Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

OTTORATO DI RICERCA IN
INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 35

Settore Concorsuale: **01/A6 - RICERCA OPERATIVA**

Settore Scientifico Disciplinare: **MAT/09 - RICERCA OPERATIVA**

Models and Algorithms for Real-World Optimization Problems

Presentata da: Antonio Punzo

Coordinatore Dottorato **Supervisore**

Prof. Michele Monaci **Prof. Daniele Vigo**

Prof. Michele Monaci

Esame finale anno 2023

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

CICLO XXXV
Settore Concorsuale: 01/A6 - RICERCA OPERATIVA
Settore Scientifico Disciplinare: MAT/09 - RICERCA OPERATIVA

Models and Algorithms for Real-World Optimization Problems

Author:
Antonio PUNZO

Supervisor:
Dr. Daniele VIGO

Co-Supervisor:
Dr. Michele MONACI

Esame Finale anno 2023

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Abstract

DIPARTIMENTO DI INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE "GUGLIELMO MARCONI"- DEI

Ingegneria Biomedica, Elettrica e dei Sistemi
(Curriculum Ricerca Operativa)

Models and Algorithms for Real-World Optimization Problems

by Antonio PUNZO

This thesis deals with efficient solution of optimization problems of practical interest.

The first part of the thesis deals with bin packing problems. The bin packing problem (BPP) is one of the oldest and most fundamental combinatorial optimization problems. The problem is defined as follow: Given a set of n items with weight $w_j, j = 1 \ldots n$, and an unbounded set of identical bins with capacity c, assign each item to a bin so that the sum of the weights of the items assigned to a bin does not exceed c and the number of used bins is minimized.

The bin packing problem and its generalizations arise often in real-world applications, from manufacturing industry, logistics and transportation of goods, and scheduling.

After an introductory chapter, I will present two applications of two of the most natural extensions of the bin packing: Chapter 2 will be dedicated to an application of bin packing in two dimension to a problem of scheduling a set of computational tasks on a computer cluster, while Chapter 3 deals with the generalization of BPP in three dimensions that arise frequently in logistic and transportation, often complemented with additional constraints on the placement of items and characteristics of the solution, like, for example, guarantees on the stability of the items, to avoid potential damage to the transported goods, on the distribution of the total weight of the bins, and on compatibility with loading and unloading operations.

The second part of the thesis, and in particular Chapter 4 considers the Transmission Expansion Problem (TEP), where an electrical transmission grid must be expanded so as to satisfy future energy demand at the minimum cost, while maintaining some guarantees of robustness to potential line failures. These problems are gaining importance in a world where a shift towards renewable energy can impose a significant geographical reallocation of generation capacities, resulting in the necessity of expanding current power transmission grids.

In the TEP, the objective is to find a subset of candidate expansion measures to be installed in a transmission network so as to increase its capability to satisfy the predicted future demand while minimizing both the installment costs and the operational costs.

Acknowledgements

First of all, I thank my supervisors Prof. Daniele Vigo and Prof. Michele Monaci, who supported me during all the three years of my Ph.D and has always been willing to devote their time to clarify any kind of doubt. Thanks also to all the other professors of the Operations Research group of the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi" (DEI) of the University of Bologna: Prof. Andrea Lodi, Prof. Enrico Malaguti, Prof. Valentina Cacchiani, Prof. Silvano Martello, Prof. Paolo Toth, and thanks to all my colleagues: Paolo Paronuzzi, Silvia Anna Cordieri, Alan Osorio Mora, Francesco Cavaliere, Federico Michelotto and Henri Bertrand Roger Jean-Marc Arthur Lefebvre.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Operations Research

Operations research (OR) is a discipline that uses mathematical models, statistical analysis, and computer algorithms to improve decision-making in complex systems. It involves the application of advanced analytical methods and techniques to solve problems in areas such as optimization, simulation, network analysis, and decision analysis.

The objective of operations research is to identify the best possible solutions to complex problems by considering all possible alternatives and evaluating them based on various criteria, such as efficiency, effectiveness, cost, and risk. OR is used in a wide range of industries, including transportation, logistics, manufacturing, healthcare, finance, and government.

One of the principal technique used in operations research is the field of Mathematical optimization (also also known as mathematical programming) i.e, the field of mathematics that deals with finding the best possible solution to a problem within a set of constraints. The goal of mathematical optimization is to maximize or minimize an objective function while satisfying a set of constraints that describe the problem.

