubašić and Vasquez [12] proposed a local search algorithms where a Tabu search and descent search with add and drop moves were used to explore the search space. In Hu et al. [38], a set-covering model is solved by a branch-and-price (B&P) algorithm. In Heßler et al. [37], there is a stabilized branch-and-price algorithm with dual optimal cuts. Recently, Wei et al. [53] developed a branch-and-price algorithm for the vector packing with two dimensions, where a goal cut approach is used to obtain lower bounds and a dynamic programming with branch-and-bound solves the pricing problem, improving previous results of the literature.

To the best of our knowledge, the recent work by Bódis and Balogh [9] is the only one in the literature concerned with the BPPS. Motivated by its interesting theoretical aspects, its general applicability and its relation to well-studied problems, this paper proposes theoretical and practical results for the BPPS. The contributions of the current work are the following: (i) an absolute approximation algorithm for the BPPS based on the VBPP; (ii) an *asymptotic polynomial time approximation scheme* (APTAS) for the version of the problem with a constant number of scenarios; (iii) an exact method for the problem based on a branch-and-price algorithm for an exponential *Integer Linear Programming* (ILP) model; and (iv) a *Variable Neighborhood Search* (VNS) heuristic.

This paper is organized as follows. In Section 3.2 we provide formal definitions to be used to contextualize the contributions of the paper. In Section 3.3, we show that an approximate solution of the BPPS can be obtained by solving an instance of the VBPP, leading to an absolute approximation algorithm for the BPPS. In Section 3.4, we present an APTAS for the BPPS, when the number of scenarios is a constant. As for practical results, Section 3.5 describes a VNS algorithm for the BPPS and Section 3.6 presents a branch-and-price algorithm to solve an exponential model for the BPPS. The evaluation of the proposed VNS and branch-and-price algorithms is performed in Section 3.7, from experiments based on the solution of randomly generated instances. Finally, Section 3.8 presents the conclusions and directions for future research.

3.2 Formal Definitions

In this section, we provide some definitions to contextualize the contributions of this work. For simplicity, throughout the paper, we denote, for any natural n, the set $\{1, \ldots, n\}$ simply by $[n]$.

Bin Packing Problem with Scenarios (BPPS) In the BPPS, we are given a number d of scenarios, a set $\mathcal{I} = [n]$ of n items, with each $i \in \mathcal{I}$ having a size $s_i \in \mathbb{Q}_+$ and a set of scenarios $\mathcal{K}_i \subseteq [d]$, and an unlimited number of identical bins of capacity $W \in \mathbb{Q}_+$. For each $k \in [d]$, the set of items in scenario k is denoted by $S_k = \{i \in \mathcal{I} \mid k \in \mathcal{K}_i\}$. A solution $\mathcal{B}$ of the BPPS is a partition of $\mathcal{I}$ such that for each part $B \in \mathcal{B}$ and for each scenario $k \in [d]$, $\sum_{i \in B \cap S_k} s_i \leq W$. The objective of the BPPS is to find a solution that minimizes

$$V_{BPPS}(\mathcal{B}) = \max_{k \in [d]} |\{B \in \mathcal{B} : B \cap S_k \neq \emptyset\}|.$$

The objective corresponds to minimize the number of bins of the worst-case scenario, i.e., the scenario with the largest number of bins. A closely related problem is the VBPP.

Vector Bin Packing Problem (VBPP) In the VBPP, we are given a number d of resources, a set $\mathcal{I} = [n]$ of n items, each $i \in \mathcal{I}$ having a vector $s_i \in \mathbb{Q}_+^d$ of resource consumption, and an unlimited number of identical bins of resource capacity given as a d-dimensional vector $W = (w_1, w_2, \ldots, w_d) \in \mathbb{Q}_+^d$. A solution $\mathcal{B}$ is a partition of $\mathcal{I}$ such that for each part $B \in \mathcal{B}$ and each resource $k \in [d]$, $\sum_{i \in B} s_{ik} \leq w_k$. The objective is to find a solution that minimizes

$$V_{VBPP}(\mathcal{B}) = |\mathcal{B}|.$$

The VBPP is a generalization of the BPP, and thus it is also APX-hard. Woeginger [54] showed that there is no APTAS for the two-dimensional version of this problem unless P = NP. Chekuri and Khanna [14] gave an asymptotic $O(\ln d)$-approximation for the case where d is a fixed constant. Later on, this result was improved to $1 + \ln d$ by Bansal et al. [5]. Also, as noted by Christensen et al. [15], the APTAS of Fernandez de la Vega and Lueker [28] implies a $(d + \varepsilon)$-approximation for the VBPP.

For the theoretical results of this paper in Sections 3.3 and 3.4, we assume, without loss of generality, that the instances of the BPPS and the VBPP are normalized by a proper scaling of the items, so that the capacity of the bins corresponds to 1 on each dimension.

An important concept in both theoretical and practical results for the BPP and related variants is the one of *cutting pattern*: a feasible combination of items in a single bin. For the BPPS, we represent a cutting pattern as a vector $p = (a_{1p}, \ldots, a_{np}, b_{1p}, \ldots, b_{dp})$ of binary coefficients, such that: each coefficient a_{ip} is equal to 1 if and only if item i is in pattern p, and each coefficient b_{kp} is equal to 1 if and only if some item of the scenario k is in pattern p. We denote by $\mathcal{P}$ the set of all feasible patterns for the BPPS.

3.3 An Absolute Approximation Algorithm

In the following, we consider a mapping between the set of instances of the BPPS and a subset of instances of the VBPP. Starting with a normalized instance $I = (d, \mathcal{I}, \mathcal{K}, s)$ of the BPPS, we construct a normalized instance $I' = (d, \mathcal{I}, s')$ of the VBPP. For that, let, for each $i \in \mathcal{I}$ and $k \in [d]$,

$$s_{ik} = \begin{cases} s_i & \text{if } i \in S_k, \\ 0 & \text{otherwise.} \end{cases}$$

It is simple to verify that a solution $\mathcal{B}$ for I is a solution for I', and vice-versa.

Definition 2. *A solution $\mathcal{B}$ is minimal for the BPPS if, for any two bins $A, B \in \mathcal{B}$, there exists a scenario k, such that*

$$\sum_{i \in A \cap S_k} s_i + \sum_{i \in B \cap S_k} s_i > 1.$$

Notice that the value of $\mathcal{B}$ for the BPPS is not larger than the value of $\mathcal{B}$ for the VBPP. In the opposite direction, we have Theorem 4.

Theorem 4. *If $\mathcal{B}$ is a minimal solution for the BPPS, then $V_{VBPP}(\mathcal{B}) \leq \sqrt{d}\, V_{BPPS}(\mathcal{B})$.*

Proof. We say that two bins A and B are incompatible in a scenario k if both bins, A and B, are non-empty in scenario k. Let G be a multigraph where each bin represents a vertex, and for each pair of bins A and B and scenario k, there is one edge between A and B (labeled as k) if they are incompatible in k.

For each scenario $k \in [d]$, we define $h_k = |\{B \in \mathcal{B} : B \cap S_k \neq \emptyset\}|$, i.e., h_k is the number of bins used by $\mathcal{B}$ in k. Let $h = \max_{k\in[d]} h_k$, and notice that, since, for each scenario, there is an edge for each two used bins, the number of edges in G is

$$|E(G)| = \sum_{k=1}^{d} \frac{h_k(h_k-1)}{2} \leq d\frac{h(h-1)}{2}.$$

Let r be the number of bins. Since $\mathcal{B}$ is minimal, any two bins are incompatible in at least one scenario and G must have at least one edge between any two vertices, i.e., $|E(G)| \geq \frac{r(r-1)}{2}$. Thus,

$$\frac{r(r-1)}{2} \leq d\frac{h(h-1)}{2},$$

and, since $r \leq dh$, as each bin is non-empty in at least one scenario, we have that

$$r^2 \leq dh^2 + r - dh \leq dh^2.$$

The result follows as $r = V_{VBPP}(\mathcal{B})$ and $h = V_{BPPS}(\mathcal{B})$. □

Corollary 2. *Suppose there exists an α-approximation for the VBPP, then there exists an $\alpha\sqrt{d}$-approximation for the BPPS.*

Proof. Consider an instance I of BPPS and the instance I' of VBPP obtained as described above. Also, let $\mathcal{B}$ be the solution found by the α-approximation for instance I', $\mathcal{B}^*_V$ be an optimal solution for VBPP with instance I' and $\mathcal{B}^*_S$ be a minimal optimal solution for BPPS with instance I. Then, we have

$$V_{BPPS}(\mathcal{B}) \leq V_{VBPP}(\mathcal{B}) \leq \alpha\, V_{VBPP}(\mathcal{B}^*_V) \leq \alpha\, V_{VBPP}(\mathcal{B}^*_S) \leq \alpha\sqrt{d}\, V_{BPPS}(\mathcal{B}^*_S).$$

□

The bound given by this strategy cannot improve the dependency on d, by the following lemma.

Theorem 5. *For each $d \geq 1$, there exists an instance I of the BPPS with d scenarios and a solution $\mathcal{B}$ for I, which is minimal for the BPPS, such that $V_{VBPP}(\mathcal{B}) \geq \frac{\sqrt{d}}{2} V_{BPPS}(\mathcal{B})$.*

Proof. Let $r = \left\lceil \sqrt{d} \right\rceil$. We construct an instance $I = (d, \mathcal{I}, \mathcal{K}, s)$ that contains r items.

We consider a complete graph G with r vertices. For each vertex i of G, create an item i with size $s_i = 1$, and for each edge $\{i, j\}$ of G, create a scenario k with items

$S_k = \{i, j\}$. Notice that each item is in exactly $r - 1$ scenarios, and that the number of scenarios is

$$|E(G)| = \frac{r(r-1)}{2} = \frac{\lceil\sqrt{d}\rceil(\lceil\sqrt{d}\rceil - 1)}{2} \leq \frac{(\sqrt{d}+1)\sqrt{d}}{2} \leq d.$$

Now, we create a solution $\mathcal{B}$ with r bins, such that, for each item i, there is one bin $B_i = \{i\}$. We claim that $\mathcal{B}$ is minimal for the BPPS. Indeed, for any two bins B_i and B_j, items i and j are contained in a common scenario k, with $S_k = \{i, j\}$, and $s_i + s_j > 1$.

Notice that $V_{VBPP}(\mathcal{B}) = r$. Let $h = V_{BPPS}(\mathcal{B})$, so h is the maximum number of non-empty bins in a given scenario. Since each scenario contains exactly 2 items, then $h = 2$. Therefore,

$$r = \lceil\sqrt{d}\rceil \geq \frac{\sqrt{d}}{2}2 = \frac{\sqrt{d}}{2}h.$$

□

Finally, using the $1 + \ln d$-asymptotic approximation algorithm of Bansal et al. [5] for the VBPP (for constant d) and the fact that the APTAS of Fernandez de la Vega and Lueker [28] implies a $(d + \varepsilon)$-approximation for the VBPP [15], we have the following result.

Corollary 3. *There exists a $(d + \varepsilon)\sqrt{d}$-approximation for the BPPS. Also, for any constant d, there exists a $(1 + \ln d)\sqrt{d}$-approximation for the BPPS.*

3.4 An Asymptotic Polynomial Time Approximation Scheme

We present an APTAS for the BPPS when the number of scenarios d is a constant. First, we observe that, if all items are large and the number of distinct item sizes is bounded by a constant, then the number of valid patterns of items in a bin is also bounded by a constant. It implies that for this restricted case, a polynomial algorithm can be readily obtained by simply enumerating the frequencies of all patterns.

To obtain such a restricted instance, we use the *linear grouping* technique, following Fernandez de la Vega and Lueker [28], by grouping items of similar sizes and creating a map from smaller to larger items. A naive approach does not work, however, since one must take into account the scenarios. Thus, we group items according to their scenarios separately, but we do the mapping simultaneously.

Finally, to solve a general instance, we combine small items into artificial large items. Again, this is done on a per-scenario basis, so as only compatible items are combined. If the small items do not fit exactly into the area of the artificial items, then this change may cause bins to be slightly augmented. The surpassing items are then relocated to new bins greedily, increasing the number of bins by just a fraction of the optimal value.

In the following, we use the notion of type. The *type* of item i is the set of scenarios which contain i. The set of all types is denoted by $\mathcal{T}$. Also, the value of an optimal

solution for an instance I is denoted by $\mathrm{OPT}(I)$.

3.4.1 Restricted instances

In this subsection, we consider instances of the BPPS in which the items are large and the number of distinct sizes is bounded by a constant. Precisely, let V be a set of positive numbers and consider a constant ε, with $0 < \varepsilon \leq 1/4$ and $1/\varepsilon$ is an integer. An instance $(d, \mathcal{I}, \mathcal{K}, s)$ of the BPPS is said to be *V-restricted* if $\{s_i : i \in \mathcal{I}\} = V$ and $s_i \geq \varepsilon^2$ for every $i \in \mathcal{I}$.

Recall that a bin $B \subseteq \mathcal{I}$ is feasible if, for any scenario $k \in [d]$, $\sum_{i \in B \cap S_k} s_i \leq 1$. For a type $t \in \mathcal{T}$ and $v \in V$, let B_{tv} be the subset of items in B with type t and size v. Observe that the family of all sets B_{tv} form a partition of B. The *pattern* of B is the vector $(|B_{tv}|)_{t \in \mathcal{T}, v \in V}$.

Lemma 10. *If $|V|$ is constant, the number of distinct patterns for a V-restricted instance is bounded by a constant.*

Proof. Consider a feasible bin B. Since $s_i \geq \varepsilon^2$, for any $i \in B$,

$$|B|\varepsilon^2 \leq \sum_{i \in B} s_i \leq \sum_{k=1}^{d} \sum_{i \in B \cap S_k} s_i \leq \sum_{k=1}^{d} 1 = d,$$

where the last inequality follows because B is feasible. It means that the number of items in any bin is at most d/ε^2. Since a vector corresponding to a pattern has $|\mathcal{T}||V|$ elements, the number of patterns is at most $(1 + d/\varepsilon^2)^{|\mathcal{T}||V|}$, which is constant since d is constant. □

Lemma 11. *If $|V|$ is constant, then an optimal solution of a V-restricted instance can be computed in polynomial time.*

Proof. Consider a V-restricted instance I, and let M be the set of distinct patterns for I. Given a solution $\mathcal{B}$ and a pattern $p \in M$, the number of bins in $\mathcal{B}$, which has (a non-empty) pattern p, is denoted by n_p. Clearly $n_p \leq |\mathcal{I}|$ for every $p \in M$, as there are only $|\mathcal{I}|$ items, and each bin has at least one item. The configuration of $\mathcal{B}$ is the vector $(n_p)_{p \in M}$. Thus, the number of possible configurations is bounded by $|\mathcal{I}|^{|M|}$, which is polynomial since, by Lemma 10, $|M|$ is constant.

Notice that, given a vector $(n_p)_{p \in M}$, one can either find a solution $\mathcal{B}$ with this configuration, or decide there is no such solution. Indeed, it is sufficient to count the total number of items of each type and size over all the patterns. Therefore, one can list all configurations in polynomial time and return the one with the minimum value. □

3.4.2 An APTAS for large items

Suppose now that we have an instance $I = (d, \mathcal{I}, \mathcal{K}, s)$, such that all items are large, i.e., for each $i \in \mathcal{I}$, we have $s_i \geq \varepsilon^2$, but the number of distinct sizes is not necessarily bounded by a constant.

Definition 3. *Given instances $I = (d, \mathcal{I}, \mathcal{K}, s)$ and $\bar{I} = (d, \bar{\mathcal{I}}, \bar{\mathcal{K}}, \bar{s})$, we say that I dominates $\bar{I}$ if there exists a subset $\mathcal{I}' \subseteq \mathcal{I}$ and a bijection $f : \mathcal{I}' \to \bar{\mathcal{I}}$, such that, for every $i \in \mathcal{I}'$, items i and $f(i)$ have the same type and $s_i \geq \bar{s}_{f(i)}$. In this case, we write $I \succcurlyeq \bar{I}$.*

Lemma 12. *If $I \succcurlyeq \bar{I}$, then $OPT(I) \geq OPT(\bar{I})$.* □

In the following, we create an instance $\bar{I} = (d, \bar{\mathcal{I}}, \bar{\mathcal{K}}, \bar{s})$ with $I \succcurlyeq \bar{I}$, such that all items of $\bar{I}$ are large and the number of distinct sizes is bounded by a constant.

First, for each $t \in \mathcal{T}$, let $\mathcal{I}_t$ be the set of items of type t and let $m = \frac{2^d}{\varepsilon^3} - 1$. We consider the set $\mathcal{I}_t$ sorted in decreasing order of size, and build $m+1$ groups of consecutive items, obtaining a partition $\{G_t^0, G_t^1, \ldots, G_t^m\}$, where the first m groups have size $\lceil |\mathcal{I}_t|/(m+1) \rceil$, and the last group either has the same size, or is smaller.

Now, for each $t \in \mathcal{T}$ and $\ell \in \{1, 2, \ldots, m\}$, we create a set $\bar{G}_t^\ell$ as follows: for each item $i \in G_t^\ell$, add an item to $\bar{G}_t^\ell$ with the same set of scenarios t, and with size equal to the size of the largest item in G_t^ℓ. Let $\bar{\mathcal{I}}$ be the union of all sets $\bar{G}_t^\ell$, and $\bar{I}$ be the instance induced by $\bar{\mathcal{I}}$.

Lemma 13. *$I \succcurlyeq \bar{I}$.*

Proof. For $\ell \in \{1, 2, \ldots, m\}$, as $|G_t^{\ell-1}| \geq |G_t^\ell| = |\bar{G}_t^\ell|$, we can find a bijection f between a subset of $G_t^{\ell-1}$ and $\bar{G}_t^\ell$ for $\ell \in \{1, 2, \ldots, m\}$ such that i and $f(i)$ has the same type. Finally, as $s_i \geq s_j$ for $i \in G_t^{\ell-1}$ and $j \in G_t^\ell$ and $\bar{s}_{f(i)}$ is the maximum size of an item in G_t^ℓ, we have that $s_i \geq \bar{s}_{f(i)}$. □

For a type $t \in \mathcal{T}$, the group of the largest items is G_t^0. For each item i in this group, we pack it into a separate bin $\{i\}$. By joining all these bins, we obtain a packing $\mathcal{P}^0$.

Lemma 14. *$V_{BPPS}(\mathcal{P}^0) \leq \varepsilon\, OPT(I) + 2^d$.*

Proof. Let $t \in \mathcal{T}$ and fix an arbitrary scenario k of t. We consider an optimal solution for I, and let $\mathcal{P}_k$ be the set of bins used in scenario k in this solution. Therefore, $|\mathcal{P}_k| \leq \mathrm{OPT}(I)$. Observe that, since k belongs to t, the set of items included in bins of $\mathcal{P}_k$ must contain $\mathcal{I}_t$. As, for each $i \in \mathcal{I}_t$, $s_i \geq \varepsilon^2$,

$$\varepsilon^2 |\mathcal{I}_t| \leq \sum_{i \in \mathcal{I}_t} s_i \leq \sum_{B \in \mathcal{P}_k} \sum_{i \in B \cap S_k} s_i \leq \sum_{B \in \mathcal{P}_k} 1 = |\mathcal{P}_k| \leq \mathrm{OPT}(I).$$

This implies that the value of $\mathcal{P}^0$ can be bounded by

$$\sum_{t \in \mathcal{T}} |G_t^0| = \sum_{t \in \mathcal{T}} \left\lceil \frac{\varepsilon^3}{2^d} |\mathcal{I}_t| \right\rceil \leq 2^d + \sum_{t \in \mathcal{T}} \frac{\varepsilon\, \mathrm{OPT}(I)}{2^d} \leq \varepsilon\, \mathrm{OPT}(I) + 2^d.$$

□

To obtain a packing of the remaining items, we solve instance $\bar{I}$. For this, let V be the set of distinct sizes in $\bar{I}$. For each $t \in \mathcal{T}$, we created m groups, where each group has items of the same size. Therefore, $|V| \leq 2^d m$. This means that $\bar{I}$ is a V-restricted instance for $|V|$ bounded by a constant. Using Lemma 11, we obtain a solution $\bar{\mathcal{P}}^1$ for

$\bar{I}$ in polynomial time. Since $I \succcurlyeq \bar{I}$, we can obtain a solution $\mathcal{P}^0 \cup \mathcal{P}^1$ for I, where $\mathcal{P}^1$ is obtained from $\bar{\mathcal{P}}^1$ by replacing items of $\bar{G}_t^\ell$ by items of G_t^ℓ. Let I^1 be the instance obtained from the items of $\mathcal{P}^1$. Observe that $\mathcal{P}^1$ is feasible, since we replaced items of larger size which appear in the same set of scenarios, thus for each scenario, the capacity of each bin remains respected.

Lemma 15. *$V_{BPPS}(\mathcal{P}^1) \leq OPT(I)$.*

Proof. Note that the number of unused bins in each scenario is the same for $\mathcal{P}^1$ and $\bar{\mathcal{P}}^1$, thus $V_{BPPS}(\mathcal{P}^1) = V_{BPPS}(\bar{\mathcal{P}}^1)$. Since, $\bar{\mathcal{P}}^1$ is optimal for $\bar{I}$, we have $V_{BPPS}(\bar{\mathcal{P}}^1) = \mathrm{OPT}(I^1)$. Now, using Lemma 12 and the fact that $I \succcurlyeq \bar{I}$, we have $\mathrm{OPT}(I^1) \leq \mathrm{OPT}(I)$. □

Combining the previous results, one obtains a APTAS for instances with large items only.

Lemma 16. *Suppose for every $i \in \mathcal{I}$, $s_i \geq \varepsilon^2$. Then one can find in polynomial time a packing $\mathcal{P}$ of $\mathcal{I}$ with $V_{BPPS}(\mathcal{P}) \leq (1+\varepsilon)\, OPT(I) + 2^d$.*

Proof. Define $\mathcal{P} = \mathcal{P}^0 \cup \mathcal{P}^1$. Notice that $\mathcal{P}$ contains every item of $\mathcal{I}$, and thus it is feasible. Using lemmas 14 and 15,

$$V_{BPPS}(\mathcal{P}) \leq V_{BPPS}(\mathcal{P}^0) + V_{BPPS}(\mathcal{P}^1) \leq \varepsilon\, \mathrm{OPT}(I) + 2^d + \mathrm{OPT}(I).$$

□

3.4.3 An APTAS for the general case

In the general case of the BPPS, an instance $I = (d, \mathcal{I}, \mathcal{K}, s)$ may contain large and small items. Let $\mathcal{L} = \{i \in \mathcal{I} : s_i \geq \varepsilon^2\}$ be the set of large items, and $\mathcal{S} = \mathcal{I} \setminus \mathcal{L}$ be the set of small items.

To obtain a packing of $\mathcal{L} \cup \mathcal{S}$, we first replace $\mathcal{S}$ by the set of large items, $\hat{\mathcal{S}}$, which we define as follows. For each type $t \in \mathcal{T}$, let $\mathcal{S}_t$ be the set of small items with type t. Now, let $\hat{\mathcal{S}}_t$ be a set of $\lceil \sum_{i \in \mathcal{S}_t} s_i/(\varepsilon - \varepsilon^2) \rceil$ items, such that each $j \in \hat{\mathcal{S}}_t$ has size $\hat{s}_j = \varepsilon$ and is of type t. The set $\hat{\mathcal{S}}$ is the union of sets $\hat{\mathcal{S}}_t$ for every type t.

Thus, we create an instance $\hat{I} = (d, \hat{\mathcal{I}}, \hat{\mathcal{K}}, \hat{s})$ whose set of items is $\hat{\mathcal{I}} = \mathcal{L} \cup \hat{\mathcal{S}}$, each of which has size at least ε^2. Lemma 17 compares the optimal value of I and $\hat{I}$.

Lemma 17. *$OPT(\hat{I}) \leq (1 + 2^{d+1}(d+1)\varepsilon)\, OPT(I) + 1$.*

Proof. Consider an optimal solution $\mathcal{P}^*$ for I. For each bin $B \in \mathcal{P}^*$ and type $t \in \mathcal{T}$, we define $B_t = B \cap \mathcal{S}_t$, i.e., B_t is the set of small items with type t. Then, we replace the items in B_t by $\lceil \sum_{i \in B_t} s_i/(\varepsilon - \varepsilon^2) \rceil$ new items of size ε, and with the same scenarios of t. We call the modified packing by $\mathcal{P}'$.

Observe that $\mathcal{P}'$ may be infeasible, since new items may surpass the bins' capacity. We can obtain a feasible solution $\mathcal{P}''$ by relocating some created items from $\mathcal{P}'$ to new

bins. For each type t and each bin $B \in \mathcal{P}'$, the number of items needed to be picked is the additional area divided by ε rounded up, that is,

$$\begin{aligned}\left\lceil \frac{\left\lceil \sum_{i\in B_t} \frac{s_i}{\varepsilon-\varepsilon^2} \right\rceil \varepsilon - \sum_{i\in B_t} s_i}{\varepsilon} \right\rceil &\leq 1 + \frac{\varepsilon + \varepsilon \sum_{i\in B_t} \frac{s_i}{\varepsilon-\varepsilon^2} - \sum_{i\in B_t} s_i}{\varepsilon} \\ &= 2 + \frac{1}{1-\varepsilon} \sum_{i\in B_t} s_i \\ &\leq 2 + 2 \sum_{i\in B_t} s_i\end{aligned}$$

where the last inequality follows from the fact that $\varepsilon \leq 1/2$. Let R be the set of all picked items. Observe that $|\mathcal{P}^*| \leq d\,\mathrm{OPT}(I)$, since each bin is not empty for at least one scenario. Thus, as there are 2^d types and $\sum_{B\in\mathcal{P}^*} \sum_{i\in B_t} s_i \leq \mathrm{OPT}(I)$, the number of items in R is at most

$$\begin{aligned}\sum_{t\in\mathcal{T}} \sum_{B\in\mathcal{P}^*} \left(2 + 2 \sum_{i\in B_t} s_i\right) &= 2^{d+1}|\mathcal{P}^*| + 2^{d+1}\,\mathrm{OPT}(I) \\ &= 2^{d+1} d\,\mathrm{OPT}(I) + 2^{d+1}\,\mathrm{OPT}(I) \\ &= 2^{d+1}(d+1)\,\mathrm{OPT}(I).\end{aligned}$$

Since $1/\varepsilon$ is an integer, one can rearrange all items of R into at most

$$\left\lceil \frac{2^{d+1}(d+1)\,\mathrm{OPT}(I)}{1/\varepsilon} \right\rceil$$

new bins of unit capacity. Let $\mathcal{R}$ be this set of new bins, and obtain a feasible packing $\mathcal{P}''$ of I, by making a copy of $\mathcal{P}'$, removing the items of R, and adding the bins of $\mathcal{R}$.

Notice that $V_{BPPS}(\mathcal{P}') = V_{BPPS}(\mathcal{P}^*) = \mathrm{OPT}(I)$. Using these facts, the value of $\mathcal{P}''$ can be estimated as

$$\begin{aligned}V_{BPPS}(\mathcal{P}'') &\leq V_{BPPS}(\mathcal{P}') + |\mathcal{R}| \\ &= \mathrm{OPT}(I) + \left\lceil \frac{2^{d+1}(d+1)\,\mathrm{OPT}(I)}{1/\varepsilon} \right\rceil \\ &= \mathrm{OPT}(I) + 2^{d+1}(d+1)\varepsilon\,\mathrm{OPT}(I) + 1.\end{aligned}$$

We notice that $\mathcal{P}''$ contains all large items $\mathcal{L}$ and, for each type t, a certain number of items with scenarios t and size ε. Thus, to obtain a packing for $\hat{\mathcal{I}}$ from $\mathcal{P}''$, it is sufficient to show that for each t, the number of created items corresponding to t is not smaller than $|\hat{\mathcal{S}}_t|$. Indeed, for a given t, the number of created items is

$$\sum_{B\in\mathcal{P}^*} \left\lceil \sum_{i\in B_t} s_i/\varepsilon \right\rceil \geq \left\lceil \sum_{i\in\mathcal{S}_t} s_i/\varepsilon \right\rceil = |\hat{\mathcal{S}}_t|.$$

To conclude the proof, we create a packing $\hat{\mathcal{P}}$ for $\hat{\mathcal{I}}$ by removing the exceeding items

of $\mathcal{P}''$. Since removing such items can only decrease the objective value, $V_{BPPS}(\hat{\mathcal{P}}) \leq V_{BPPS}(\mathcal{P}'')$. □

Finally, to obtain a packing for I, one can first obtain a packing of $\hat{I}$, then replace items from $\hat{\mathcal{S}}_t$ by items in $\mathcal{S}_t$. This leads to Theorem 6.

Theorem 6. *For every constant $\varepsilon' > 0$, one can find in polynomial time a packing $\mathcal{P}$ of $\mathcal{I}$ such that $V_{BPPS}(\mathcal{P}) \leq (1+\varepsilon')\, OPT(I) + 2 + 2^d$.*

Proof. Define $r = 2^{d+1}(d+1)$ and define ε as the largest number such that $\varepsilon \leq \min(\frac{\varepsilon'}{3r}, 1/4)$ and $1/\varepsilon$ is an integer. Let $\hat{I}$ be the instance obtained from I by replacing, for each type, the items smaller than ε^2 with items of size ε, as previously mentioned. Since each item of $\hat{I}$ has size at least ε^2, by Lemma 16, we obtain a packing $\hat{\mathcal{P}}$ for $\hat{I}$ with value $V_{BPPS}(\hat{\mathcal{P}}) \leq (1+\varepsilon)\,\mathrm{OPT}(\hat{I}) + 2^d$ in polynomial time.

For each $t \in \mathcal{T}$, we place all items of $\mathcal{S}_t$ over the items of $\hat{\mathcal{S}}_t$. Precisely, we do the following: start by making a copy $\mathcal{P}$ of $\hat{\mathcal{P}}$; select a set A of unpacked items in $\mathcal{S}_t$ whose total size is at least $\varepsilon - \varepsilon^2$ and at most ε. If there is not such a set A, then let A be the remaining set of unpacked items in $\mathcal{S}_t$. Find a bin $B \in \mathcal{P}$ with an item $j \in \hat{\mathcal{S}}_t \cap B$ and replace j by the items of A. Since there are $\lceil \sum_{i \in \mathcal{S}_t} s_i/(\varepsilon - \varepsilon^2) \rceil$ items in $\hat{\mathcal{S}}_t$, and each group A, except perhaps one of them, has size at least $\varepsilon - \varepsilon^2$, we always find an unused item $j \in \hat{\mathcal{S}}_t$; repeat these steps while there are unpacked items in $\mathcal{S}_t$ for some t.

The resulting packing $\mathcal{P}$ contains all items in $\mathcal{I}$. As no bin was added, $\mathcal{P}$ has the same value of $\hat{\mathcal{P}}$. Using Lemma 17, we finally obtain

$$\begin{aligned} V_{BPPS}(P) = V_{BPPS}(\hat{P}) &\leq (1+\varepsilon)\,\mathrm{OPT}(\hat{I}) + 2^d \\ &\leq (1+\varepsilon)[(1+2^{d+1}(d+1)\varepsilon)\,\mathrm{OPT}(I) + 1] + 2^d \\ &= (1 + r\varepsilon^2 + r\varepsilon + \varepsilon)\,\mathrm{OPT}(I) + 2^d + 1 + \varepsilon \\ &\leq (1+3r\varepsilon)\,\mathrm{OPT}(I) + 2^d + 1 + \varepsilon \\ &\leq (1+\varepsilon')\,\mathrm{OPT}(I) + 2 + 2^d. \end{aligned}$$

□

3.5 A Variable Neighborhood Search Algorithm

The Variable Neighborhood Search is a metaheuristic that searches in different neighborhood structures and has systematic change of neighborhood to find better solutions and escape from local optima. Mladenović and Hansen [42] proposed the VNS not only for combinatorial optimization problems, but for optimization in general. Due to its high applicability to similar problems [44, 30, 6], we apply the VNS to solve the BPPS.

In order to successfully implement it, we need to clarify three key characteristics of the problem: the local search neighborhood structures and how to navigate the search space; sensitive and computationally viable objective functions; and the stopping criteria. In the following, we detail each of these and describe the resulting algorithm.

3.5.1 Neighborhood structures

A neighborhood structure specifies a well-defined way to get from any given solution into another solution that is "close" to the first one. We refer to these neighborhood structures as a "movement" that takes one solution as input and, depending on some structure-dependent input parameters, it returns a different but "close" solution. We define four neighborhood structures for the BPPS in the following order:

- N_1: Given items i and j packed into different bins, move i to the bin currently containing j, then move j to the bin that contained i;
- N_2: Given an item i and a bin B that currently does not contain i, move i to B;
- N_3: Given bins B_1, B_2, and B_3 and items i and j packed into B_1, move these items to B_2 and B_3, respectively;
- N_4: Given a bin B, remove it from the solution and repack the items in the first bin that can accommodate each of them, respecting the order that they appeared in B.

Neighborhoods N_1, N_2, and N_3 are prioritized according to their complexity. Although neighborhood structure N_4 is the smallest one, it can take comparatively more computational effort to compute since we have to repack all items of a given bin, so we consider it as the last one.

3.5.2 Objective function

The objective function for the BPPS may not be sensitive enough to rank two neighbor solutions. Thus, to guide the VNS, we define a fitness function that penalizes solutions in which the bin occupation per scenario is small.

Let $\mathcal{B}$ be a solution for the BPPS and $\mathcal{B}_k$ be the subset of bins that contain items in scenario k, where $|\mathcal{B}_k|$ is the number of bins used in k. Furthermore, let $\text{ocp}(k, B)$ be the total size of the items in bin B from scenario k. Given a solution $\mathcal{B}$, the VNS moves to a neighbor solution $\mathcal{B}'$ such that $f(\mathcal{B}) > f(\mathcal{B}')$, for f defined as:

$$f(\mathcal{B}) = V_{BPPS}(\mathcal{B}) - \sum_{k \in \mathcal{K}} \sum_{B \in \mathcal{B}_k} \left(\frac{\text{ocp}(k, B)}{|\mathcal{B}_k| W} \right)^2$$

3.5.3 Stopping criteria

Since the VNS cannot tell whether an optimal solution is reached, we use two stopping criteria to decide when the algorithm stops. The first criterion is the *timeout*, where a preset CPU time limit ($t_{\max}$) is imposed, so the VNS stops as soon as the time limit is exceeded. The second criterion is the *local-convergence*, where a preset number of iterations ($c_{\max}$) is imposed, so the VNS stops if the best known solution is not updated within this number of iterations.

3.5.4 Algorithm

The VNS takes as input an initial solution x, a number of neighborhood structures $N_{\max} = 4$, a time limit $t_{\max} = 1800$ seconds and a convergence limit $c_{\max} = 500$ iterations (these values were achieved after preliminary computational tests). The initial solution x is obtained as follows: items are sorted in non-increasing order of their sizes and then packed in the first open bin where it fits, considering the capacity constraints in every scenario and opening a new bin whenever necessary.

The overall structure of the proposed VNS is given in the following algorithm:

```
VNS(x, N_max, t_max, c_max)
 1  t = 0, c = 0
 2  while t < t_max and c < c_max
 3       κ = 1
 4       improvement = False                  // Assume there will be no improvement
 5       repeat
 6            x' = Shake(x, κ)                // Get a random solution in N_κ(x)
 7            x'' = Local-Search(x', N_max)   // Perform VND to improve x'
 8            if f(x'') < f(x)                // There has been an improvement
 9                 x = x''
10                 κ = 1
11                 improvement = True
12            else                            // Local optima, change neighborhood
13                 κ = κ + 1
14       until κ == N_max
15       if improvement                       // There has been an improvement
16            c = 0
17       else                                 // No improvement in this iteration
18            c = c + 1
19       t = CPU-Time()                       // Update processing time.
```

The VNS uses a Shake procedure that simply returns a solution x' randomly selected from the neighborhood N_κ of x. The Local-Search procedure is based on the Variable Neighborhood Descent (VND) approach, which is the deterministic descent-only version of the VNS. Our local search takes as input a solution x and the number of neighborhood structures $N_{\max} = 4$, resulting in the following algorithm:

```
Local-Search(x, N_max)
1  κ = 1
2  repeat
3       x' = arg min_{y∈N_κ(x)} f(y)     // Find the first best neighbor of x
4       if f(x') < f(x)                  // There has been an improvement
5            x = x'
6            κ = 1
7       else                             // Local optima, change neighborhood
8            κ = κ + 1
9  until κ == N_max
```

3.6 A Branch-and-Price Algorithm

We propose an exact method for the BPPS based on column generation and branch-and-bound, that is, a branch-and-price algorithm. This method consists of modelling the problem as an ILP with a very large (usually exponential) number of variables, which are generated and added to the model only as they are needed, in each node of the branch-and-bound tree. Branch-and-price has been used successfully to solve practical instances of the BPP and several of its variants [26, 46, 23, 10, 8].

One of the most successful formulations for the classical BPP and variants is based on associating a variable to each cutting pattern (see, e.g., Delorme et al. [26]). In practice, such kinds of formulations lead to models with an exponential number of variables, as the number of cutting patterns is usually exponential with respect to the number of items. Due to this, to solve practical instances with such models, one has to rely on column generation to solve the corresponding linear relaxation.

The *column generation method*, proposed by Ford Jr. and Fulkerson [31] and generalized by Dantzig and Wolfe [20], solves the linear relaxation of models with a large number of variables. Gilmore and Gomory [33, 34] were the first to present practical experiments based on column generation, by solving the linear relaxation of an exponential model for the BPP. The column generation method solves the linear relaxation by iteratively solving the *restricted master problem* (RMP), which considers only a subset of variables (columns) of the original model. At each iteration, the method has to determine the existence of non-basic columns which are candidate to improve the current solution of the RMP, i.e., columns with negative reduced cost (on minimization problems). If the method determines that no such column exists, then the solution of the RMP is an optimal solution for the original linear relaxation, providing a bound for the integer problem.

3.6.1 A pattern-based model

Recalling that $\mathcal{P}$ is the set of all cutting patterns for the BPPS, the following exponential ILP model solves the BPPS:

$$\text{minimize} \quad \mathcal{F}, \tag{3.1}$$

$$\text{s.t. :} \quad \sum_{p \in \mathcal{P}} a_{ip} X_p \geq 1, \qquad \forall i \in \mathcal{I}; \tag{3.2}$$

$$\sum_{p \in \mathcal{P}} b_{kp} X_p \leq \mathcal{F}, \qquad \forall k \in \mathcal{K}; \tag{3.3}$$

$$X_p \in \{0, 1\}, \qquad \forall p \in \mathcal{P}; \tag{3.4}$$

$$\mathcal{F} \in \mathbb{Z}_+. \tag{3.5}$$

For each pattern $p \in \mathcal{P}$, binary variable X_p is equal to 1 if and only if p is used in the solution. The integer variable $\mathcal{F}$ represents an upper bound on the number of bins (patterns) that is used by each individual scenario. The objective function (3.1) is to minimize $\mathcal{F}$. Constraints (3.2) guarantees that every item belongs to at least one cutting pattern of the solution. Constraints (3.3) ensures that, for every scenario $k \in \mathcal{K}$, the

number of bins that packs at least one item from k is at most $\mathcal{F}$. The domain of variables is in (3.4) and (3.5).

Next, we describe a column generation algorithm to solve the linear relaxation of (3.1)–(3.5). As the initial set of columns for the RMP, we consider each cutting pattern with a single item. The dual solution of the RMP consists of a value $\alpha_i \geq 0$ for each item i associated with a constraint in (3.2), and a value $\beta_k \leq 0$ for each scenario k associated with a constraint in (3.3). The dual solution is used to solve the *pricing problem*, which determines a column with minimum reduced cost, that is

$$\min_{p \in \mathcal{P}} \left\{ c_p - \left(\sum_{i \in \mathcal{I}} \alpha_i a_{ip} + \sum_{k \in K} \beta_k b_{kp} \right) \right\},$$

which is equivalent to

$$\max_{p \in \mathcal{P}} \left\{ \sum_{i \in \mathcal{I}} \alpha_i a_{ip} + \sum_{k \in K} \beta_k b_{kp} \right\}. \tag{3.6}$$

3.6.2 The pricing problem

In order to generate a new column, we need to find a feasible pattern that optimizes (3.6). Since it includes a term for each item and a term for each scenario used in the pattern, we can describe the pricing problem as the following Knapsack Problem with Scenarios.