Optimization problems can be classified based on the characteristics of the variables and the constraints involved in the problem formulation: for example, problems with linear constraints and continuous variables are classified as Linear Programming problems where problem with continuous variables and convex constraints are classified as Convex Programming and ones dealing with countable object are classified as Combinatorial Optimization. While linear and convex optimization problems are "easy" to solve, presenting polynomial time, combinatorial ones can be particularly hard to solve, especially when instances coming from real-world application are considered. Indeed, many well known and studied combinatorial problems are strongly NP-hard.

Optimization problems can be classified based on the characteristics of the variables and the constraints involved in the problem formulation: for example, problems with linear constraints and continuous variables are classified as Linear Programming problems where problem with continuous variables and convex constraints are classified as Convex Programming and ones dealing with countable object are classified as Combinatorial Optimization. While linear and convex optimization problems are "easy" to solve, presenting general algorithms that can solve to optimality instances of these problems within a running time that is bounded by a polynomial of the size of the input, combinatorial ones can be particularly hard to solve, especially when instances coming from real-world application are considered. Indeed, many well known and studied combinatorial problems are strongly NP-hard. For each optimization problem, the techniques used to find a solution can be divided in three main categories: exact algorithms, approximation algorithms and heuristics. Exact methods are algorithms that can find the exact optimal solution to a problem and guarantee that the solution found is the best possible solution given the constraints and the objective function. Examples of exact methods include the simplex algorithm for linear programming and interior point methods for both linear and convex programming, dynamic programming and branch-and-bound for integer programming. Some exact algorithms are quite general and can be applied to a wide set of problems sharing the same characteristics like for example the branch-and-bound algorithm can solve any problem (given enough time) that can be expressed as a set of linear constraints involving integer and continuous variables. There are also more specialized algorithms for a specific class of problem like for examples Dijkstra's algorithm for the shortest path problem or the Hungarian algorithm for the assignment problem. These algorithms have a lower computational complexity than the generic ones.

On the other hand, heuristics are a class of algorithms designed to resolve a very specific problem in a fast way, without giving any formal guarantee on the quality of the solution. Given the complexity of some problems and the need of solving big instances of practical interest, sometimes heuristics are the only viable methods. Approximation algorithms sit in a middle ground between exact and heuristic algorithms as they are usually faster than exact methods and can be used to find solutions that are guaranteed to be within a certain factor or percentage of the optimal solution In this thesis I will present both exact and heuristic solution strategies for a set of problems arising from real-world applications.

1.2 Bin Packing Problems

The Bin Packing Problem (BPP) is one of the most studied combinatorial problems due to both its rather simple description and its vast practical applications.

The classical bin packing problem in one dimension asks to pack a set of n items each with a weight w_j in the minimum number of identical bins with capacity c so that the sum of the wights of the items inserted in a bin does not exceed the bin capacity.

Many variation and generalization of the BPP where studied through the years. one of the first appearing in literature is the cutting stock problem where instead of n distinguished items we are given n item *types* each one with a weight w_j and a demand d_j of copies of type j to pack/cut. The bin packing problem can be viewed as a specialization of the cutting stock problem where all the demand d_j equals to 1.

Another class natural extensions are the ones to higher dimensions, where both the items and the bins have more than one dimension and the problem ask to pack all the items in the minimun number of bins with no overlap between items. Although is possible to generalize the problem to arbitrary dimensions, for pratical reason the most studied variants are the two-diemnsional bin packing (2D-BPP) and the three-diemnsional bin packing (3D-BPP).

Two-dimensional packing problems appear for the first time in P. Gilmore and R. Gomory, 1965 where the authors presented a column generation approach that generalize the method used in P. C. Gilmore and R. E. Gomory, 1961 by the same authors for the one-dimensional case. In their paper, Gilmore and Gomory, for the pricing problem use a more tractable case where the items have to be packed in row forming layers.

A coordinate approach to the formulation of the two-dimensional variant was considered Beasley, 1985 for a problem where there is a single rectangular bin and to each item is associated a profit and the objective is to maximize the profit of the item packed.

Those two approaches, the layer-based one and the coordinate-based one, are the base for most of the formulations and heuristics for this class of problem.