Knapsack Problem with Scenarios (KPS) In the KPS, we are given: a number d of scenarios; a set $\mathcal{I} = [n]$ of n items, with each item $i \in \mathcal{I}$ having a size s_i, item value $v_i \in \mathbb{Q}$, and a set of scenarios $\mathcal{K}_i \subseteq [d]$; a vector $u \in \mathbb{Q}^d$ of scenario values, and a knapsack of capacity $W \in \mathbb{Q}_+$. For simplicity, for each item $i \in \mathcal{I}$, define:

$$s_i^k = \begin{cases} s_i, \text{ if item } i \text{ is in the scenario } k; \\ 0, \text{ otherwise.} \end{cases}$$

A solution is a subset B of $\mathcal{I}$ such that for each scenario $k \in \mathcal{K}$, $\sum_{i \in B} s_i^k \leq w_k$. The objective is to find a solution that maximizes

$$\sum_{i \in B} v_i + \sum_{k : s_i^k > 0,\, i \in B} u_k.$$

Theorem 7. *KPS is NP-hard in the strong sense.*

Proof. Let us recall the Maximum Independent Set problem, which given a graph G consists of finding a subset of maximum cardinality of its vertices such that no two of its elements are adjacent in G. This problem is known to be strongly NP-hard [32], so all we need to do is present a polynomial reduction from this problem into the KPS.

Let $I = G(V, E)$ be an instance of the Maximum Independent Set problem. We construct an instance $I' = (d, \mathcal{I}, s, v, \mathcal{K}, u, W)$ of the KPS such that $|\mathcal{I}| = |V|$, $d = |V|$. For each item $i \in \mathcal{I}$ there is a scenario $k_i \in [d]$ associated with it. We define $\mathcal{K}_i =$

$\{k_i\} \cup \{k_j : (i,j) \in E\}$ and we let $s_i^k = 1$ if $k \in \mathcal{K}_i$. We define $v_i = 1$ for all $i \in \mathcal{I}$, $u_k = 0$ for all $k \in [d]$ and $W = 1$.

Now notice that there is a bijection between solutions of I and I' which preserves the value of the objective function two items corresponding to two adjacent vertices of V cannot be in the same solution of I', the items are unit-valued and the scenarios are zero-valued. □

As a consequence of Theorem 7, our pricing subproblem does not admit a pseudo-polynomial time algorithm, unless $P = \text{NP}$, thus, we propose the following ILP model to solve it.

We can use the dual solution (α, β) of the RMP as item-values and scenario-values (u, v) to describe the following model for the pricing subproblem

$$\text{maximize} \quad \sum_{i \in \mathcal{I}} \alpha_i A_i + \sum_{k \in \mathcal{K}} \beta_k B_k \tag{3.7}$$

$$\text{s.t. :} \quad \sum_{i \in \mathcal{I}} s_i^k A_i \leq W B_k, \qquad \forall k \in \mathcal{K}; \tag{3.8}$$

$$A_i, B_k \in \{0,1\}, \qquad \forall i \in \mathcal{I}, k \in \mathcal{K}. \tag{3.9}$$

In model (3.7)-(3.9), variables A_i are related to the coefficients a_{ip}, and variables B_k are related to the coefficients b_{kp}. Hence, variables A and B determine the cutting pattern that is generated. The objective function (3.7) is related to the pricing formula in (3.6). Constraints (3.8) assures that the solution of the model represents a valid cutting pattern for the BPPS. Notice that W is the capacity of a bin, then if a scenario $k \in \mathcal{K}$ is used ($B_k = 1$), the sum of the sizes of all items that belong to the scenario k must be at most W, and if this scenario is not used ($B_k = 0$), then the sum of the sizes of all items that belong to k must be equal to 0.

3.6.3 The branch-and-price algorithm

To solve the model (3.1)-(3.5), we propose a branch-and-price algorithm. The branch-and-price is based on an enumeration of the fractional solution of the column generation algorithm. For this end, we use the branching scheme of Ryan and Foster [45]. Given a fractional solution X^*, we know, from Vance et al. [49], that there exists rows l and m such that

$$0 < \sum_{p : a_{lp} = 1, a_{mp} = 1} X_p < 1.$$

We use this pair of rows to create two branches for the current node. On one side we enforce that items l and m must be packed in the same pattern, and on the other side we enforce that they must be packed in different patterns. This is achieved by the following

branching constraints

$$\sum_{p:a_{lp}=1,a_{mp}=1} X_p \geq 1 \tag{3.10}$$

$$\sum_{p:a_{lp}=1,a_{mp}=1} X_p \leq 1 \tag{3.11}$$

Rather than explicitly adding these constraints to the RMP, they are enforced by setting the infeasible columns' upper bound to zero. On the branch of (3.10), this is equivalent to combine items l and m. For the branch of (3.11), this is equivalent to making rows l and m disjoint. However, this is not enough, since we must also avoid infeasible columns to be re-generated on each branch, which can be achieved by modifying the underlining pricing subproblem as follows:

- For the branches in which items l and m are combined, it suffices to modify the set of items to $\bar{\mathcal{I}} = \{1, \ldots, n\} \setminus \{l, m\} \cup \{n+1\}$, with $\alpha_{n+1} = \alpha_l + \alpha_m$,

$$s^k_{n+1} = \begin{cases} s^k_l, & \text{if } s^k_l > 0 \text{ and } s^k_m = 0, \\ s^k_m, & \text{if } s^k_l = 0 \text{ and } s^k_m > 0, \\ s^k_l + s^k_m, & \text{if } s^k_l > 0 \text{ and } s^k_m > 0, \\ 0, & \text{otherwise;} \end{cases} \tag{3.12}$$

- For branches in which items l and m must be packed in different patterns, we simply add the following disjunction constraint

$$A_l + A_m \leq 1. \tag{3.13}$$

These changes can be easily accommodated into the model (3.7)-(3.9) without much effort.

To obtain an upper bound at each node, we solve the current restricted master problem with the integrality constraint. In other words, we solve an ILP model that consists only of the columns that have been already generated. The use of this method at each node can be computationally expensive, but it usually provides tight upper bounds.

Some instances of the problem may be too complex to be solved by the column generation algorithm in practical time. In this manner, we would need faster methods for obtaining lower bounds for such instances. Two methods are proposed to get a lower bound at the root node of the branch-and-price algorithm. One of them is based on the continuous lower bound for the bin packing problem, and it is given by the following equation:

$$LB_{CON} = \max_{k \in \mathcal{K}} \left\{ \left\lceil \sum_{i \in S_k} s^k_i / W \right\rceil \right\}. \tag{3.14}$$

The idea behind the continuous lower bound (3.14) is to break items into unitary-sized items to fill empty spaces in a packing, which could not be filled otherwise. The LB_{CON} computes the continuous lower bound for every scenario, and returns the largest value

between them.

The other lower bound that we use is based on dual feasible functions. A *dual feasible function* (DFF) is a function f such that, for any finite set X of real numbers, if $\sum_{x \in X} x \leq 1$, then $\sum_{x \in X} f(x) \leq 1$. For a survey on DFFs, we refer to Clautiaux et al. [16]. Given a DFF f, we can construct a lower bound based on it, similarly to the continuous lower bound, as given in the following equation:

$$LB_{DFF} = \max_{k \in K} \{\lceil \sum_{i \in S_k} f(s_i^k / W) \rceil\} \tag{3.15}$$

In the proposed branch-and-price algorithm, given $\lambda \in \{1, \ldots, W/2\}$, we consider the DFF proposed by Fekete and Schepers [27] and described in (3.16):

$$f(s) = \begin{cases} W, \text{ if } s > W - \lambda; \\ 0, \text{ if } s \leq \lambda; \\ s, \text{ otherwise.} \end{cases} \tag{3.16}$$

3.7 Numerical Experiments

The efficiency of the proposed VNS and B&P algorithm is evaluated with computational experiments. Both solutions methods were implemented in the C++ programming language, with the B&P algorithm using the Gurobi Solver version 8.1 [36] to solve the linear programming models. The experiments were carried out in a computer with an Intel(R) Xeon(R) CPU E5–2630 v4 processor at 2.20 GHz, with 64 GB of RAM and under the Ubuntu 18.04 operating system.

A total of 120 instances were generated, divided into 12 different classes of 10 instances each. The classes are combinations of the number of items $n \in \{10, 50, 100, 200\}$ and the number of scenarios $d \in \{0.5n, n, 2n\}$, which depends on the number of items of the class. For every instance, the size of the items is randomly generated over the set $\{1, \ldots, 99\}$ under a uniform probability distribution, whereas the size of the bin is fixed to 100. The scenarios were generated randomly, similarly to Bódis and Balogh [9], such that an item $i \in \mathcal{I}$ belongs to the scenario $k \in \mathcal{K}$ with probability 0.5.

A summary of the computational results is presented in Table 3.1. Columns n and d present, respectively, the number of items and the number of scenarios of each class of instances. We consider three sets of columns, each representing one of the tested algorithms, with the last one, VNS+B&P, being the B&P algorithm using the solution obtained from the VNS as a warm-start. The columns *gap* represent the average optimality gap that each algorithm achieved in each class, computed as $\frac{UB-LB}{UB}$, where UB and LB are respectively the upper and lower bounds obtained. Notice that since the VNS does not produce a lower bound on its own, we used the lower bound from the VNS+B&P to compute its optimality gap. Columns *time* represent the average time in seconds that each algorithm spent to solve each instance of the corresponding class, being 1800 seconds the time limit for the VNS and VNS + B&P algorithms, and a time limit of 3600 for B&P. Columns $|OPT|$ represent the number of instances from each class that were solved to

optimality (out of 10) by each algorithm. Columns *cols* and *nodes* represent the average number of columns generated and the number of explored nodes, respectively, considering the B&P algorithm.

First, we discuss the performance of the VNS algorithm. From Table 3.1, we can notice that most instances with 10 items are easily solved by the heuristic. However, for instances of size 50 and 100 we can notice a significant larger gap, even though the algorithm time never reaches its time limit. This indicates that the VNS quickly found a local-optima and was not able to escape it, thus prematurely converging. Something similar can be observed for the instances of size 200, although the running time gets closer to the time limit in these cases. For this algorithm, only two instances of size $50, 100$, and 200 are solved to optimality, achieving a total average gap of 8.32% in an average time of 390.01 seconds.

Observing the results for algorithm B&P, we can notice a significant improvement. All instances of 10 and 50 items are solved to optimality very quickly (100 seconds on average). We start to notice a drop in performance for the instances of size 100 and 200. For instances of size 100, the algorithm was able to solve 11 instances to optimality, out of 30, averaging over 1800 seconds for all number of scenarios. We can also observe that the average gap for these instances is very small (around 2% for all number of scenarios). The algorithm times out for all instances of size 200, with an average gap of 14%. Interestingly, the average gaps achieved by the B&P for these instances are not far from the ones obtained by the VNS, with the latter having half the time limit of the former. The total average gap for this algorithm is 4.71% in an average time of 1648.28 seconds.

Finally, we analyze the VNS + B&P algorithm, that is, B&P using the VNS solution as a warm-start. Once again, all instances of size 10 are completely solved in negligible time. Differing from the pure B&P algorithm though, this one could not find optimal solutions for a few instances of size 50. These 5 instances brought all the averages for this group of instances up, since they were tried until the time limit of 1800 seconds, totaling thousands of columns and hundreds of nodes more than their non-warm-started counterpart. This might happen because the solution of the VNS algorithm might impact the branching step of the B&P, making it harder for the algorithm to converge depending on the branch it takes initially. However, we can notice that for instances of size 100, the VNS + B&P solved 19 instances to optimality, 8 more than the pure B&P and with a significant smaller average time. This algorithm also timed out for all instances of size 200, while presenting a greater gap in average compared to the pure B&P because of the smaller time limit. When compared to the pure B&P algorithm, the VNS + B&P gives an average gap greater by 0.16% while using less than half the time in average.

The data from Table 3.1 shows us that although the VNS algorithm struggles to avoid local-optima, its results serve as good upper bounds to be fed into a more elaborated model such as the B&P presented. By using the VNS's result as a warm-start, the B&P algorithm was able to solve more instances in less time, this was not enough to tackle instances with 200 items.

		VNS			B&P					VNS + B&P				
n	d	*gap*	*time*	|OPT|	*gap*	*nodes*	*cols*	*time*	|OPT|	*gap*	*nodes*	*cols*	*time*	|OPT|
10	5	0.00%	0.00	10	0.00%	1.0	6.9	0.02	10	0.00%	0	0	0.00	10
10	10	0.00%	0.00	10	0.00%	1.6	14.0	0.03	10	0.00%	0.3	5.4	0.00	10
10	20	1.67%	0.00	9	0.00%	1.2	11.5	0.03	10	0.00%	0.3	3.4	0.00	10
50	25	7.11%	14.80	1	0.00%	6.4	156.1	11.21	10	1.01%	955.8	98256.4	364.99	8
50	50	5.24%	28.80	1	0.00%	15.7	289.8	30.08	10	0.50%	321.8	34501.3	187.00	9
50	100	7.26%	28.80	0	0.00%	270.2	2401.1	569.41	10	0.93%	129.8	10871.7	370.18	8
100	50	9.34%	134.30	0	1.44%	80.8	2381.6	1832.76	5	0.56%	6.6	4575.5	516.59	8
100	100	8.25%	181.20	0	2.21%	85.3	2399.6	3019.09	2	1.38%	29.2	12419.1	1193.98	5
100	200	7.78%	240.40	0	1.51%	41.4	1028.1	2710.84	4	1.72%	5.8	3577.8	1052.04	6
200	100	16.32%	1048.80	0	13.74%	1.0	532.7	3814.57	0	20.13%	1	3323.1	1817.98	0
200	200	18.56%	1407.70	0	17.52%	1.0	460.3	3925.27	0	20.07%	1	4801.9	1828.71	0
200	400	18.31%	1595.30	0	20.16%	1.0	362.7	3866.15	0	20.29%	1	4832.4	1817.93	0
Average		8.32%	390.01	2.58	4.71%	42.21	837.0	1648.28	5.91	5.55%	121.05	14764	762.45	6.16

Table 3.1: Summary of the computational results for all the algorithms tested.

3.8 Concluding Remarks

This work deals with a variant of the bin packing problem where items occupy the bin capacity when they belong to the same scenario. For this problem, a variety of solution methods are proposed, including an absolute approximation algorithm, an asymptotic polynomial time approximation scheme, a variable neighborhood search based heuristic, and a branch-and-price that solves a set-cover-based formulation. Results of the heuristic and the branch-and-price illustrate how effective is the latter to solve small and medium size instances, having its efficiency improved when the heuristic provides a start solutions.

While the approximation algorithms depend either on an approximation algorithm for the vector bin packing problem, for the absolute approximation algorithm, or the number of scenarios, for the APTAS, the heuristic and branch-and-price can be used to handle the problem in practical contexts. The heuristic is not so competitive with the branch-and-price algorithm, since the former only optimally solved 26% of the instances, while the latter obtained optimal solutions for 59% of the instances. If solutions of the heuristic are used as a warm-start for the branch-and-price, the number of optimal solutions increases to about 62%. In total, we were able to prove optimality for 79 out of the 120 proposed instances.

Future research can focus on new approximation algorithms that explore the influence of the scenarios. Improvements in the variable neighborhood search are also expected, including new neighborhood structures and local searches. Regarding the branch-and-price, we will work on valid cuts and the possibility of dynamic programming algorithms for the pricing problem.

Compliance with Ethical Standards

The authors acknowledge the financial support of the National Counsel of Technological and Scientific Development (CNPq grants 144257/2019–0, 312186/2020–7, 311185/2020–7, 311039/2020–0, 405369/2021–2, 313146/2022–5), the State of Goiás Research Foundation (FAPEG), and the State of São Paulo Research Foundation (FAPESP grant 2017/11831–1). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] Cláudio Alves, José Valério de Carvalho, François Clautiaux, and Jürgen Rietz. Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *European Journal of Operational Research*, 233(1):43–63, 2014.

[2] Atefeh Baghalian, Shabnam Rezapour, and Reza Zanjirani Farahani. Robust supply chain network design with service level against disruptions and demand uncertainties: A real-life case. *European journal of operational research*, 227(1):199–215, 2013.

[3] Mauro Maria Baldi, Daniele Manerba, Guido Perboli, and Roberto Tadei. A generalized bin packing problem for parcel delivery in last-mile logistics. *European Journal of Operational Research*, 274(3):990–999, 2019.

[4] János Balogh, József Békési, György Dósa, Jiří Sgall, and Rob van Stee. The optimal absolute ratio for online bin packing. *Journal of Computer and System Sciences*, 102: 1–17, 2019.

[5] Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2009. doi: 10.1137/080736831.

[6] Jesús David Beltrán, Jose Eduardo Calderón, Rayco Jorge Cabrera, José A Moreno-Pérez, and J. Marcos Moreno-Vega. GRASP-VNS hybrid for the strip packing problem. *Hybrid metaheuristics*, 2004:79–90, 2004.

[7] Luca Bertazzi, Bruce Golden, and Xingyin Wang. The bin packing problem with item fragmentation: A worst-case analysis. *Discrete Applied Mathematics*, 261:63–77, 2019.

[8] Andrea Bettinelli, Alberto Ceselli, and Giovanni Righini. A branch-and-price algorithm for the variable size bin packing problem with minimum filling constraint. *Annals of Operations Research*, 179(1):221–241, 2010.

[9] Attila Bódis and János Balogh. Bin packing problem with scenarios. *Central European Journal of Operations Research*, pages 1–19, 2018.

[10] Yulle G. F. Borges, Flavio K. Miyazawa, Rafael C. S. Schouery, and Eduardo C. Xavier. Exact algorithms for class-constrained packing problems. *Computers & Industrial Engineering*, 144:106455, 2020.

[11] Filipe Brandão and João Pedro Pedroso. Bin packing and related problems: General arc-flow formulation with graph compression. *Computers & Operations Research*, 69: 56–67, 2016.

[12] Mirsad Buljubašić and Michel Vasquez. Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. *Computers & Operations Research*, 76:12–21, 2016.

[13] Alberto Caprara and Paolo Toth. Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, 111(3):231–262, 2001.

[14] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.

[15] Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017. doi: 10.1016/j.cosrev.2016.12.001.

[16] François Clautiaux, Cláudio Alves, and José Valério de Carvalho. A survey of dual-feasible and superadditive functions. *Annals of Operations Research*, 179(1):317–342, 2010.

[17] François Clautiaux, Mauro Dell'Amico, Manuel Iori, and Ali Khanafer. Lower and upper bounds for the bin packing problem with fragile objects. *Discrete Applied Mathematics*, 163:73–86, 2014.

[18] Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. Bin packing approximation algorithms: survey and classification. In *Handbook of combinatorial optimization*, pages 455–531. Springer, 2013.

[19] Nadia Dahmani, François Clautiaux, Saoussen Krichen, and El-Ghazali Talbi. Iterative approaches for solving a multi-objective 2-dimensional vector packing problem. *Computers & Industrial Engineering*, 66(1):158–170, 2013.

[20] G. B. Dantzig and P. Wolfe. The decomposition algorithm for linear programs. *Econometrica*, 29(4):767–778, 1961.

[21] Vinícius Loti de Lima, Manuel Iori, and Flávio Keidi Miyazawa. Exact solution of network flow models with strong relaxations. *Mathematical Programming*, 197(2): 813–846, 2023. doi: 10.1007/s10107-022-01785-9.

[22] Mauro Dell'Amico, Fabio Furini, and Manuel Iori. A branch-and-price algorithm for the temporal bin packing problem. *Computers & Operations Research*, 114:104825, 2020.

[23] Mauro Dell'Amico, Fabio Furini, and Manuel Iori. A branch-and-price algorithm for the temporal bin packing problem. *Computers & Operations Research*, 114:104825, 2020.

[24] Maxence Delorme and Manuel Iori. Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS Journal on Computing*, 32(1): 101–119, 2019.

[25] Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

[26] Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016. ISSN 0377-2217.

[27] Sándor P. Fekete and Jörg Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91(1):11–31, 2001.

[28] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[29] Esteban Feuerstein, Alberto Marchetti-Spaccamela, Frans Schalekamp, René Sitters, Suzanne van der Ster, Leen Stougie, and Anke van Zuylen. Scheduling over scenarios on two machines. In *International Computing and Combinatorics Conference*, pages 559–571, 2014.

[30] Krzysztof Fleszar and Khalil S. Hindi. New heuristics for one-dimensional bin-packing. *Computers & operations research*, 29(7):821–839, 2002.

[31] L. R Ford Jr. and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958.

[32] Michael R. Garey and David S. Johnson. "strong" NP-completeness results: Motivation, examples, and implications. *Journal of the ACM (JACM)*, 25(3):499–508, 1978.

[33] P. Gilmore and R. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9:849–859, 1961.

[34] P. Gilmore and R. Gomory. A linear programming approach to the cutting stock problem - part II. *Operations Research*, 11:863–888, 1963.

[35] Aristide Grange, Imed Kacem, and Sébastien Martin. Algorithms for the bin packing problem with overlapping items. *Computers & Industrial Engineering*, 115:331–341, 2018.

[36] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2023. URL `http://www.gurobi.com`.

[37] Katrin Heßler, Timo Gschwind, and Stefan Irnich. Stabilized branch-and-price algorithms for vector packing problems. *European Journal of Operational Research*, 271 (2):401–419, 2018.

[38] Qian Hu, Wenbin Zhu, Hu Qin, and Andrew Lim. A branch-and-price algorithm for the two-dimensional vector packing problem with piecewise linear cost function. *European Journal of Operational Research*, 260(1):70–80, 2017.

[39] Angel A Juan, W David Kelton, Christine SM Currie, and Javier Faulin. Simheuristics applications: dealing with uncertainty in logistics, transportation, and other supply chain areas. In *2018 winter simulation conference (WSC)*, pages 3048–3059, 2018.

[40] L. V. Kantorovich. Mathematical methods of organizing and planning prodution. *Management Science*, 6:363–422, 1960. (in Russian 1939).

[41] Tayfun Kucukyilmaz and Hakan Ezgi Kiziloz. Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. *Computers & Industrial Engineering*, 125:157–170, 2018.

[42] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[43] Jordi Pereira. Procedures for the bin packing problem with precedence constraints. *European Journal of Operational Research*, 250(3):794–806, 2016.

[44] Jakob Puchinger and Günther R. Raidl. Bringing order into the neighborhoods: relaxation guided variable neighborhood search. *Journal of Heuristics*, 14(5):457–472, 2008.

[45] David M. Ryan and Brian A. Foster. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pages 269–280, 1981.

[46] Ruslan Sadykov and François Vanderbeck. Bin packing with conflicts: a generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255, 2013.

[47] Michael Saint-Guillain, Célia Paquay, and Sabine Limbourg. Time-dependent stochastic vehicle routing problem with random requests: Application to online police patrol management in brussels. *European Journal of Operational Research*, 292 (3):869–885, 2021.

[48] Paul E. Sweeney and Elizabeth Ridenour Paternoster. Cutting and packing problems: a categorized, application-orientated research bibliography. *Journal of the Operational Research Society*, 43(7):691–706, 1992.

[49] Pamela H. Vance, Cynthia Barnhart, Ellis L. Johnson, and George L. Nemhauser. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational optimization and applications*, 3(2):111–130, 1994.

[50] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[51] G. Wäscher, H. Haussner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

[52] L. Wei, Z. Luo, R. Baldacci, and A. Lim. A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing*, 32(2): 428–443, 2020.

[53] Lijun Wei, Minghui Lai, Andrew Lim, and Qian Hu. A branch-and-price algorithm for the two-dimensional vector packing problem. *European Journal of Operational Research*, 281(1):25–35, 2020.

[54] Gerhard J. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters*, 64(6):293–297, 1997.

[55] Xin-yue Xu, Jun Liu, Hai-ying Li, and Man Jiang. Capacity-oriented passenger flow control under uncertain demand: Algorithm development and real-world case study. *Transportation Research Part E: Logistics and Transportation Review*, 87:130–148, 2016.

Chapter 4

Smart Energy Pricing for Demand-side Management in Renewable Energy Smart Grids

Abstract The Smart Grids are expected to provide various benefits to society by integrating advances in power engineering with recent developments in the field of Information and Communications Technology. One of the advantages is the support to efficient demand-side management (DSM), e.g. changes in consumer demands for energy based on using incentives. Indeed, DSM is expected to help grid operators balance time-varying generation by wind and solar units, and the optimization of their usage. This paper focuses on DSM considering renewable energy generation and proposes an auction, in which consumers submit bids to renewable energy usage plans. An additional model is introduced to allow consumers to compute their bid for a given usage plan. Both models have been extended to include energy storage devices. The proposed model is compared to a system with time-varying pricing for energy, where it is shown to allow consumers to use more appliances, to lead to a larger profit and to reduce the peak-to-average ratio of energy consumption. Finally, the impact of the use of energy storage in households and in the energy provider is also considered.

Keywords Demand-side Management; Smart Grid; Smart Pricing; Auction; Energy Storage Devices

4.1 Introduction

The European Technology & Innovation Platform – Smart Networks for Energy Transition (ETIP-SNET) defines smart grids as "electricity networks that can intelligently integrate the actions of all users connected to it — generators, consumers, and those that do both — in order to efficiently deliver sustainable, economic and secure electricity supplies." [12]. In a usual smart grid scenario, each consumer is equipped with *energy consumption scheduling* (ECS) devices that are able to automatically start, stop, increase or decrease the energy consumption of appliances, as well as smart metering equipment

that allows the grid to gather advanced information on energy consumption on the end users. Having such infrastructure, energy providers can obtain more information on how and when the energy is consumed, thus charging accordingly and leading the system to a better use overall instead of coping with peak demands. Consumers also benefit from the infrastructure by making better scheduling for their appliance usage while taking into consideration the changes in the energy cost set by the providers at a certain time. However, this comes at a high cost of investments in Information and Communication Technology (ICT), such as Cognitive Radio Network infrastructures, smart sensing equipment [16, 27] and research on the efficient algorithmic implementation of such systems.

Initiatives such as the Europe 2020 Climate and Energy Package — which aims to cut greenhouse gas emissions by 20%, establish 20% of the EU's energy from renewable sources, and show an improvement of 20% in energy efficiency by 2020 — have pushed for the implementation of smart grids through legislation and incentives, indicating that they may, in fact, play a key role in the transition into a more sustainable energy production, distribution, and consumption [11]. As opposed to the current grid, smart grids take advantage of advanced metering infrastructure, and supervisory control and data acquisition, while also being capable of self-healing. There is a vast body of literature regarding smart grids, however this paper focuses on the connection between game theory and demand-side management in smart grids. For an overview of the applications of game theory in smart grids, please refer to Saad et al. [25].

Demand-Side Management (DSM) is a set of techniques implemented by utility companies designed to influence the energy consumption of their end users in order to achieve a more efficient grid operation in relation to the available power plant capacity [22]. The main DSM techniques include *load management* and *demand response*. Load management is usually implemented as direct load control where based on an agreement between the utility company and consumers, the utility company can remotely control the operation of certain appliances of their consumers in order, for instance, to avoid global peaks of usage. On the other hand, demand response is based on implementing financial or other incentives to influence consumers' demand for energy. One common way to implement demand response is to use *smart pricing*, in which the utility company sets the price of the energy according to the aggregated load of the consumers, encouraging their end-users to shift their load to off-peak hours.

In this paper, a smart grid is considered in which traditional power plants (carbon, nuclear, etc.) are integrated with renewable power plants (solar panels, wind turbines, etc.) to provide power to small communities of consumers. A novel smart pricing scheme is proposed to allocate energy to consumers (households) on the basis of constraints imposed by energy production. The proposed scheme charges consumers for an energy usage plan. As usual in Auction Theory, consumers' satisfaction with a usage plan is modeled as a monetary gain, which translates to an auction bid for the respective usage plan. A separate model is also proposed to help consumers to compute their value for a given energy usage plan based on a discrete model for describing appliance usage. Plans are assigned to consumers in such a way that the renewable energy capacity is not exceeded, yet maximizes the overall value of these assigned energy plans. The Vickrey–Clarke–Groves (VCG) mechanism [33, 9, 14] is used to incentivize consumers to be truthful by charging

them in a way that it is in their best interest (financially speaking) to report their true values for the usage plans.

Since consumers only report their auction bids for usage plans, this smart pricing scheme allows for more freedom in modeling of household appliances while also maintaining full privacy for consumers since their demands are never explicitly reported to the utility company (or the other consumers).

The proposed model is able to shift energy consumption from peak hours to off-peak hours. As a consequence, it can better exploit the energy generated by renewable sources. When compared to a time-varying pricing model, it also allows consumers to use more appliances, which in turn, leads to better social welfare (the satisfaction in society). Additionally, it also leads to a larger overall profit for the utility company and it reduces the peak-to-average ratio of energy consumption.

The contributions of this paper are the following:

- Introduction of a model for handling demand-side management in which power is allocated according to consumers bids for a set of possible usage plans, with the advantages of being appropriate for renewable power plants and ensuring consumers' privacy. The energy usage plans can be designed by the power provider or developed by considering historical values.

- A discrete model to describe consumers' value for a given energy usage plan which is appropriate for modeling a wide range of operating scenarios in terms of power demand. For example, it is possible to model *must-run appliances* (appliances that must run within a specific time window), *non-interruptible appliances* (appliances that cannot be paused in order to save energy, e.g., an electric stove), *interruptible appliances* (appliances that can be paused in order to save energy, e.g., charging a Plug-in Hybrid Electrical Vehicle – PHEV) and various combinations of these. This discrete model provides a way to describe customers' utility gained from executing each appliance activity, thus incorporating the discomfort that comes from the user not executing or delaying the execution of such appliances.

- Both models are extended to consider the use of energy storage devices on both the premises of the consumers and the energy provider. The introduction of energy storage is particularly significant in scenarios involving renewable power plants since it allows the partial reshaping of the power production pattern, e.g. providing energy at times when the production is low but demand is high (for example at night, when there is no solar energy production). In general, household energy storage devices allow consumers to utilize an energy usage plan better, possibly increasing the value that can be gained from it. Moreover, power plant energy storage devices improve energy availability, and as a consequence, increase overall social welfare by allowing the assignment of more valuable plans to consumers.

The remainder of this paper is organized as follows. An overview of the related work is shown in Section 4.2. Section 4.3 describes the overall system model including: an optimization model to allocate providers' energy usage plans while maximizing the social welfare; a model that can be used by consumers to compute their utility for a usage plan,

given the utility values of using their appliances; an extension of the proposed models to incorporate energy storage devices. In Section 4.4, the practical performance of the proposed models is evaluated with numerical experiments. Finally, in Section 4.5, the final remarks are presented.

4.2 Related Work

Due to the intricate nature of energy providing, many models of DSM have been proposed in the literature, mostly focusing on achieving a smaller peak-to-average ratio (PAR, the ratio between the highest load and the average load) by shifting load to off-peak hours.

Fahrioglu and Alvarado [13] presented the first article in the literature to consider smart pricing and game theory in this context. In it, the use of incentives is considered that encourage consumers to sign up for the desired demand management contract with the (non-renewable) energy supplier while revealing their actual value for energy usage. Afterward, several articles utilized game theory in order to price energy consumption.

Mohsenian-Rad et al. [21] consider a demand-side management model consisting of an incentive-based consumption schedule scheme by using ECS devices for each consumer. In their model, the price of energy at each time is computed as a quadratic function of the aggregated load of the consumers, and the consumers react by letting their ECS devices change the starting or stopping time of their appliances. They prove that the proposed game converges to the unique Nash Equilibrium, i.e. a point in which no single player can improve the price paid by shifting any of their appliances if all players always choose their best strategy selfishly. The game is solved with both a centralized algorithm on the provider's side and a distributed algorithm to be executed individually in each consumer's ECS, thus propagating only the total load of each consumer and partially preserving their privacy.

Zhu et al. [38] expand on the model proposed by Mohsenian-Rad et al. [21] allowing it to consider a broader amount of appliances, including power-shiftable appliances which are allowed to shift the amount of power provided to them. The authors use integer linear programming and integer quadratic programming to solve both centralized and distributed versions of the proposed game, although they do not provide any mathematical guarantees that the proposed models admit a unique Nash Equilibrium. Afterward, Liu et al. [18] propose a PAR constrained model while taking into consideration the consumers' preferences such as minimizing the operation delay (considering shifts). The article proposes a distributed algorithm to minimize the amount of information exchanged and compare their results empirically with the model proposed by Zhu et al. [38].

Logenthiran et al. [19] propose a DSM model based on load shifting in which the objective is to minimize the difference between an objective load profile, defined by the utility company, and the actual consumption profile. A heuristic-based Evolutionary Algorithm is proposed to solve the problem.

Many DSM models rely on shifting the use of some appliances to off-peak hours by means of monetary incentives. However, an argument can be made that these shifts are not always desired by the consumers, thus the price paid by energy does not necessarily

reflect the actual utility of the consumers. Li et al. [17] propose a remodeling of DSM as a sparse load shifting problem that minimizes the number of interruptions and restarts for consumers' appliances. This model admits multiple Nash equilibria, and therefore the algorithm proposed searches for an equilibrium that minimizes PAR. Wang et al. [34] use a discomfort function (along with a cost of energy generation and cost of energy from the provider) to compute the utility of each consumer while planning the use of energy storage. Yang et al. [35] propose a time of use pricing scheme for DSM in which the utility of the consumers is modeled as the price paid for their load plus a user satisfaction function based on the difference between their demand and their load. A backward induction technique is used to find an equilibrium, but no proof of convergence of the algorithm is provided. Srinivasan et al. [30] present an evaluation of different dynamic pricing techniques, including the one proposed by Yang et al. [35], using real data from the Singapore electricity market.

From the perspective of the utility company, a real-time pricing scheme can be implemented considering a scenario where DSM is used. Bu and Yu [7] proposes a decision-making framework for the utility companies modeled as a four-stage Stackelberg game, in which during the three first stages the utility companies decide what kind of energy source to use, how much energy to procure and then the retail price to offer to the consumers. In the fourth stage, the consumers adjust their demands according to the prices offered. Backward induction is used to determine a sub-game perfect equilibrium. Belhaiza and Baroudi [6] propose a model for demand management in neighborhood area networks that involves strategies from both the side of the consumers and the side of the utility companies.

Most DSM models rely on repeatedly exchanging information between consumers and the utility company (sometimes even between consumers) until an equilibrium is reached. This raises concerns about revealing private information about consumers' energy usage. Because of this, it is preferable to do all computations in a distributed way so that unnecessary data is not transmitted to other parties. Even then, most DSM models require consumers to report their true demand for energy. Furthermore, due to the cyclic nature of best-response games, convergence is not always achievable in viable computational time. Because of these difficulties, auctions as a pricing model for DSM in smart grids have been studied recently in the literature, where consumers' usually bid once for each energy offering and based on these bids the utility company chooses how much energy every consumer should receive. In this context, Atzeni et al. [5] use auctions in the day-ahead phase of its DSM model. Samadi et al. [26] apply the VCG mechanism for dynamic pricing in such a way to maximize the aggregation of the utility functions of all consumers subtracted by the (non-renewable) energy production cost. Since a VCG mechanism is used, there is no incentive for consumers to lie about their true preferences regarding energy usage. Finally, Ma et al. [20] propose an enhanced Arrow-d'Aspremont-Gerard-Varet (AGV) mechanism [2, 10] for this model, which also ensures truthfulness.

Differently from the works of Samadi et al. [26] and Ma et al. [20], the proposed model does not restrict consumers on how they can obtain value from energy usage and on how they can use the energy provided. The proposed model for bid computation can consider several different types of household appliances. Moreover, as in the proposed

system, the energy allocation is independent of how bids are generated in the auction; the bid computation model can be exchanged for more suitable ones, especially in the case of commercial or industrial consumers. Additionally, since the bids only represent how much a plan is valued for a consumer, the consumer never has to explicitly state their actual load demands which preserve their privacy.

The inclusion of renewable energy sources into the grid adds more complexity to it. For a comprehensive view of the difficulties of integrating renewable energy resources and some of the recent efforts in doing so, please refer to the work of Rehmani et al. [23].

Due to the unpredictable nature of renewable energy sources, the actual energy produced on a certain day can vary with factors such as wind speed and solar irradiance. Because of this, methods of forecasting these factors have been developed in order to provide accurate predictions for the amount of renewable energy that can be generated by a plant in a given time period. Thus, forecasting of wind and solar power have been extensively studied in the literature. However, evaluating these predictions can be challenging since their performances can also vary with many factors, such as forecast time horizons, quality of the data used, distribution of wind speeds, and topography of the area studied [31]. One common metric used to evaluate these models is the Root Mean Squared Error (RMSE). This metric is simply the square root of the quadratic mean of the differences between predicted and observed values (standard deviation of the prediction errors), and it gives some insight into how far the observed values are from the predicted line.

Soman et al. [29] present an overview and comparative analysis of various forecasting techniques for wind power and speed. They separate the techniques in the following time horizons: very short term (few seconds to 30 seconds ahead); short term (30 minutes to 6 hours ahead); medium term (6 hours to 1 day ahead); and long term (1 day to 1 week ahead). Ssekulima et al. [31] present a review of both solar and wind forecasting in the context of integration of renewable energy to the grid and its challenges. More recently, Ahmed and Khalid [1] review several renewable energy forecast techniques in the literature with an application-oriented approach. One of the applications presented by the authors is the impact of using forecasting of renewable energy generation on energy markets.

In order to achieve higher accuracy, a trend has developed of combining several different models to mitigate the individual weaknesses of each of them [31]. Ren et al. [24] review several ensemble methods based on existing models for accurately forecasting wind and solar power. For example, one can notice in the authors' experiments that for time horizons of 24 hours, many techniques achieved an RMSE of around 0.10. Because of this, it is reasonable to assume that a DSM model can use a state-of-the-art forecasting method with relatively low errors to predict the amount of renewable energy available in a day-ahead fashion.

Regarding models for renewable energy usage in demand-side management, some works [4, 28, 8, 36, 34, 37] consider the integration of decentralized renewable energy sources, as well as energy storage devices. The model proposed in this paper is different from theirs since it considers that the energy supplier is the one supplying renewable energy, not the consumers.

Works, such as the ones by Atzeni et al. [4] and Zazo et al. [37], consider the intermittent and stochastic nature of renewable energy generation and the unreliability of the consumers' load prediction when it comes to real-time consumption. Since the model proposed in this paper considers that the energy is provided by selling energy usage plans which are complemented with non-renewable energy, the consumers can modify their load according to their needs. Furthermore, regarding the uncertain nature of renewable energy generation, the model proposed in this paper does not address this problem directly, but it can be slightly modified to consider the case where the energy supplier utilizes batteries and non-renewable energy to reduce its impact on the system. In the experiments, it is shown that this does not significantly impact the quality of the proposed model. Another possibility would be to be conservative when predicting the energy generation in such a way that any excedent renewable energy could be sold as non-renewable energy (as the model already does).

4.3 System Model

In this section, a system model for smart pricing in DSM in the context of smart grids is presented. First, a general model of smart pricing is described, then in the subsequent subsections, two optimization problems that compose the proposed model are formally defined: the allocation of usage plans to consumers who have given their bids; and the evaluation of energy usage plans for bidding on the side of the consumers. Both problems are modeled as integer linear programming formulations that can be solved to optimality by the commercial solver Gurobi [15] in a reasonable time.

4.3.1 General Description

In the scenario considered in this paper, the utility company controls a centralized renewable energy source and access to supplementary non-renewable energy. The company aims at allocating the clean energy produced to consumers in a way that maximizes the social welfare of the community while attending to the constraints imposed by the capacity for production of renewable energy. This scenario can be encountered, for instance, in communities that have access to renewable energy sources and want to avoid using non-renewable energy as much as possible or in governments which want to provide social welfare using renewable energy instead of relying on non-renewable sources.

In the model, a set T of discrete time slots $\{1, 2, \ldots, t_{\max}\}$ with $t_{\max}$ being a positive integer is considered. For each time slot, consumers receive a constant amount of energy from the utility company. For instance, one possibility would be to consider each time slot as an hour in a day, therefore $t_{\max}$ would be 24.

The utility company has a set P of *energy usage plans*. An energy usage plan is a non-negative vector that represents the maximum energy (in kWh) a consumer is allowed to utilize during each time slot from this source (the user can buy supplementary energy which will possibly be non-renewable). The consumer will, however, only receive the amount of energy actually used. The energy usage plans can be developed by the utility

company considering historical data, such as energy production and consumption. Alternatively, they can consist of simple usage plans such as constant plans in which the same amount of energy is provided for every hour or plans which provide low levels of energy for some hours and higher levels for others. In fact, the power provider can adjust the energy usage plans as time passes in order to control the energy demand. In Section 4.4, some sample usage plans are presented.

Energy allocation is modeled as a sealed-bid auction using the VCG mechanism [33, 9, 14]. That is, energy usage plans are provided to a set C of consumers who will bid for them. Each consumer $c \in C$ bids a value $b_{c,p}$ for the energy usage plan $p \in P$. Bids can be computed in any way desired by the consumer, but the VCG mechanism guarantees that the best strategy for the consumers is to report their values truthfully, i.e. consumers will not benefit from giving a bid that does not correspond to their true intentions. As the auction is a sealed-bid, each consumer submits their bids only once and the auction is executed after all bids are submitted. The utility company has a limited amount of energy available for each time slot limited by their capacity to produce renewable energy, therefore if w_t is the available energy (in kWh) at time t, consumers compete against each other for this (limited) energy resource.