Another interesting approach is the one proposed in Fekete and Schepers, 2003, Fekete, Schepers, and Veen, 2006 the feasible packings are represented with a graph-theoretical characterization.

The three-dimensional version (3D-BPP) is a generalization of the classical problem where each items is characterized by three dimensions (d_j, w_j, h_j) and must be packed in the minimum number of three-dimensional bins of size (D, W, H) so that each item is inscribed in a bin and there is no overlap with others items. Additional constraints may arise in real-world applications, for example, in road transportation, it is important that each item has enough support from the items beneath so to guarantee the stability of the cargo and avoid potential damage of the goods.

Although the BPP problem and the BPP-3D, being a BPP generalization, are NP-Hard problems, different exact methods are present in the literature. For example, in (Martello, Pisinger, and Vigo, 1998) the authors presented a two level branch-and-bound algorithm that use the and extension to the concept of corner points to the three-dimensional case and a new proposed lower bound L_2. Another exact algorithms is proposed in (Fekete, Schepers, and Veen, 2006), here, the authors presented a two level tree search based on a characterization of feasible solution as interval graph by projecting the items dimensions to the "walls" of the bin and the use of fast heuristics for dismissing infeasible solutions.

However, exact solver, especially for the three-dimensional variant, are impractical for the size of the instances of real-world problems, even more so if additional constraints are included in the model. Thus, for the BPP-3D there is also a rich literature of heuristic methods: (Faroe, Pisinger, and Zachariasen, 2003) presented an heuristic based on the Guided Local Search (GLS). First, a initial solution is build with a greedy approach, then the algorithm use the GLS procedure to reduce iteratively the number of bins by moving the items in the last bin to other bins and than minimize an objective function given by the sum of the overlaps between pair of items.

In (Lodi, Martello, and Vigo, 2002) the authors proposed a tabu search algorithm that solves a three-dimensional by solving first a two-dimensional packing problem and than a one-dimensional packing problem. The algorithm packs items in layers where the top of a layer is the base for the next one. The heuristic tries to both produce a good vertical filling of the space by inserting items with similar height in the same layer and a good horizontal space occupation by producing good solution for the two-dimensional problem for each layer.

In (Crainic, Perboli, and Tadei, 2009) the authors presented a two phase tabu search algorithm where a first phase assign the items to the bins and the second one use the interval graph representation to optimize the actual accommodation of the items in the bins.

In Chapter 4, we propose an algorithm based on the biased random-key genetic algorithm framework (BRKGA) presented in (Gonçalves and Resende, 2013) and we consider some extensions for considering some additional constraints, in particular balancing and stability constraints.

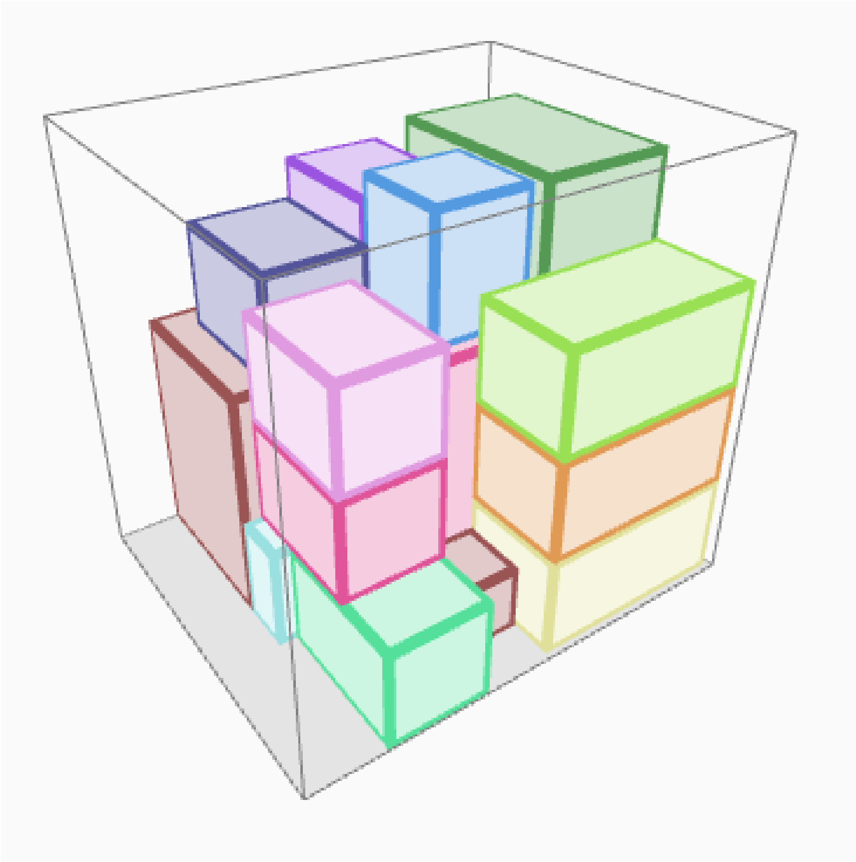

FIGURE 1.1: Example of a BPP3D solution

Our algorithm uses the BRKGA to evolve the order in which the boxes are inserted into the bin while a constructive heuristic based on the maximal space representation for the free spaces within the bin is used for decide the position of each item.

1.3 Transmission Expansion Problem

The Transmission Expansion Planning (TEP) aims at identifying cost-efficient expansion and congestion management measures to ensure the system security and reliability of future electrical transmission grids.

From a modelling point of view, the problem consist of a graph where every node is characterized by a generation capacity and a power demand and arcs represent the transmission lines that connect the center of power production/consumption. A set of decision variables control the installment of measures to expand the capabilities of the infrastructure while the operation of the network follow from the Kirchhoff's law.

The nonlinear, nonconvex nature of the problem make the TEP an challenging problem. Since it first appearance, different optimization techniques have been presented in order to solve the TEP problem, both exact and heuristic. In (Mahdavi et al., 2019) the authors provide a complete classification of the models and solution methods proposed in literature.

in Chapter 4 we propose a expanded model based on (Franken et al., 2019) where different expansion and reinforcement measures are taken into account and a solution method based on Benders decomposition.

Figure 1.2 show a solution which combines all the expansion measures determined expansion measures, determined by the TEP to be part of the cost optimized solution.

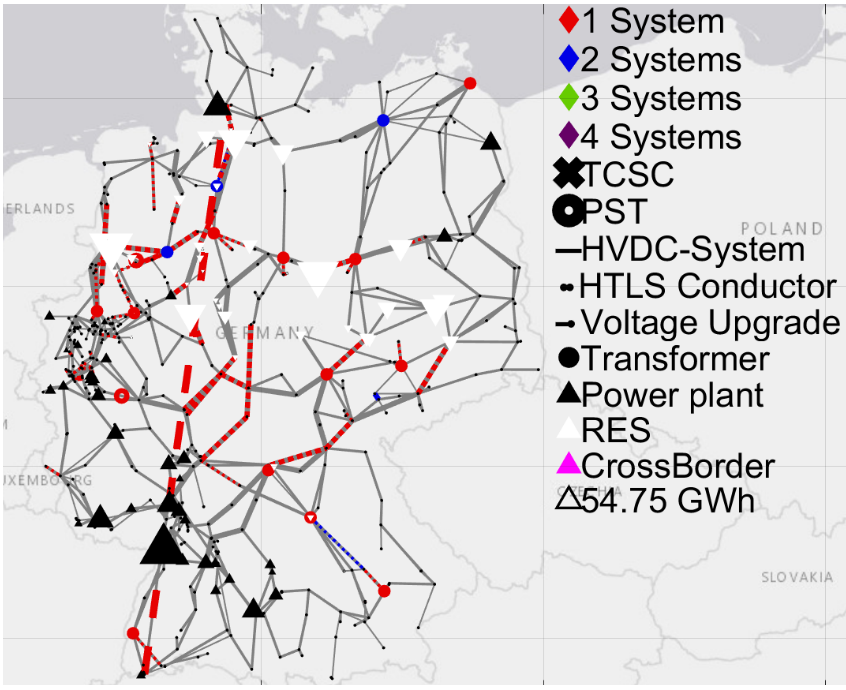

FIGURE 1.2: Sample Expansion and operation measures (source: D6.2 PlaMES EU Project)

Chapter 2

Two dimensional strip bin packing for HPC clustering problems

2.1 Introduction

A High-Performance Computing (HPC) system is a specialized computing environment designed to perform large-scale computations at very high speeds, using a large number of interconnected processors and a massive amount of memory and storage.