The objective is to assign at most a single usage plan to every consumer, constrained by the power plant capacity w in order to maximize *social welfare*, which is the sum of $b_{c,p}$, for each consumer $c \in C$ and plan p. Since consumers could strategically misrepresent their actual value in order to profit, the VCG mechanism is utilized to guarantee that it is in the best interest of consumers to declare values truthfully. Thus, in this paper it is considered that $b_{c,p}$ represents the real value of usage plan p for consumer c, i.e. consumers do not provide wrong or forged information. In Section 4.3.3, one way for consumer c to compute a bid $b_{c,p}$ for a given usage plan p is introduced. Figure 4.1 presents a flowchart of the proposed system model.

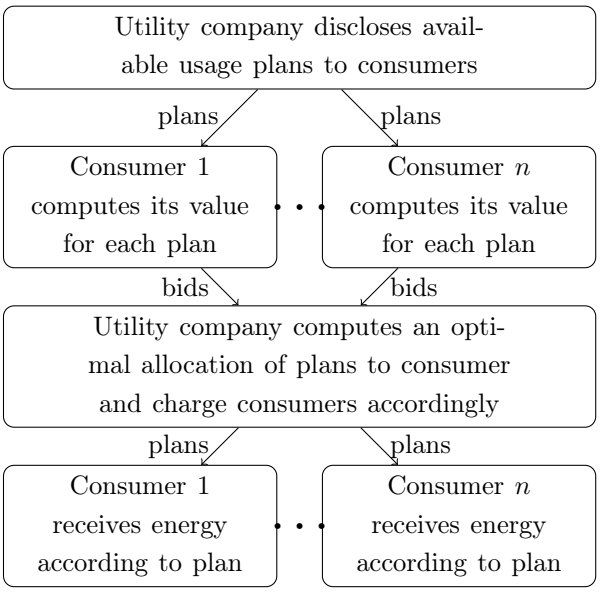

Figure 4.1: A flowchart representing the system model.

In the proposed model, the cost of generating energy is not explicitly considered,

since the model is designed for renewable energy sources such as wind and solar energy, where the output is not controlled by how much resources are consumed. Thus, the pricing strategy focuses on guaranteeing fair resource usage instead of covering energy production expenses or maximizing profit. Nonetheless, profits can be reverted to system maintenance or improvement as a consequence of using renewable energy.

4.3.2 Energy Allocation

In the proposed model, the utility company needs to define how to assign energy plans to consumers with the aim of maximizing social welfare based on consumers' bids but constrained by the power plant capacity. That is, the company must allocate usage plans to consumers so that the consumer that values the most each usage plan obtains it as long as there is enough renewable energy to accommodate it in the aggregated load.

This problem is modeled as an integer linear program. Consider that $T = \{1, 2, \ldots, t_{\max}\}$ is a set of discrete time slots, P is a set of usage plans, C is the set of consumers, $b_{c,p}$ is the bid of consumer $c \in C$ for plan $p \in P$, p_t is the amount of energy (in kWh) of usage plan p at time slot t and w_t is the power plant capacity (in kWh) at time t. Let x be a binary vector indexed by $C \times P$, where $x_{c,p}$ indicates if usage plan p is assigned to consumer c. The following integer program formulation consists of finding the values of x that

$$
\begin{aligned}
\max \quad & \sum_{c \in C} \sum_{p \in P} b_{c,p}\, x_{c,p} \\
\text{s. t.} \quad & \sum_{p \in P} x_{c,p} \leq 1, && \forall c \in C, \\
& \sum_{c \in C} \sum_{p \in P} p_t\, x_{c,p} \leq w_t, && \forall t \in T, \\
& x_{c,p} \in \{0, 1\}, && \forall c \in C, \forall p \in P.
\end{aligned}
$$

This integer linear programming formulation can be solved using a commercial solver such as Gurobi [15] to obtain an optimal allocation of energy usage plans to consumers at a reasonable time.

Consumer c will be charged the value $\pi_c(b, x)$, which depends on the bids b reported by the consumers as well as on the optimal solution x of the integer linear program computed for b.

In order to compute prices, the VCG mechanism is used in conjunction with the Clarke pivot rule. This guarantees that every consumer will pay a non-negative price for energy consumption, that no consumer will pay more for an energy usage plan than their bid for it, and that consumers will be compelled to report their bids truthfully. Thus, applying the VCG mechanism with the Clarke pivot rule, the value of $\pi_c(b, x)$ is defined as:

$$
\pi_c(b, x) = \max_{x' \in \mathcal{S}} \left\{ \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x'_{c',p} \right\} - \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x_{c',p}, \tag{4.1}
$$

where $\mathcal{S}$ is the set of feasible solutions for the integer linear program. That is, every consumer is charged their *externality*, the difference between the maximum social welfare obtained by other consumers when consumer c is not in the market and the social welfare obtained by the other consumers when consumer c is in the market.

Even though the model can have multiple optimal solutions, any of these solutions would be fair to the consumers, since, for any two optimal solutions x^* and $\tilde{x}$ in which a consumer c receives plan p^* in x^* and plan $\tilde{p}$ in $\tilde{x}$, the value obtained from the usage plan minus the price paid is the same, that is, $b_{c,p^*} - \pi_c(b, x^*) = b_{c,\tilde{p}} - \pi_c(b, \tilde{x})$. In fact,

$$b_{c,p^*} - \pi_c(b, x^*) = b_{c,p^*} - \max_{x' \in \mathcal{S}} \left\{ \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x'_{c',p} \right\} + \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x^*_{c',p} \tag{4.2}$$

$$= \sum_{c' \in C} \sum_{p \in P} b_{c',p} x^*_{c',p} - \max_{x' \in \mathcal{S}} \left\{ \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x'_{c',p} \right\} \tag{4.3}$$

$$= \max_{x \in \mathcal{S}} \left\{ \sum_{c' \in C} \sum_{p \in P} b_{c',p} x_{c',p} \right\} - \max \left\{ \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x'_{c',p} \right\} \tag{4.4}$$

$$= \sum_{c' \in C} \sum_{p \in P} b_{c',p} \tilde{x}_{c',p} - \max_{x' \in \mathcal{S}} \left\{ \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x'_{c',p} \right\} \tag{4.5}$$

$$= b_{c,\tilde{p}} - \max_{x' \in \mathcal{S}} \left\{ \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x'_{c',p} \right\} + \sum_{c' \in C \setminus \{c\}} \sum_{p \in P} b_{c',p} x_{c',p} \tag{4.6}$$

$$= b_{c,\tilde{p}} - \pi_c(b, \tilde{x}) \tag{4.7}$$

where the equality between (4.3) and (4.4) follows from the optimality of x^* and the equality between (4.4) and (4.5) follows from the optimality of $\tilde{x}$.

4.3.3 Bid Computation

In order to consume energy according to the scheduled energy in a usage plan, a consumer must decide which appliances to turn on or off and at what specific time. A consumer has many alternatives to choose from when deciding this appliance usage. Let an *alternative* be a combination of the energy usage of one appliance in kWh for every time slot, with a value, representing consumer satisfaction associated with using such appliances in the times described by the alternative. For example, the consumer can choose to watch TV from 20:00 to 22:00, consuming a constant amount of energy for each time unit, or use a washing machine from 09:00 to 12:00, which consumes a variable amount of energy dependent on the washing cycle. Some alternatives, however, are mutually exclusive. For example, the consumer can charge their Plug-in Hybrid Electric Vehicle (PHEV) from 21:00 to 06:00 using a certain amount of energy during all of those hours or charge the PHEV from 00:00 to 05:00 but using a larger amount of energy per hour. The consumer can, however, choose only one of the alternatives since the PHEV is only charged once.

In the case of renewable energy, it may be unrealistic to expect consumers to be able

to satisfy their demands using only this type of energy in the fulfillment of their usage plan. It is thus assumed that consumers can also buy supplementary energy at a fixed hourly rate, possibly from another energy provider, in order to supplement the energy which a usage plan provides. This system is well suited for renewable energy, especially in the case of small generators in a community, and is actually optional, since the price of the supplementary energy can be set to infinity (or, in practice, a relatively large value), which will forbid the usage of non-renewable energy.

More formally, this scenario can be described in the following way:

- a set $T = \{1, 2, \ldots, t_{\max}\}$ of discrete time slots,
- a usage plan p, where p_t is the capacity of the usage plan at time t,
- a set A of alternatives where, for every $a \in A$, v_a is the value gained from choosing alternative a and $u_{a,t}$ is the amount of energy consumed by alternative a at time t,
- a family $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ where $A_i \subseteq A$ and exactly one alternative must be chosen from A_i,
- a cost r_t for every supplementary kWh consumed at time t.

Let y be a binary vector indexed by A where y_a indicates if alternative a is chosen and λ be a vector indexed by T where λ_t is the amount of supplementary energy (in kWh) consumed at time t. The following integer linear program formulation computes the optimal bid $b_{c,p}$ for a consumer c with alternative set A for an energy plan p.

$$
\begin{aligned}
b_{c,p} = \max & \sum_{a \in A} v_a y_a - \sum_{t \in T} r_t \lambda_t \\
\text{s. t.} \quad & \sum_{a \in A_i} y_a = 1, && \forall A_i \in \mathcal{A}, \\
& \sum_{a \in A} u_{a,t} y_a \leq p_t + \lambda_t, && \forall t \in T, \\
& y_a \in \{0, 1\}, && \forall a \in A, \\
& \lambda_t \geq 0, && \forall t \in T.
\end{aligned}
$$

Appliance i can be modeled by defining a set A_i of alternatives for it. Since exactly one alternative for every set A_i is chosen, a must-run appliance is represented by a set of alternatives, as exactly one of them will be chosen. Optional appliances can be represented by adding an alternative with zero energy consumption and zero value.

Non-interruptible appliances can be modeled considering only alternatives where the appliance runs from start to finish, that is, alternatives a where $u_{a,t} > 0$ if and only if t is in a pre-specified time interval. Interruptible appliances can be modeled by considering all possible interruptions that can occur. For example, an interruptible appliance which should run for 4 hours, but it can be interrupted only once in the middle of the cycle and that consumes $1kW$ per hour can be modeled by considering as alternatives every vector

u_a so that there are k and ℓ in T with $u_{a,k} = u_{a,k+1} = u_{a,\ell} = u_{a,\ell+1} = 1$ and $u_{a,t} = 0$ for every $t \notin \{k, k+1, \ell, \ell+1\}$.

The model can also represent the arbitrary mutual exclusion of alternatives for different appliances (e.g. a consumer might decide either to watch TV or to use a computer in the evening, but not both). In Sect. 4.4.1, more concrete examples of appliances are given.

4.3.4 Incorporating Energy Storage

Extensions of the previous optimization problems considering energy storage devices [4, 8, 34] are presented in this section. Energy storage devices help consumers to better utilize their energy usage plans so that unused energy can be consumed at times of greater need. Energy storage devices are also useful to utility companies. By using energy storage devices, for example, utility companies can store energy during days when solar energy production is high and use it at night, when no solar energy is generated.

Let y be a binary vector indexed by A where y_a indicates if alternative a is chosen, λ be a vector indexed by T where λ_t is the amount of supplementary energy (in kWh) consumed at time t, κ, z^+ and z^- be vectors indexed by T such that κ_t represents the amount of energy (in kWh) stored at the beginning of time slot t, z_t^- represents how much energy (in kWh) is discharged at time slot t and z_t^+ represents how much energy (in kWh) is charged during time slot t. The following integer linear program formulation consists of finding y, λ, κ, z^+ and z^- that

$$
\begin{aligned}
b_{c,p} = \max \sum_{a \in A} v_a y_a - \sum_{t \in T} r_t \lambda_t & \\
\text{s. t.} \quad \sum_{a \in A_i} y_a = 1, & \qquad \forall A_i \in \mathcal{A}, \\
\sum_{a \in A} u_{a,t} y_a \leq \beta z_t^- - \alpha z_t^+ + p_t + \lambda_t, & \qquad \forall t \in T, \\
z_t^+ - z_t^- \leq \delta^{\max}, & \qquad \forall t \in T, \\
\kappa_t \leq K, & \qquad \forall t \in T, \\
\kappa_{t+1} = \gamma \kappa_t + z_t^+ - z_t^-, & \qquad \forall t \in T, \\
y_a \in \{0, 1\}, & \qquad \forall a \in A, \\
\lambda_t, \kappa_t, z_t^-, z_t^+ \geq 0, & \qquad \forall t \in T,
\end{aligned}
$$

where $\kappa_{t_{\max}+1} = \kappa_1$, K is the capacity of the energy storage device in kWh, $\alpha \geq 1$ and $\beta \leq 1$ are parameters associated with charging and discharging efficiencies, respectively, $\gamma \leq 1$ is the leakage rate, and $\delta^{\max}$ is the maximum charging rate in kWh. Even though this model allows the battery to be charged and discharged at the same time, notice that this would waste more energy. In fact, any feasible solution where the battery is both charged and discharged at time t can be modified to either charge $z_t^+ - z_t^-$ kWh if $z_t^+ - z_t^- \geq 0$ or discharge $z_t^- - z_t^+$ kWh if $z_t^+ - z_t^- \geq 0$. This integer linear programming model is a generalization of the model previously described in Section 4.3.3.

The model for energy allocation to consumers can be extended to include energy storage in a similar way and will not be shown in this paper. Moreover, one can compute

the prices, as shown before in Section 4.3.2 using Equation (4.1).

4.4 Simulation and Experimental Results

In this section, the performance of the proposed model using a randomly generated instance is assessed. Results comparing the proposed model with a system in which consumers do not have ECS are omitted since it is well established in the literature that the usage of ECS devices can improve the social welfare [26]. In the evaluation, the impact of adding energy storage devices both on the consumers' and energy providers' side is considered. First, the way in which instances are generated is described in detail and, later, the results are discussed.

4.4.1 Data generation and assumptions

Consumers are assumed to have a set of appliances in which the alternatives represent different usage patterns for those appliances. Thus, every A_i of $\mathcal{A}$ represents the alternatives for a specific appliance i. For many of those appliances, all alternatives are equally good for the consumer regardless of the starting and ending times and, all have the same satisfaction value. For other appliances, the value depends on the starting and ending times. For one of the appliances, there is a basic alternative a which brings the consumer the most satisfaction when using the appliance, and this is the basis for setting the value of the other alternatives.

The level of satisfaction obtained by using the appliance at time t, according to the basic alternative a^*, is computed by using the following function (as done by Samadi et al. [26]):

$$v(\omega, u_{a^*,t}) = \begin{cases} \omega\, u_{a^*,t} - \dfrac{u_{a^*,t}^2}{4}, & \text{if } 0 \leq u_{a^*,t} < 2\omega \\ \omega^2, & \text{if } u_{a^*,t} \geq 2\omega \end{cases}$$

where ω is a random parameter chosen uniformly from a list of possibilities. That is, the value v_{a^*} of alternative a^* is computed as $\sum_{t \in T} v(\omega, u_{a^*,t})$. Thus, v_{a^*} is the maximum value for that appliance. This function is used only to generate an instance for the model and, in fact, a consumer could give any non-negative value for any alternative.

Table 4.1 presents the parameters considered for the appliances for which any alternative is equally good. It is also possible to define the time relationships between appliances, such as the following: the clothes dryer (if used) starts only immediately after the washing machine is finished; if the clothes dryer is going to be activated, the corresponding washing machine activation is free to start at any time.

For air conditioning, lighting and entertainment, the consumer has an ideal time interval, I, of usage and obtains the value v_{a^*} if they can be used in this time interval. Moreover, all the discrete non-empty subintervals of I are alternatives, each with value $\rho_a^k v_{a^*}$, where $\rho_a \in [0, 1]$ is the value discount rate of appliance a and k is the number of hours missing in the corresponding time interval. Table 4.2 shows the associated parameters for these appliances. The ideal time interval for air conditioning usage is randomly chosen for each

Appliance	ω	Starting times	Duration	Power (kW)
Clothes iron	$4, 6, 8, 10$	08:00–16:00	2 h	1.00
Dishwasher	$4, 6, 8, 10$	Any time	1 h	1.44
Generic (×4)	$2, 4, 6$	06:00–23:00	1 h	1.50
Pool pump	$2, 4, 6, 8$	Any time	2 h	2.00
Vaccum cleaner	$4, 6, 8, 10$	08:00–16:00	2 h	1.50
Washing machine	$4, 6, 8, 10$	08:00–16:00 or any time (with clothes dryer)	2 h	1.70
Clothes dryer	$4, 6, 8, 10$	Optionally after washing machine	2 h	1.25
Water heaters (x2)	$2, 4, 6, 8, 10, 12$	06:00–07:00 or 18:00–22:00	1 h	1.00

Table 4.1: Parameters of appliances whose alternatives have the maximum value.

Appliance	ω	Target int.	ρ	Power (kW)
Air Conditioning	$2, 4, 6, 8, 10, 12$ or 14	See description	0.8	1.00
Entertainment	$6, 8, 10, 12, 14, 16, 18$ or 22	18:00–00:00	0.9	1.50
Lighting (evening)	$2, 4, 6, 8, 10$ or 12	18:00–00:00	0.9	0.25
Lighting (morning)	$2, 4, 6, 8, 10$ or 12	06:00–08:00	0.9	0.25

Table 4.2: Parameters of appliances based on a target usage time interval.

consumer: it is given by r – 22:00 where the same r is chosen uniformly in the interval from 09:00 to 17:00.

It is also possible to consider alternatives, such as using a stove in the morning and in the evening, with 1 kWh being consumed for 2 hours after being activated. The value of ω, in this case, is chosen from the set $\{10, 12, 14, 16, 18\}$. In the morning version, the starting times are chosen from the set $\{11{:}00, 12{:}00, 13{:}00\}$ in which 12:00 is the preferred one and, for the evening time slot, the starting time is chosen from $\{18{:}00, 19{:}00, 20{:}00, 21{:}00, 22{:}00\}$, with 20:00 being the preferred time. Let p be the preferred starting time of the appliance, an alternative starting at time t has a value of $0.9^{|t-p|}v_{a^*}$.

In relation to charging a PHEV, ω is chosen from the set $\{2, 4, 6, 8, 10\}$, and, for every subinterval I between 18:00–08:00, there is an alternative A that consumes 9.9 kWh evenly distributed in I. The value of A is thus given by $0.9^k v_{a^*}$, where k is the number of hours not in the interval of 20:00–06:00, and a^* is the alternative which charges the PHEV from 20:00 to 06:00.

Three types of energy usage plans are considered:

- $\textsc{Flat}_k$: a usage plan where every position has value k,
- $\mathrm{U}_{k,i}$: a usage plan where the value is k for time slots $1, 2, \ldots, i-1$ and $k+1$ elsewhere,
- $\mathrm{D}_{k,i}$: a usage plan where the value is k for time slots $i+1, i+2, \ldots, t_{\max}$ and $k+1$ elsewhere.

where $k \in \{0, 0.25, \ldots, 3.0\}$ and $i \in \{0, \ldots, 23\}$.

The instance considered has $n = 100$ consumers and, if some of the consumers have energy storage devices, only 30% are considered to do so.

The renewable energy production capacity was obtained by scaling the mean solar and wind energy produced in the CSUD region of Italy for each hour in a five day period (from 06–11–2016 to 10–11–2016). Fig. 4.2 shows the mean energy production in the period. Let W'_t be the mean renewable energy produced at time t. In order to obtain the energy production capacity, W'_t is scaled by $2n|T|/\sum_{t\in T} W'_t$, and, thus, the mean energy capacity per hour and per consumer is 2 kWh; this is sufficient to induce some competition among clients while meeting the clients' basic energy needs.

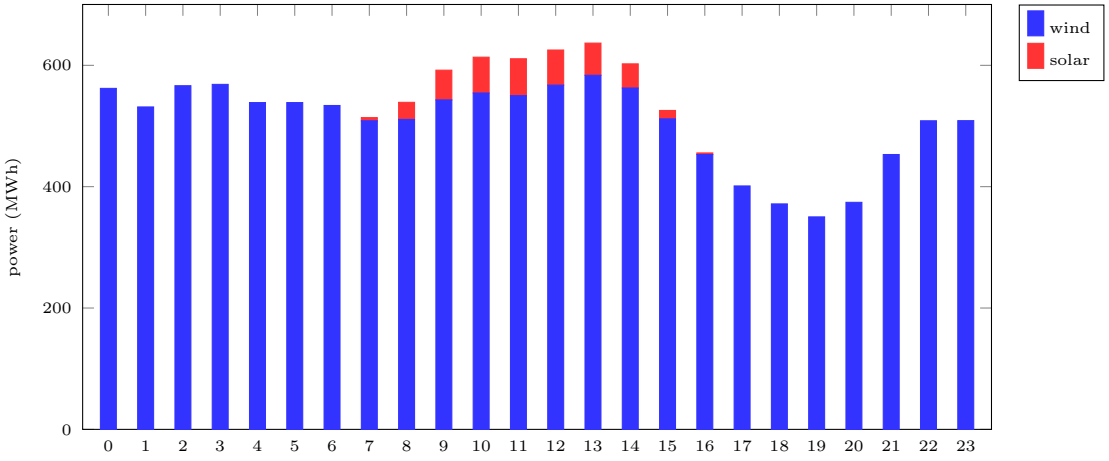

Figure 4.2: Mean solar and wind energy production in Italy (CSUD region) from 06-11-2016 to 10-11-2016 [32].

The consumers' energy storage device is a lithium-ion battery with parameters $\alpha = 0.9^{-1}$, $\beta = 1.1^{-1}$, $\delta^{\max} = 0.5\,\text{kWh}$, $\gamma = \sqrt[24]{0.9}$ and $K = 4\,\text{kWh}$ as specified by Atzeni et al. [3], and the energy provider's battery has a value of 40 for the parameter K and 5 for the parameter $\delta^{\max}$, and the same values for parameters α, β and γ.

In order to establish a baseline for the proposed model, we compare it to a time-varying pricing model, where consumers can buy energy at the price r_t at time t. In this model, renewable energy is considered to have priority over non-renewable energy, that is, non-renewable energy will only be sold after selling all the renewable energy at time t. Thus, this model only wastes renewable energy when there is no demand at a specific time. This scenario describes the situation where, for instance, the consumers are not willing to participate in the auction for acquiring renewable energy, so they pay the same fixed price that they would pay if they only used the supplemental non-renewable energy (although some of the renewable energy may be delivered to them to diminish the environmental impact of the community).

A similar assumption is made for the proposed system. Any unallocated renewable energy can be sold as supplementary energy at a time-varying price with renewable energy sold first and non-renewable energy afterward. Thus, in this model, there are two types of renewable energy loss: one because consumers may not consume all the energy allocated to them in their given usage plan; and the other because the demand at a given time is less than the renewable energy produced.

In the evaluation, three different *scenarios* are considered; one where supplementary energy is "inexpensive" ($r_t = 6$ for every $t \in T$), one in which supplementary energy is "expensive" ($r_t = 9$ for every $t \in T$) and, finally, one "mixed" scenario where supplementary energy is expensive at peak demand hours and inexpensive otherwise ($r_t = 6$ for $1 \leq t \leq 18$ and $r_t = 9$ for $19 \leq t \leq 24$).

4.4.2 Social welfare and revenue

Table 4.3 presents the number of appliances used on average by a consumer, the percentage of the maximum possible social welfare, and the percentage of the maximum possible revenue obtained by the two models.

	Proposed			Time-varying		
Scenario	Appliances	Welfare	Revenue	Appliances	Welfare	Revenue
Inexpensive	15.99	90.50%	36.55%	12.07	79.78%	37.25%
Mixed	15.67	88.81%	37.07%	11.87	74.85%	32.66%
Expensive	14.55	85.33%	41.19%	10.24	63.09%	18.89%

Table 4.3: The number of appliances used on average and percentage of the maximum social welfare and revenue obtained by the two models.

The proposed model was able to increase the average number of appliances used by a consumer and the social welfare by 32.48% and 13.44% in the inexpensive scenario, by 32.01% and 18.66% in the mixed scenario, and by 42.09% and 35.24% in the expensive scenario, respectively. From this, it can be concluded that the consumers have a strong incentive to participate in the auction, since by doing so they will greatly increase their welfare and the number of appliances used.

This result was expected since in the time-varying pricing model alternatives are chosen according to the difference between value and price, which leads consumers to choose only alternatives with high value and low energy consumption. On the other hand, in the proposed model a consumer considers all appliances at once when bidding for an energy usage plan. Moreover, it is the objective of the proposed model to maximize social welfare (even though this is done indirectly by considering energy usage plans), while the time-varying model is not concerned with this.

Finally, the revenue in the proposed model decreased by 1.89% in the inexpensive scenario, but increased by 13.50% in the mixed scenario and 118.02% in the expensive scenario. This means that since the proposed model allows consumers to use more energy, it is able to earn more money for energy.

4.4.3 Energy allocation

Figures 4.3, 4.5 and 4.7 present renewable and non-renewable energy consumption and the renewable energy waste in the proposed model for the three scenarios considered, while Figures 4.4, 4.6 and 4.8 present those statistics for the time-varying model of pricing.

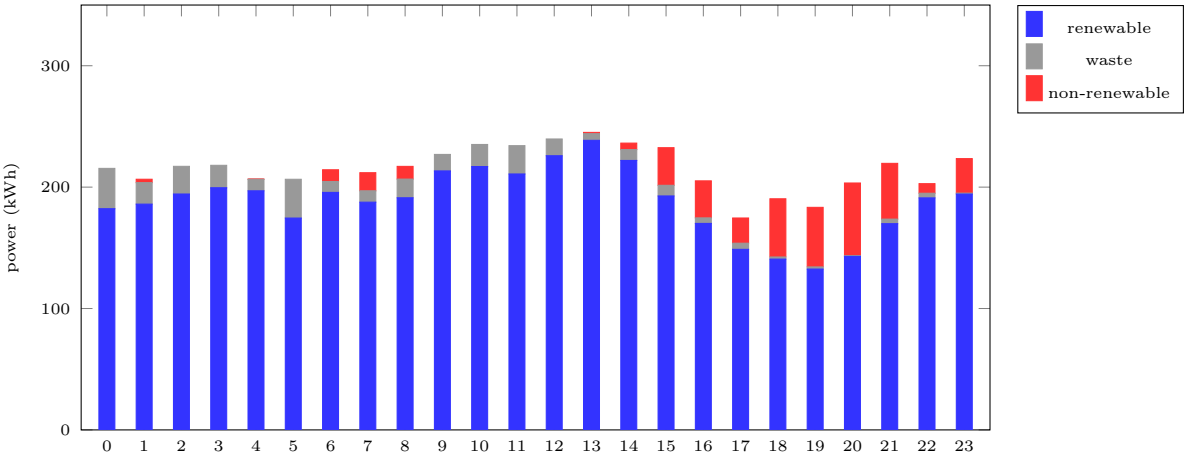

Figure 4.3: Energy consumption using the proposed model in the inexpensive scenario.

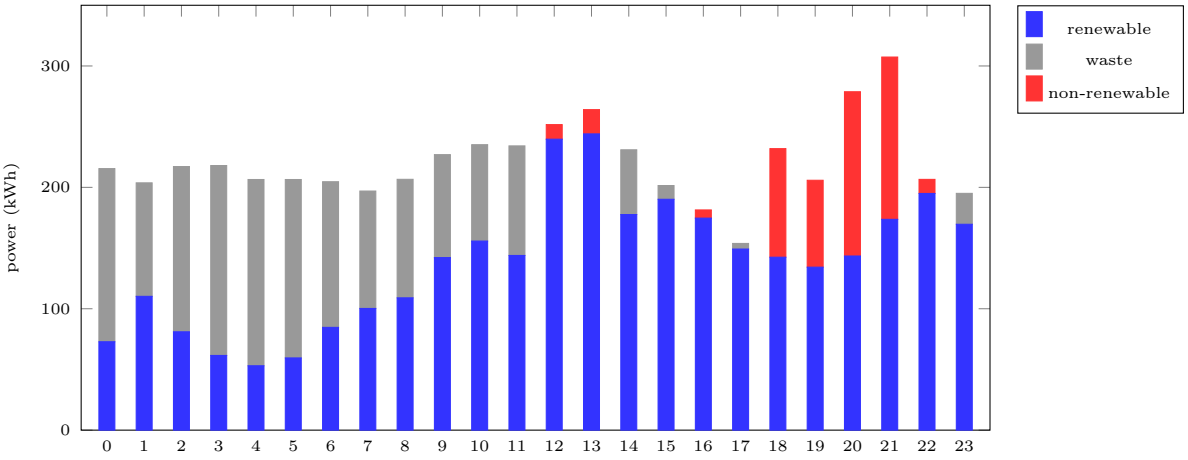

Figure 4.4: Energy consumption using the time-varying model of pricing in the inexpensive scenario.

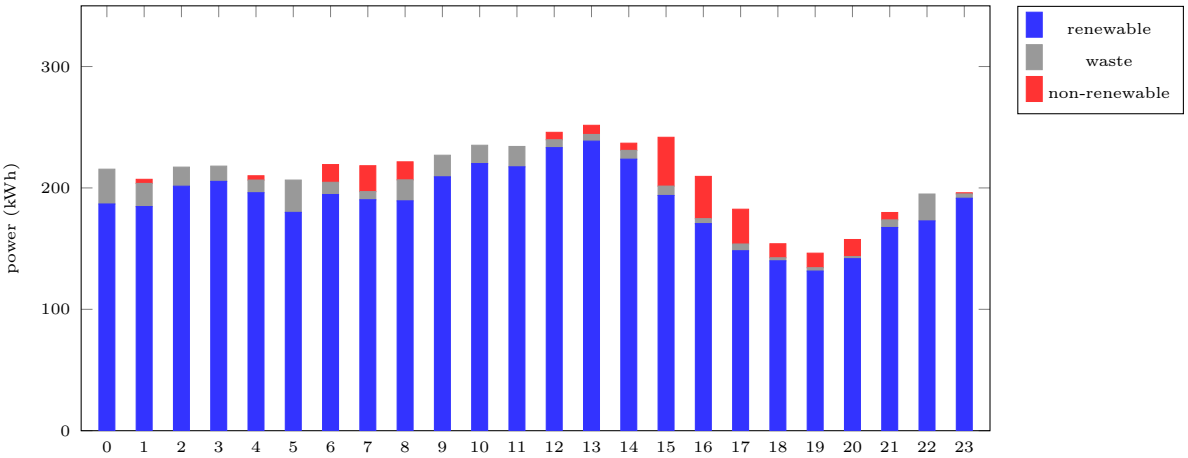

Figure 4.5: Energy consumption using the proposed model in the mixed scenario.

As the figures show, the proposed model wastes less renewable energy than the time-varying model for all three scenarios. In fact, renewable energy waste with the proposed model was 5.86% for the inexpensive model, 5.67% for the mixed model, and 10.77% for

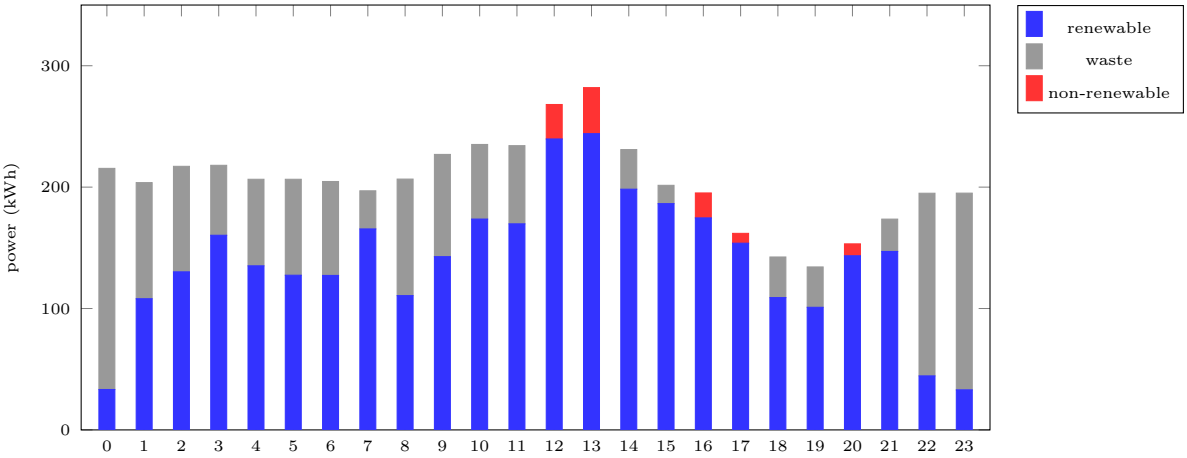

Figure 4.6: Energy consumption using the time-varying model of pricing in the mixed scenario.

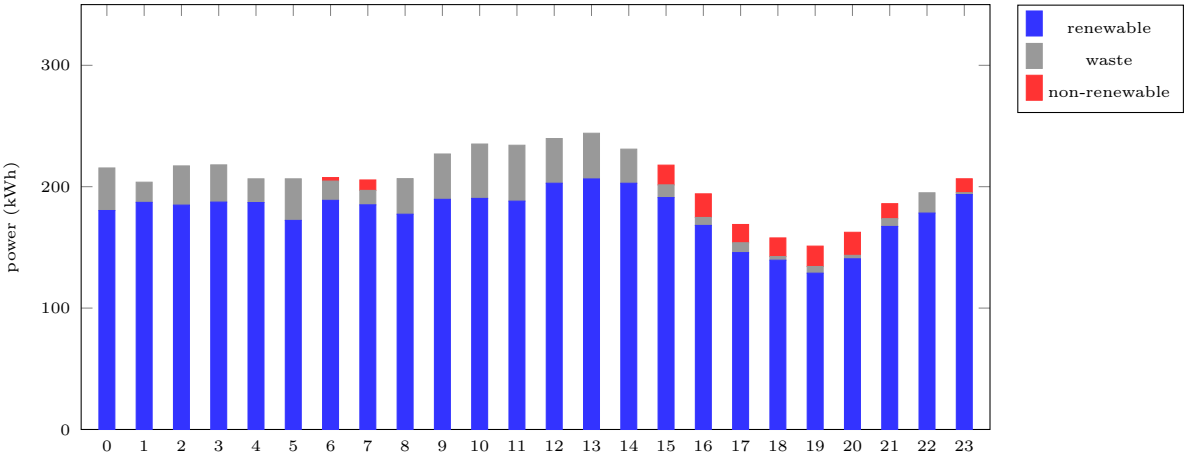

Figure 4.7: Energy consumption using the proposed model in the expensive scenario.

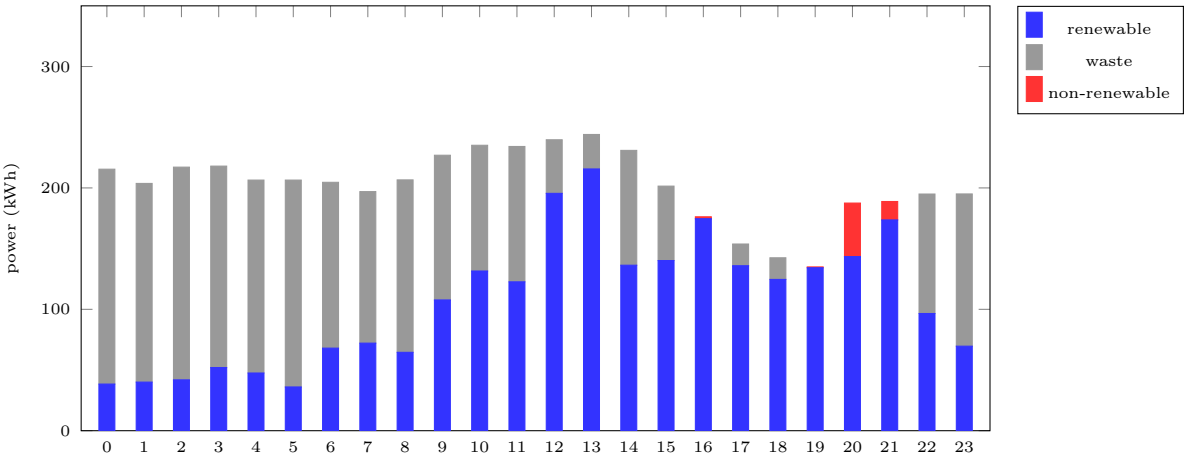

Figure 4.8: Energy consumption using the time-varying model of pricing in the expensive scenario.

the expensive model, while, for the time-varying pricing model, it was 31.09% for the inexpensive model, 30.24% for the mixed model, and 46.60% for the expensive model.

Interestingly, the proposed model suggested using less non-renewable energy than the time-varying model of pricing in the inexpensive scenario but suggested using more energy than that suggested in the mixed and expensive scenarios. The proposed model suggested using 365.81 kWh, 225.69 kWh and 137.17 kWh for the inexpensive, mixed, and expensive scenarios, respectively, with the time-varying model suggesting 480.40 kWh, 104.78 kWh and 61.49 kWh for the same scenarios. This happens because when energy is expensive, the consumers tend to use fewer appliances in the time-varying model.

Fortunately, it is possible to decrease the suggested non-renewable energy use by increasing the price of supplementary energy. In preliminary experiments, it was noticed that with higher values of r_t (such as 12 or 15) the proposed model led to a smaller consumption of non-renewable energy while still keeping better social welfare, revenue, number of appliances used and PAR when compared with the time-varying model.

Finally, as shown in Figures 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8 and summarized in Table 4.4, the proposed model has a smaller peak energy consumption and a higher mean energy consumption, and consequently a smaller PAR.

	Proposed			Time-varying		
Scenario	peak	average	peak/average	peak	average	peak/average
Inexpensive	239.58	203.51	1.18	307.35	157.82	1.95
Mixed	246.07	198.05	1.24	281.95	144.31	1.95
Expensive	207.27	184.15	1.13	215.62	109.35	1.97

Table 4.4: Mean, peak and average to peak energy consumption for both models.

4.4.4 Energy storage devices usage

Figures 4.9, 4.10 and 4.11 present the renewable and non-renewable energy consumption and the renewable energy waste for the proposed model when energy storage devices are available.

As the figures show, the renewable energy waste decreased to the level of 27.99%, 32.80%, and 45.87% of that generated by the proposed model without energy storage devices, for the inexpensive, mixed and expensive models, respectively.

Moreover, the proposed model with energy storage devices increased social welfare by 0.76%, 1.26% and 2.06% in the inexpensive, mixed and expensive scenarios, respectively, when compared with the proposed model without energy storage devices.

The amount of non-renewable energy used by the proposed model with energy storage devices decreased 1.15% for the inexpensive model, but increased 16.46% for the mixed scenario and 13.68% for the expensive scenario. This was expected since there is an increase in social welfare (the consumers use more energy), a decrease in renewable energy waste (there is less renewable energy available) and the possibility of modifying the energy usage plan by storing energy when inexpensive and using it when expensive.

As for revenue, when the proposed model was used with energy storage devices, it increased by 4.44%, 4.79%, and 4.20% for the inexpensive, mixed and expensive scenarios, respectively.

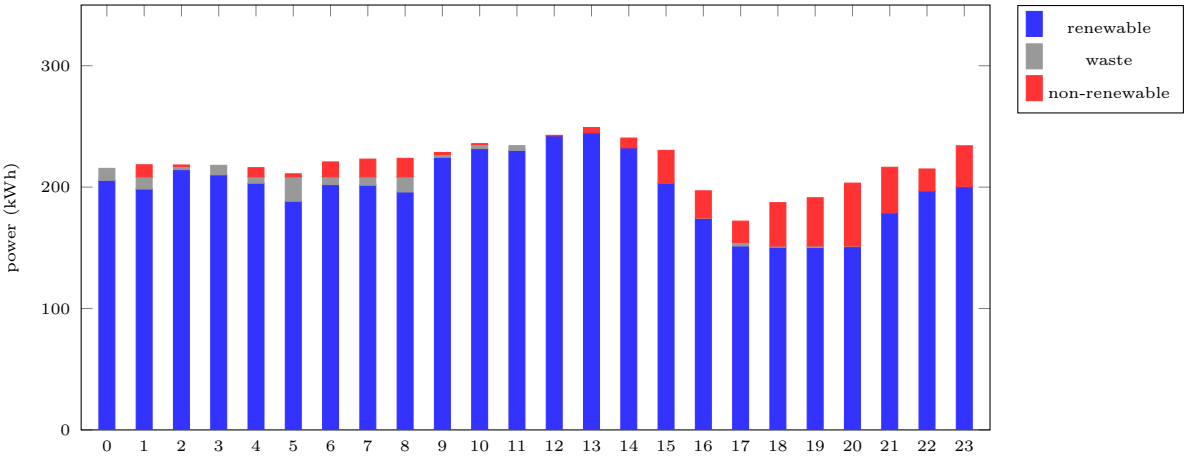

Figure 4.9: Energy consumption using the proposed model with energy storage devices in the inexpensive scenario.