There are two distinct utilization patterns observed in high-performance computing systems: multi user varying workload and repetitive throughput oriented. The former is commonly seen in research compute facilities where a diverse set of users submit different types of jobs in a highly variable sequence, necessitating online scheduling with limited scheduling system flexibility. The latter is typical of "production" HPC sites such as the European Centre for Medium-Range Weather Forecasts (ECMWF), where the system executes the same set of jobs at predetermined times of the day in a repetitive manner for long periods, with minor variations in job properties resulting from software changes that can gradually alter the execution properties of some jobs over time.

In this work we are focusing on the repetitive-throughput-oriented. We cast the problem of scheduling m tasks, each of which must be repeated a given number of times, on a cluster with n cores as a two dimensional strip bin packing problem with deformable items. We propose two MILP models adapted from the literature and a mat-heuristic obtained by combining the two models.

2.2 Problem Description

The problem ask to schedule *m tasks* on W computer resources (for example, processors) with the objective of minimizing the total time required to complete the whole batch of tasks (makespan).

To each task j is associated a set of possible *configurations* I_j that represent the way a task can be executed: a configuration is characterized by the number of computer resources assigned to the task $w_i, w_i \leq W$ and the resulting executing time h_i. Also, every task has to be repeated R_j times.

We define $n = \sum_{j=1}^{m} |I_j|$ as the number of all available configurations and R_i as the upper bound on the number of configurations of type i that can be packed, that correspond to the number of repeat of the task j, that is: $R_i = R_j \mid i \in I_j, (i = 1, \ldots, n)$.

2.3 Models

The problem can be modelled as a two-dimensional strip packing problem (2SP) with deformable items.

We propose two models, both adapted from models presented in the literature of 2SP and extended to handle repetitions and the possibility of different configurations for each task. The first model is a layer-based approach presented in Lodi and Monaci, 2003. The second model is a coordinate-based approach presented in Chen, Lee, and Shen, 1995.

2.3.1 Shelf based model

A typical heuristic approach for two-dimensional cutting or packing problems is to restrict the cutting or packing of items only to horizontal slice of the strip/bin called *shelves* with width W and height given by the tallest item in the shelf. The first shelf has as basis the bottom of the strip while all the subsequent shelves have as basis the horizontal line that coincide with the top of the tallest item in the precedent shelf.

As noted in Lodi, Martello, and Vigo, 2004, The following observation holds: For any optimal solution of the problem where items are packed/cut from shelves, there is a equivalent solution so that:

- in each shelf, the tallest item is the one on the left;
- the bottom shelf is the tallest one.

The left-most item in each shelf is said to *initialize* it.

These considerations allow to take into account only the solutions that satisfy these conditions. For these reasons, we can considers the configurations ordered in such a way that $h_1 \geq h_2 \geq \cdots \geq h_n$.

The model assume that there are R_i possible shelves for each configuration i that may be initialized. The initialization of a shelf is described by the following binary variables, notice that if shelf (i, r) is used, must be initialized by configuration i.

$$y_{ir} = \begin{cases} 1 & \text{if shelf } (i,r) \text{ is used,} \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \ldots, n;\ r = 1, \ldots, R_i) \tag{2.1}$$

The number of configurations of type k packed in a shelf is described by the following integer variables:

$$x_{kir} = \begin{cases} \text{number of configurations } k \text{ in shelf } (i,r), & \text{if } i \neq k \\ \text{additional number of configurations } k \text{ in shelf } (i,r) & \text{if } i = k \end{cases} \tag{2.2}$$

$$(k = 1, \ldots, n;\ i = 1, \ldots, n;\ r = 1, \ldots, R_i) \tag{2.3}$$

The model is than as follows:

$$
\begin{aligned}
&\min \sum_{i=1}^{n} \sum_{r=1}^{R_i} h_i y_{ir} && && (2.4)\\
&\text{s.t} \sum_{i \in I_j} \sum_{r=1}^{R_i} y_{ir} + \sum_{k \in I_j} \sum_{i=1}^{k} \sum_{r=1}^{R_i} x_{kir} = R_j && (j = 1, \dots, m) && (2.5)\\
&\sum_{k=i}^{n} w_k x_{kir} \le (W - w_i) y_{ir} && (i = 1, \dots, n;\ r = 1, \dots, R_i) && (2.6)\\
&y_{ir} \in \{0, 1\} && (i = 1, \dots, n;\ r = 1, \dots, R_i) && (2.7)\\
&x_{kir} \in \mathbb{N} && (k = 1, \dots, n;\ i = 1, \dots, n;\ r = 1, \dots, R_i) && (2.8)
\end{aligned}
$$

The objective (2.4) minimizes the sum of the heights of the used shelves. Constraint (2.5) imposes that for each task, the sum of the used shelves associated with a task and the number of configuration associated with the task inserted in other shelves must be equal to the repeat number R_j. Lastly, constraint (2.6) imposes the knapsack constraint on the width of the used shelves.