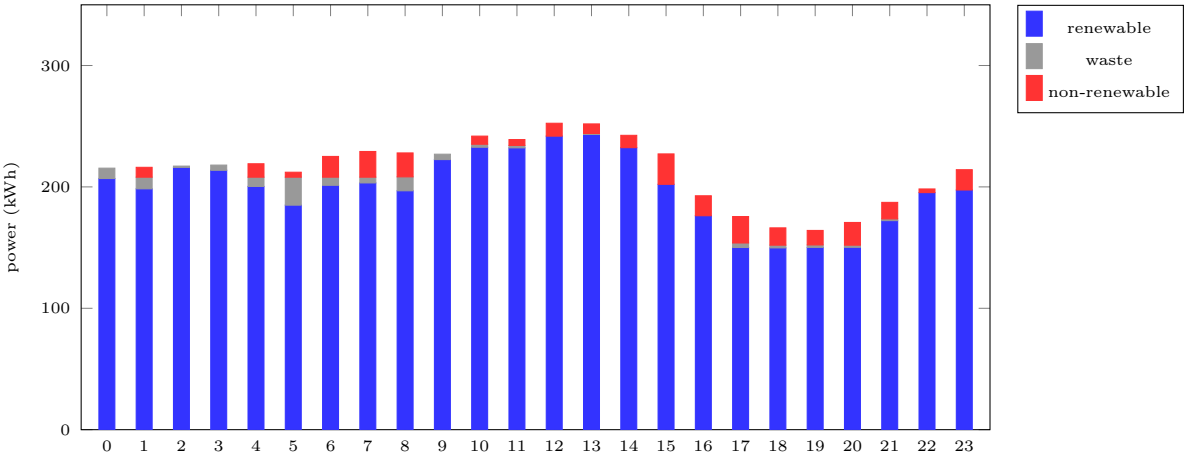

Figure 4.10: Energy consumption using the proposed model with energy storage devices in the mixed scenario.

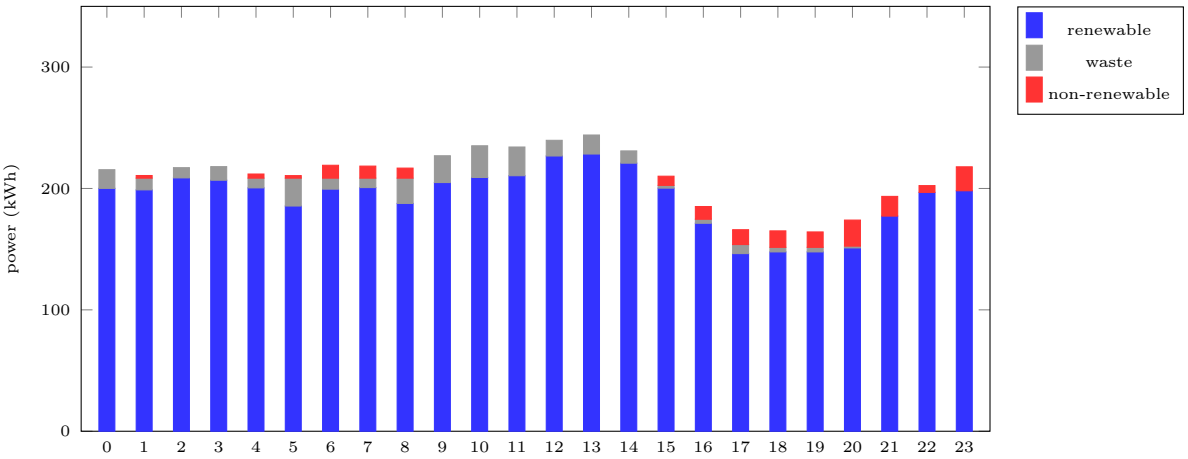

Figure 4.11: Energy consumption using the proposed model with energy storage devices in the expensive scenario.

Finally, the ratio of peak-to-average consumption for the proposed model with energy storage devices was 1.16, 1.20, and 1.15 for the inexpensive, mixed and expensive scenarios, respectively, a result similar to that obtained without energy storage devices.

4.4.5 Uncertainty of Renewable Energy Generation

Since the presented model does not directly handle the unpredictability of renewable energy sources, it must use a day-ahead prediction technique in order to approximate the capacity for the plant. As one of the techniques discussed in the literature to mitigate the impact of renewable energy generation prediction errors is to use storage units on the provider's side [31], it is proposed that the presented model should be extended in the following way: if the predicted value is greater than that observed, then the system will store renewable energy in batteries (or waste it if they are at full capacity); and, if the predicted value is smaller than that observed, then the system will first try to use the energy stored in the batteries. If they are empty, it will acquire non-renewable energy in order to fulfill the auction contracts with consumers.

In the following section, a comparison between the cases with certainty and uncertainty in renewable energy production considering the proposed model (for simplicity without using energy storage devices as described in Section 4.3.4) is presented. It was considered that there are 2 additional provider-side batteries (with the same parameter considered in Section 4.4.1) used specifically for mitigating the prediction errors.

In order to simulate uncertainty, for every period of time t, the actual amount of energy produced at time t was chosen from a normal distribution with mean w_t and standard deviation $w_t/10$ independently at random. The analysis was made considering 100 trials randomly generated.

On average, for the inexpensive scenario, the renewable energy used decreased by 0.31% (from 4518.54 kWh to 4504.43 kWh, with a standard deviation of 22.26 kWh), the non-renewable energy used increased by 3.86% (from 365.81 kWh to 379.92 kWh, with a standard deviation of 22.26 kWh) and the waste decreased by 15.32% (from 281.45 kWh to 238.34 kWh, with a standard deviation of 64.92 kWh). For the mixed scenario, on average, the renewable energy used decreased by 0.38% (from 4527.68 kWh to 4510.44 kWh, with a standard deviation of 24.30 kWh), the non-renewable energy used increased by 7.64% (from 225.69 kWh to 242.93 kWh, with a standard deviation of 24.30 kWh), and the waste decreased by 16.57% (from 272.31 kWh to 227.18 kWh, with a standard deviation of 64.78 kWh). Finally, for the expensive scenario, on average, the renewable energy used decreased by 0.05%, (from 4282.57 kWh to 4280.29 kWh, with a standard deviation of 24.30 kWh), the non-renewable energy used increased by 1.66% (from 137.17 kWh to 139.45 kWh, with a standard deviation of 6.60 kWh), and the waste decreased by 16.95% (from 517.42 kWh to 429.71 kWh, with a standard deviation of 64.78 kWh). Thus, as long as there is a reasonably accurate forecast model and storage units on the provider's side to help mitigate the errors, the presented model remains effective despite the uncertainty of the energy sources. See Figure 4.12 (in contrast with Figure 4.5) for an example of energy consumption in the mixed scenario when there is uncertainty, including the usage of storage devices for prediction error mitigation.

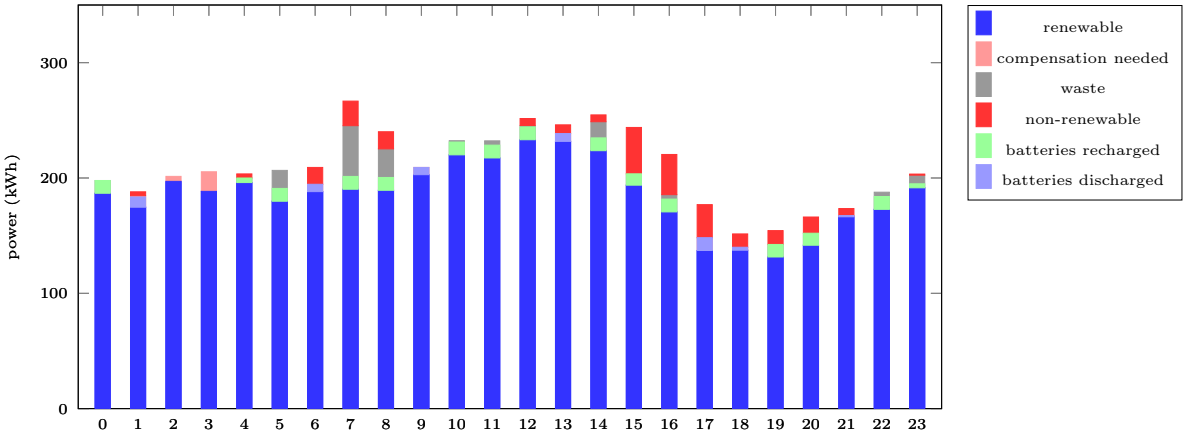

Figure 4.12: Energy consumption using the proposed model with uncertainty mitigation for a test case of the mixed scenario. The amount of external non-renewable energy needed to compensate for the prediction error is shown in light red, the amount of energy used to charge the mitigation storage units is shown in light green, and the amount of energy taken from the storage units to mitigate the prediction errors is shown in light blue.

4.4.6 Performance Evaluation

All the numerical experiments were executed in a computer with the following configuration: Intel (R) Xeon (R) CPU X3430 with 2.40GHz of clock rate, 4 cores and 8GB of RAM with Linux 64bits (Ubuntu). Both optimization problems were modeled as integer linear programs that were solved using the python2 API of Gurobi [15] version 8.1.1. All the simulations and analysis were coded in Python 2.7.

Although the problems that appear in the proposed model are NP-hard, these problems could be solved reasonably fast with a commercial ILP solver when considering an instance of a size that is common in the literature (for example, Samadi et al. [26] considers an instance with 50 consumers). For the energy allocation problem, which is solved in a centralized way, the execution time was 853, 583 and 710 seconds for the inexpensive, mixed and expensive scenarios of the proposed model (without batteries) respectively. When considering energy storage devices, the extended model was solved in 295, 214 and 456 seconds for the inexpensive, mixed and expensive scenarios, respectively.

For the bid computations, several ILP problems were solved, i.e. one for each consumer and usage plan. The total time to solve all the bid computation problems was 1494, 1785 and 2193 seconds for the inexpensive, mixed and expensive scenarios respectively. The average computing time per consumer was about 14, 17 and 21 seconds for the inexpensive, mixed and expensive scenarios respectively. The computation times for the model with energy storage devices was close to its counterpart, in which it was 14, 16 and 20 seconds for the inexpensive, mixed and expensive scenarios, respectively.

4.5 Conclusions

This paper has presented a new model for smart pricing for demand-side management by using energy consumption scheduling devices for renewable energy. The proposed model

is based on an auction where consumers submit bids for a set of possible energy usage plans designed by the utility company and then allocates energy according to the power plant production capacity. A model for generating consumers' bids which can consider must-run, non-interruptible, interruptible appliances and other variations is presented. Both models are extended to consider the usage of energy storage devices.

The main advantage of the proposed model is that it does not make any assumptions about how consumers will derive value from energy consumption nor does it depend on how bids are computed by consumers. That is, the model presented for generating consumers bids is only a suggestion, and can be replaced by any other model with the same purpose. This simplifies the perspective of the consumer about the system since a bid represents how much a consumer is willing to pay for a specific usage plan. Moreover, this allows for the creation of models for other consumers such as universities, hospitals or industries, as long as one can define how those consumers allocate energy from a given energy usage plan and how they obtain value from energy usage.

Numerical examples show that the proposed model offers great improvement when compared with the time-varying model of pricing. In fact, it was able to increase social welfare, as well as the mean number of appliances used by a consumer and revenue, while reducing the renewable energy waste and peak-to- average ratio. Even though more non-renewable energy is consumed, this can be reduced by increasing the price of supplementary energy, so that the values are lower than those obtained using the time-varying pricing model while still obtaining better social welfare and high revenue. It has also been shown that energy storage devices can improve social welfare while reducing energy waste. Finally, the numerical simulations give us confidence that the models could be implemented efficiently with existing ILP solvers both for the energy allocation and bid computation problems.

As future work, the proposed model could be extended to better consider the unpredictable nature of renewable energy generation. In the experiments, it is shown that this unpredictability does not significantly affect the proposed model and that these effects can be mitigated using additional batteries. Nonetheless, a solution that specifically considers the unpredictability could obtain better results.

Acknowledgements

This research was partially supported by grants 2013/21744-8, 2015/11937-9, 2016/01860-1 and 2016/23552-7 São Paulo Research Foundation (FAPESP) and grants 311499/2014-7, 425340/2016-3, 477692/2012-5 and 311499/2014-7, 308689/2017-8, National Council for Scientific and Technological Development (CNPq).

References

[1] Adil Ahmed and Muhammad Khalid. A review on the selected applications of forecasting models in renewable power systems. *Renewable and Sustainable Energy Reviews*, 100:9–21, 2019.

[2] Kenneth Joseph Arrow. The Property Rights Doctrine and Demand Revelation Under Incomplete Information, 1977.

[3] I Atzeni, L G Ordonez, G Scutari, D P Palomar, and J R Fonollosa. Demand-Side Management via Distributed Energy Generation and Storage Optimization. *IEEE Transactions on Smart Grid*, 4(2):866–876, September 2013.

[4] I. Atzeni, L. G. Ordóñez, G. Scutari, D. P. Palomar, and J. R. Fonollosa. Noncooperative and cooperative optimization of distributed energy generation and storage in the demand-side of the smart grid. *IEEE Transactions on Smart Grid*, 61(10): 2454–2472, 05 2013. ISSN 1053-587X. doi: 10.1109/TSP.2013.2248002.

[5] I. Atzeni, L. G. Ordóñez, G. Scutari, D. P. Palomar, and J. R. Fonollosa. Noncooperative day-ahead bidding strategies for demand-side expected cost minimization with real-time adjustments: A gnep approach. *IEEE Transactions on Signal Processing*, 62(9):2397–2412, May 2014. ISSN 1053-587X. doi: 10.1109/TSP.2014.2307835.

[6] S. Belhaiza and U. Baroudi. A game theoretic model for smart grids demand management. *IEEE Transactions on Smart Grid*, 6(3):1386–1393, May 2015. ISSN 1949-3053. doi: 10.1109/TSG.2014.2376632.

[7] S. Bu and F. R. Yu. A game-theoretical scheme in the smart grid with demand-side management: Towards a smart cyber-physical power infrastructure. *IEEE Transactions on Emerging Topics in Computing*, 1(1):22–32, June 2013. ISSN 2168-6750. doi: 10.1109/TETC.2013.2273457.

[8] H. Chen, Y. Li, R. H. Y. Louie, and B. Vucetic. Autonomous demand side management based on energy consumption scheduling and instantaneous load billing: An aggregative game approach. *IEEE Transactions on Smart Grid*, 5(4):1744–1754, July 2014. ISSN 1949-3053. doi: 10.1109/TSG.2014.2311122.

[9] Edward H Clarke. Multipart Pricing of Public Goods. *Public Choice*, 11:17–33, September 1971.

[10] Claude d'. Aspremont and Louis-André Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11(1):25–45, February 1979. doi: 10.1016/0047-2727(79)90043-4.

[11] European Commission. 2020 climate and energy package, 2008. `https://ec.europa.eu/clima/policies/strategies/2020_en#tab-0-0`.

[12] European Technology & Innovation Platforms. The etip smart networks for energy transition, 2018. `https://www.etip-snet.eu/about/etip-snet/`.

[13] M Fahrioglu and F L Alvarado. Designing incentive compatible contracts for effective demand management. *IEEE Transactions on Power Systems*, 15(4):1255–1260, Nov 2000. doi: 10.1109/59.898098.

[14] T Groves. Incentives in Teams. *Econometrica*, 41(4):617–631, July 1973.

[15] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2015. URL `http://www.gurobi.com`.

[16] A. A. Khan, M. H. Rehmani, and M. Reisslein. Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols. *IEEE Communications Surveys Tutorials*, 18(1):860–898, Firstquarter 2016. ISSN 1553-877X. doi: 10.1109/COMST.2015.2481722.

[17] C. Li, X. Yu, W. Yu, G. Chen, and J. Wang. Efficient computation for sparse load shifting in demand side management. *IEEE Transactions on Smart Grid*, 8(1):250–261, Jan 2017. ISSN 1949-3053. doi: 10.1109/TSG.2016.2521377.

[18] Y. Liu, C. Yuen, S. Huang, N. Ul Hassan, X. Wang, and S. Xie. Peak-to-average ratio constrained demand-side management with consumer's preference in residential smart grid. *IEEE Journal of Selected Topics in Signal Processing*, 8(6):1084–1097, Dec 2014. ISSN 1932-4553. doi: 10.1109/JSTSP.2014.2332301.

[19] T. Logenthiran, D. Srinivasan, and T. Z. Shun. Demand side management in smart grid using heuristic optimization. *IEEE Transactions on Smart Grid*, 3(3):1244–1252, Sept 2012. ISSN 1949-3053. doi: 10.1109/TSG.2012.2195686.

[20] Jinghuan Ma, Jun Deng, Lingyang Song, and Zhu Han. Incentive Mechanism for Demand Side Management in Smart Grid Using Auction. *IEEE Transactions on Smart Grid*, 5(3):1379–1388, May 2014.

[21] A. H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE Transactions on Smart Grid*, 1(3):320–331, 12 2010. ISSN 1949-3053. doi: 10.1109/TSG.2010.2089069.

[22] K-H Ng and Gerald B Sheble. Direct load control-a profit-based load management using linear programming. *IEEE Transactions on Power Systems*, 13(2):688–694, 1998.

[23] M. H. Rehmani, M. Reisslein, A. Rachedi, M. Erol-Kantarci, and M. Radenkovic. Integrating renewable energy resources into the smart grid: Recent developments in information and communication technologies. *IEEE Transactions on Industrial Informatics*, 14(7):2814–2825, 2018. doi: 10.1109/TII.2018.2819169.

[24] Ye Ren, P.N. Suganthan, and N. Srikanth. Ensemble methods for wind and solar power forecasting—a state-of-the-art review. *Renewable and Sustainable Energy Reviews*, 50:82 – 91, 2015. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2015.04.081. URL `http://www.sciencedirect.com/science/article/pii/S1364032115003512`.

[25] W. Saad, Z. Han, H. V. Poor, and T. Basar. Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications. *IEEE Signal Processing Magazine*, 29(5):86–105, Sept 2012. ISSN 1053-5888. doi: 10.1109/MSP.2012.2186410.

[26] Pedram Samadi, Hamed Mohsenian-Rad, Robert Schober, and Vincent W S Wong. Advanced Demand Side Management for the Future Smart Grid Using Mechanism Design. *IEEE Transactions on Smart Grid*, 3(3):1170–1180, September 2012.

[27] G. A. Shah, V. C. Gungor, and O. B. Akan. A cross-layer qos-aware communication framework in cognitive radio sensor networks for smart grid applications. *IEEE Transactions on Industrial Informatics*, 9(3):1477–1485, 08 2013. ISSN 1551-3203. doi: 10.1109/TII.2013.2242083.

[28] H. M. Soliman and A. Leon-Garcia. Game-theoretic demand-side management with storage devices for the future smart grid. *IEEE Transactions on Smart Grid*, 5(3): 1475–1485, May 2014. ISSN 1949-3053. doi: 10.1109/TSG.2014.2302245.

[29] S. S. Soman, H. Zareipour, O. Malik, and P. Mandal. A review of wind power and wind speed forecasting methods with different time horizons. In *North American Power Symposium 2010*, pages 1–8, Sep. 2010. doi: 10.1109/NAPS.2010.5619586.

[30] Dipti Srinivasan, Sanjana Rajgarhia, Bharat Menon Radhakrishnan, Anurag Sharma, and H.P. Khincha. Game-theory based dynamic pricing strategies for demand side management in smart grids. *Energy*, 126:132 – 143, 2017. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2016.11.142. URL `http://www.sciencedirect.com/science/article/pii/S0360544216317996`.

[31] Edward Baleke Ssekulima, Muhammad Bashar Anwar, Amer Al Hinai, and Mohamed Shawky El Moursi. Wind speed and solar irradiance forecasting techniques for enhanced renewable energy integration with the grid: a review. *IET Renewable Power Generation*, 10(7):885–989, 2016.

[32] Terna S.p.A. - Rete Elettrica Nazionale. Forecast and actual generation of wind power (so called intermittent generation). `https://www.terna.it/en-gb/sistemaelettrico/transparencyreport/generation/forecastandactualgeneration.aspx`, 2006. Acessed: 2016-11-06.

[33] William Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, March 1961.

[34] K. Wang, H. Li, S. Maharjan, Y. Zhang, and S. Guo. Green energy scheduling for demand side management in the smart grid. *IEEE Transactions on Green Communications and Networking*, PP(99):1–1, 2018. doi: 10.1109/TGCN.2018.2797533.

[35] P. Yang, G. Tang, and A. Nehorai. A game-theoretic approach for optimal time-of-use electricity pricing. *IEEE Transactions on Power Systems*, 28(2):884–892, May 2013. ISSN 0885-8950. doi: 10.1109/TPWRS.2012.2207134.

[36] F. Ye, Y. Qian, and R. Q. Hu. A real-time information based demand-side management system in smart grid. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):329–339, Feb 2016. ISSN 1045-9219. doi: 10.1109/TPDS.2015.2403833.

[37] J. Zazo, S. Zazo, and S. Valcarcel Macua. Robust worst-case analysis of demand-side management in smart grids. *IEEE Transactions on Smart Grid*, 8(2):662–673, March 2017. ISSN 1949-3053. doi: 10.1109/TSG.2016.2559583.

[38] Z. Zhu, J. Tang, S. Lambotharan, W. H. Chin, and Z. Fan. An integer linear programming and game theory based optimization for demand-side management in smart grid. In *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, pages 1205–1210, Dec 2011. doi: 10.1109/GLOCOMW.2011.6162372.

Chapter 5

Concluding Remarks

In this work, we study algorithms and mathematical models for some variants of packing problems. The thesis is presented as a collection of articles: one already published in an international journal; and two recently submitted. Each article consists of a project on a specific problem, often using different approaches to solve them.

The first project, as discussed in Chapter 2, consisted of the colored bin packing problem. This problem had already been extensively studied in the literature, both with online and approximation algorithms. However, to the best of our knowledge, there were no experimental results in the literature for it. Our main result was the enhanced arc flow formulation that used a compact multi-graph and preserved the same strong linear relaxation bound of the generalization of the model of Valério de Carvalho for the CBPP. To contextualize the quality of this model, we wanted to compare it with branch-and-price models, but since a fast algorithm for pricing in a Gilmore-Gomory formulation was not available, it was unlikely that such a model would be competitive. The VRPSolver framework was then used to propose two competitive set-partition models to complement our enhanced arc flow formulation.

Although heuristics and metaheuristics for the CBPP are not presented in this thesis, they have been considered in another work, in collaboration with an undergraduate student, during the project. Since many different metaheuristics can be applied to this problem, this area presents many possibilities for future work. Furthermore, we could investigate fast algorithms for the colored variant of the knapsack problem, making a Gilmore-Gomory-style set-partition formulation viable. Such formulation could even be improved with the study of dual feasible functions, problem-specific cuts, and other similar techniques.

The second project, as discussed in Chapter 3, consisted of the bin packing problem with scenarios. To the best of our knowledge, this recently proposed problem had only been studied from an online perspective, so the main contributions of this work are the first approximation algorithms and the first offline computational study for the BPPS. The set-cover formulation, usually associated with these types of packing problems, had limited success because the pricing sub-problems were proven to be strongly NP-hard even without any branching constraints. To circumvent this situation, we solved the sub-problems using a MILP model and employed dual feasible functions to improve the quality of the lower bounds in hopes of a faster convergence. Furthermore, we proposed a VNS

metaheuristic to provide a good initial solution for the exact model. We believe these results will serve as a good foundation for further experimental works on this problem in the future.

Additionally, arc flow formulations could also be considered for the BPPS. However, it is unclear whether we can represent items appearing simultaneously in multiple scenarios without adding a dimension per scenario. This approach has been used to model the vector bin packing problem using an arc flow formulation, but the number of variables significantly increases with the number of dimensions. A similar approach for the BPPS would be limited to instances with a lower level of uncertainty. Alternative metaheuristics could be considered for giving an initial solution to the problem, as well as other neighborhood structures for the proposed VNS framework. These directions could also be considered in future works.

The last project, as discussed in Chapter 4, consisted of a smart pricing scheme for demand-side management in smart grids. In this work, we used a game-theoretical approach to an application involving connected electrical networks. The main contribution comes from the multidisciplinary nature of the modeling approach, bringing game theory, combinatorial optimization, and network systems together to propose a solution. By modeling both the energy allocation and bid calculation as packing problems, we were able to perform simulations of the proposed auction scheme by solving both problems as MILPs. This approach also enabled a smooth integration of the use of batteries on both sides by slightly altering the objective functions and knapsack constraints. Results of the simulations show that the proposed model is effective at reducing the peak-to-average consumption ratio, an important metric for energy grids.

Since the targeted application of the proposed model was aimed at small communities, the simulations included only a few hundred consumers, which was ideal for the proposed MILP models. If larger communities, of thousands of consumers, were considered, we would have to study an alternative algorithm for the energy allocation problem that would scale better with the size of the community. Furthermore, we only considered household consumers in the bidding computation problem. If we were to account for commercial and industrial consumers, our models would have to include specific rules to accommodate these new profiles. These types of consumers also have a larger number of appliances to be allocated to their plans, so the proposed bidding computation algorithm would need to scale properly. These extensions could be considered in future works.

For all problems studied in this thesis, we provided a benchmark set of instances along with the implementation of all exact and heuristic algorithms proposed. We hope that we were able to establish a solid foundation for experimental results on each of these problems so that new algorithms developed in the future can build upon these results. All the benchmark data and code can be made available for researchers under reasonable personal request.

References

[1] Nurşen Aydın, İbrahim Muter, and Ş İlker Birbil. Multi-objective temporal bin packing problem: An application in cloud computing. *Computers & Operations Research*, 121:104959, 2020.

[2] Yulle Borges, Thiago de Queiroz, Vinícius Lima, Flavio Miyazawa, and Lehilton Pedrosa. Um esquema de aproximação para um problema de empacotamento com cenários. In *Anais do IV Encontro de Teoria da Computação*. SBC, 2019.

[3] Yulle G. F. Borges and Rafael C. S. Schouery. Modelos pseudo polinomiais para o problema do empacotamento colorido. In *Anais do V Encontro de Teoria da Computação*, pages 37–40. SBC, 2020.

[4] Yulle G. F. Borges, Rafael C. S. Schouery, Flavio K. Miyazawa, Fabrizio Granelli, Nelson L. S. da Fonseca, and Lucas P. Melo. Smart energy pricing for demand-side management in renewable energy smart grids. *International Transactions in Operational Research*, 27(6):2760–2784, 2020.

[5] Filipe Brandao and João Pedro Pedroso. Bin packing and related problems: general arc-flow formulation with graph compression. *Computers & Operations Research*, 69: 56–67, 2016.

[6] Mirsad Buljubašić and Michel Vasquez. Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. *Computers & Operations Research*, 76:12–21, 2016.

[7] V. Chvátal. *Linear Programming*. W. H. Freeman and Company, 1980.

[8] Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. Bin packing approximation algorithms: survey and classification. In *Handbook of combinatorial optimization*, pages 455–531. Springer, 2013.

[9] Renan F. F. da Silva, Yulle G. F. Borges, and Rafael C. S. Schouery. Heurísticas para o problema do empacotamento colorido. In *Anais do VI Encontro de Teoria da Computação*, pages 34–37. SBC, 2021.

[10] Renan F. F. da Silva, Yulle G. F. Borges, and Rafael C. S. Schouery. Heurísticas para o problema do empacotamento colorido. In *Anais do Simpósio Brasileiro de Pesquisa Operacional 2022*. Galoá, 2022.

[11] Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

[12] European Technology & Innovation Platforms. The ETIP smart networks for energy transition, 2018. `https://www.etip-snet.eu/about/etip-snet/`.

[13] Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133, 1995.

[14] W. Fernandez de La Vega and George S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[15] Krzysztof Fleszar and Khalil S. Hindi. New heuristics for one-dimensional bin-packing. *Computers & operations research*, 29(7):821–839, 2002.

[16] Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

[17] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Operations research*, 9:849–859, 1961.

[18] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[19] José Fernando Gonçalves and Mauricio G. C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[20] Jatinder N. D. Gupta and Johnny C. Ho. A new heuristic algorithm for the one-dimensional bin-packing problem. *Production planning & control*, 10(6):598–603, 1999.

[21] Fatma Gzara, Samir Elhedhli, and Burak C. Yildiz. The pallet loading problem: Three-dimensional bin packing with practical constraints. *European Journal of Operational Research*, 287(3):1062–1074, 2020.

[22] E. B. C. H. Hopper and Brian C. H. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1):34–57, 2001.

[23] University of Campinas Institute of Computing. Alternative thesis format. `https://ic.unicamp.br/en/pos-graduacao/informacao-para-docentes-e-pos-graduandos/vida-academica-da-pos-graduacao/formato-alternativo-de-tese/`. Accessed: 2023-05-07.

[24] David S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology, 1973.

[25] Julia Kallrath, Steffen Rebennack, Josef Kallrath, and Rüdiger Kusche. Solving real-world cutting stock-problems in the paper industry: Mathematical approaches, experience and challenges. *European Journal of Operational Research*, 238(1):374–389, 2014.

[26] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 312–320. IEEE, 1982.

[27] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[28] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[29] Tayfun Kucukyilmaz and Hakan Ezgi Kiziloz. Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. *Computers & Industrial Engineering*, 125:157–170, 2018.

[30] Andrea Lodi, Silvano Martello, Michele Monaci, and Daniele Vigo. Two-dimensional bin packing problems. *Paradigms of combinatorial optimization: Problems and new approaches*, pages 107–129, 2014.

[31] Kok-Hua Loh, Bruce Golden, and Edward Wasil. Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers & Operations Research*, 35(7):2283–2291, 2008.

[32] V. Maniezzo, T. Stützle, and S. Voß, editors. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems*. Springer US, 2010. doi: 10.1007/978-1-4419-1306-7.

[33] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[34] The International Federation of Operational Research Societies. What is OR?, 2023. URL `https://www.ifors.org/what-is-or/`. Accessed: 2023-05-07.

[35] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization*, volume 24. Prentice Hall Englewood Cliffs, 1982.

[36] Célia Paquay, Sabine Limbourg, Michaël Schyns, and José Fernando Oliveira. MIP-based constructive heuristics for the three-dimensional bin packing problem with transportation constraints. *International Journal of Production Research*, 56(4): 1581–1592, 2018.

[37] Kyung Tae Park, Jun-hyung Ryu, Ho-Kyung Lee, and In-Beum Lee. Development of a heuristic algorithm for cutting stock problems in flat glass production processes. *Journal of chemical engineering of Japan*, 45(3):219–227, 2012.

[38] Guido Perboli, Luca Gobbato, and Francesca Perfetti. Packing problems in transportation and supply chain: new problems and trends. *Procedia-Social and Behavioral Sciences*, 111:672–681, 2014.

[39] Guntram Scheithauer. *Introduction to cutting and packing optimization: Problems, modeling approaches, solution methods*, volume 263. Springer, 2017.

[40] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[41] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98: 4th International Conference, CP98 Pisa, Italy, October 26–30, 1998 Proceedings 4*, pages 417–431. Springer, 1998.

[42] Alessio Trivella and David Pisinger. Bin-packing problems with load balancing and stability constraints. *INFORMS Transportation and Logistics Society*, 2017, 2017.

[43] J. M. Valério De Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659, 1999.

[44] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, 3(2):111–130, 1994. doi: 10.1007/BF01300970.

[45] Ramiro Varela, Camino R. Vela, Jorge Puente, María Sierra, and Inés González-Rodríguez. An effective solution for a real cutting stock problem in manufacturing plastic rolls. *Annals of Operations Research*, 166(1):125–146, 2009.

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

[47] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007. ISSN 0377-2217. doi: 10.1016/j.ejor.2005.12.047. URL `http://www.sciencedirect.com/science/article/pii/S037722170600292X`.

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

[49] Laurence A. Wolsey. *Integer programming*. John Wiley & Sons, 2020.

Appendix A

Publisher's Permission To Reproduce Article

This appendix contains a copy of the John Wiley and Sons (publisher) authorization for the reproduction of the paper titled "Smart energy pricing for demand-side management in renewable energy smart grids", published in International Transactions in Operational Research, as Chapter 4 of this thesis.

JOHN WILEY AND SONS LICENSE
TERMS AND CONDITIONS

May 12, 2023

This Agreement between Mr. Yulle Borges ("You") and John Wiley and Sons ("John Wiley and Sons") consists of your license details and the terms and conditions provided by John Wiley and Sons and Copyright Clearance Center.

License Number	5546550686937
License date	May 12, 2023
Licensed Content Publisher	John Wiley and Sons
Licensed Content Publication	International Transactions in Operational Research
Licensed Content Title	Smart energy pricing for demand-side management in renewable energy smart grids
Licensed Content Author	Lucas P. Melo, Nelson L. S. da Fonseca, Fabrizio Granelli, et al
Licensed Content Date	Nov 5, 2019
Licensed Content Volume	27
Licensed Content Issue	6
Licensed Content Pages	25
Type of use	Dissertation/Thesis

Requestor type	Author of this Wiley article
Format	Electronic
Portion	Full article
Will you be translating?	No
Title	Exact and Heuristic Algorithms for Packing Problems: Color Alternation, Uncertainty, and in Smart Grids
Institution name	University of Campinas
Expected presentation date	Jul 2023
Order reference number	Autorização final
Requestor Location	Mr. Yulle Borges Rua Pedro Vieira da Silva, 144 Campinas, São Paulo 13080-570 Brazil Attn: Mr. Yulle Borges
Publisher Tax ID	EU826007151
Total	0.00 USD

Terms and Conditions

TERMS AND CONDITIONS

This copyrighted material is owned by or exclusively licensed to John Wiley & Sons, Inc. or one of its group companies (each a"Wiley Company") or handled on behalf of a society with which a Wiley Company has exclusive publishing rights in relation to a particular work (collectively "WILEY"). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the billing and payment terms and conditions established by the Copyright Clearance Center Inc., ("CCC's Billing and Payment terms and conditions"), at the time that

you opened your RightsLink account (these are available at any time at http://myaccount.copyright.com).

Terms and Conditions

- The materials you have requested permission to reproduce or reuse (the "Wiley Materials") are protected by copyright.

- You are hereby granted a personal, non-exclusive, non-sub licensable (on a stand-alone basis), non-transferable, worldwide, limited license to reproduce the Wiley Materials for the purpose specified in the licensing process. This license, **and any CONTENT (PDF or image file) purchased as part of your order,** is for a one-time use only and limited to any maximum distribution number specified in the license. The first instance of republication or reuse granted by this license must be completed within two years of the date of the grant of this license (although copies prepared before the end date may be distributed thereafter). The Wiley Materials shall not be used in any other manner or for any other purpose, beyond what is granted in the license. Permission is granted subject to an appropriate acknowledgement given to the author, title of the material/book/journal and the publisher. You shall also duplicate the copyright notice that appears in the Wiley publication in your use of the Wiley Material. Permission is also granted on the understanding that nowhere in the text is a previously published source acknowledged for all or part of this Wiley Material. Any third party content is expressly excluded from this permission.

- With respect to the Wiley Materials, all rights are reserved. Except as expressly granted by the terms of the license, no part of the Wiley Materials may be copied, modified, adapted (except for minor reformatting required by the new Publication), translated, reproduced, transferred or distributed, in any form or by any means, and no derivative works may be made based on the Wiley Materials without the prior permission of the respective copyright owner.**For STM Signatory Publishers clearing permission under the terms of the STM Permissions Guidelines only, the terms of the license are extended to include subsequent editions and for editions in other languages, provided such editions are for the work as a whole in situ and does not involve the separate exploitation of the permitted figures or extracts,** You may not alter, remove or suppress in any manner any copyright, trademark or other notices displayed by the Wiley Materials. You may not license, rent, sell, loan, lease, pledge, offer as security, transfer or assign the Wiley Materials on a stand-alone basis, or any of the rights granted to you hereunder to any other person.

- The Wiley Materials and all of the intellectual property rights therein shall at all times remain the exclusive property of John Wiley & Sons Inc, the Wiley Companies, or their respective licensors, and your interest therein is only that of having possession of and the right to reproduce the Wiley Materials pursuant to Section 2 herein during the continuance of this Agreement. You agree that you own no right, title or interest in or to the Wiley Materials or any of the intellectual property rights therein. You shall have no rights hereunder other than the license as provided for above in Section 2. No right, license or interest to any trademark, trade name, service mark or other branding ("Marks") of WILEY or its licensors is granted hereunder, and you agree that you shall not assert any such right, license or interest with respect thereto

- NEITHER WILEY NOR ITS LICENSORS MAKES ANY WARRANTY OR REPRESENTATION OF ANY KIND TO YOU OR ANY THIRD PARTY, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO THE MATERIALS OR THE

ACCURACY OF ANY INFORMATION CONTAINED IN THE MATERIALS, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY, ACCURACY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, USABILITY, INTEGRATION OR NON-INFRINGEMENT AND ALL SUCH WARRANTIES ARE HEREBY EXCLUDED BY WILEY AND ITS LICENSORS AND WAIVED BY YOU.

- WILEY shall have the right to terminate this Agreement immediately upon breach of this Agreement by you.

- You shall indemnify, defend and hold harmless WILEY, its Licensors and their respective directors, officers, agents and employees, from and against any actual or threatened claims, demands, causes of action or proceedings arising from any breach of this Agreement by you.

- IN NO EVENT SHALL WILEY OR ITS LICENSORS BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR ENTITY FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, INDIRECT, EXEMPLARY OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, PROVISIONING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

- Should any provision of this Agreement be held by a court of competent jurisdiction to be illegal, invalid, or unenforceable, that provision shall be deemed amended to achieve as nearly as possible the same economic effect as the original provision, and the legality, validity and enforceability of the remaining provisions of this Agreement shall not be affected or impaired thereby.

- The failure of either party to enforce any term or condition of this Agreement shall not constitute a waiver of either party's right to enforce each and every term and condition of this Agreement. No breach under this agreement shall be deemed waived or excused by either party unless such waiver or consent is in writing signed by the party granting such waiver or consent. The waiver by or consent of a party to a breach of any provision of this Agreement shall not operate or be construed as a waiver of or consent to any other or subsequent breach by such other party.

- This Agreement may not be assigned (including by operation of law or otherwise) by you without WILEY's prior written consent.

- Any fee required for this permission shall be non-refundable after thirty (30) days from receipt by the CCC.

- These terms and conditions together with CCC's Billing and Payment terms and conditions (which are incorporated herein) form the entire agreement between you and WILEY concerning this licensing transaction and (in the absence of fraud) supersedes all prior agreements and representations of the parties, oral or written. This Agreement

may not be amended except in writing signed by both parties. This Agreement shall be binding upon and inure to the benefit of the parties' successors, legal representatives, and authorized assigns.

- In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall prevail.

- WILEY expressly reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

- This Agreement will be void if the Type of Use, Format, Circulation, or Requestor Type was misrepresented during the licensing process.

- This Agreement shall be governed by and construed in accordance with the laws of the State of New York, USA, without regards to such state's conflict of law rules. Any legal action, suit or proceeding arising out of or relating to these Terms and Conditions or the breach thereof shall be instituted in a court of competent jurisdiction in New York County in the State of New York in the United States of America and each party hereby consents and submits to the personal jurisdiction of such court, waives any objection to venue in such court and consents to service of process by registered or certified mail, return receipt requested, at the last known address of such party.

WILEY OPEN ACCESS TERMS AND CONDITIONS

Wiley Publishes Open Access Articles in fully Open Access Journals and in Subscription journals offering Online Open. Although most of the fully Open Access journals publish open access articles under the terms of the Creative Commons Attribution (CC BY) License only, the subscription journals and a few of the Open Access Journals offer a choice of Creative Commons Licenses. The license type is clearly identified on the article.

The Creative Commons Attribution License

The Creative Commons Attribution License (CC-BY) allows users to copy, distribute and transmit an article, adapt the article and make commercial use of the article. The CC-BY license permits commercial and non-

Creative Commons Attribution Non-Commercial License

The Creative Commons Attribution Non-Commercial (CC-BY-NC)License permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.(see below)

Creative Commons Attribution-Non-Commercial-NoDerivs License

The Creative Commons Attribution Non-Commercial-NoDerivs License (CC-BY-NC-ND) permits use, distribution and reproduction in any medium, provided the original work is properly cited, is not used for commercial purposes and no modifications or adaptations are made. (see below)

Use by commercial "for-profit" organizations

Use of Wiley Open Access articles for commercial, promotional, or marketing purposes requires further explicit permission from Wiley and will be subject to a fee.

Further details can be found on Wiley Online Library
http://olabout.wiley.com/WileyCDA/Section/id-410895.html

Other Terms and Conditions:

v1.10 Last updated September 2015

Questions? customercare@copyright.com.