2.3.2 coordinate based model

This model follow the modelling approach presented for the first time in Beasley, 1985. The space of the strip is seen as a two-dimensional integer lattice where configurations are inserted by putting their bottom-left corner in one of the integer coordinates.

The set of available coordinate for the insertion of the i-th configuration are $W_i \times H_i$ where $W_i = 1, \dots, W - w_i$ and $H_i = 1, \dots, H - h_i$.

We introduce the variables set

$$
x^i_{pq} = \begin{cases} 1 & \text{if configuration } i \text{ bottom-left corner is in } (p, q), \\ 0 & \text{otherwise} \end{cases} \tag{2.9}
$$

$$
(j = 1, \dots, m;\ i \in I_j;\ p \in W_i;\ q \in H_i) \tag{2.10}
$$

$$
\begin{aligned}
&\min \quad z && && (2.11)\\
&\text{s.t} \sum_{i \in I_j} \sum_{p \in W_i} \sum_{q \in H_i} x^i_{pq} = R_j && (j = 1, \dots, m) && (2.12)\\
&\sum_{j=1}^{m} \sum_{i \in I_j} \sum_{\substack{p = r - w_i + 1 \\ p \in W_i}}^{r} \sum_{\substack{q = s - h_i + 1 \\ q \in H_i}}^{s} x^i_{pq} \le 1 && (r = 0, \dots, W - 1;\ s = 0, \dots, H - 1) && (2.13)\\
&(q + h_i) x^i_{pq} \le z && (j = 1, \dots, m;\ i \in I_j;\ p \in W_i;\ q \in H_i) && (2.14)\\
&x^i_{pq} \in \{0, 1\} && (j = 1, \dots, m;\ i \in I_j;\ p \in W_i;\ q \in H_i) && (2.15)\\
&z \in \mathbb{R}^+ && && (2.16)
\end{aligned}
$$

Constraints (2.12) Says that the number of scheduled configurations for each task must be equal to the number of repeats for the task (2.13) model the non-overlap of the configurations packed: each integer point in the lattice, can be covered at most by one configuration, and constraints (2.14) set the value of the objective function, that is, the maximum height of the top-right corner of the packed configurations.

Note that this model requires an estimation of the maximum height of the strip H, necessary to define the number of x variables, i.e, the possible coordinate for inserting the left-bottom corner of the configurations.

2.4 Matheuristic

In this section, we discuss a matherustic that combines the previously discussed models in an efficient way. We first solve the shelf based model with a short time limit, then we create a new instance from each pair of shelves containing the tasks that were packed in the shelves with the relative number of repeats and solve these new instances with the coordinate based model. If the solution produced by the coordinate base model is lower than the sum of the heights of the two shelves, the overall solution is updated accordingly. If the shelf base model does not produce a feasible solution, we use the fast first fit heuristic to produce the shelves. If the number of shelves is odd, we add a dummy shelf with zero tasks packed and a zero height.

Algorithm 1 Matheuristic

procedure MATHEURISTIC(instance)
    let $obj = 0$ be the current value of the ottimal solution.
    solve the instance with the shelf model and a timelimit of 10 seconds
    for pairs i, j of shelves in the solution **do**
        $ub = h_i + h_j$
        create a new coordinate model instance with estimated height $H = ub$ and with the tasks
        that where packed in shelves i and j with their repeats and all their configurations.
        solve the new instance with the coordinate base model model
        let $obj_{i,j}$ be the optimal solution of the coordinate based model.
        if $obj_{i,j} < ub$ **then** $obj = obj + obj_{i,j}$
        else $obj = obj + ub$
        end if
    end for
end procedure

2.5 Instances

Instances with deformable items were generated from reference base instances already present in the literature to test classical packing problems. We use the instances originally proposed for bidimensional knapsack problem Beasley, 1985, named NGCUT. This is a set of 12 instances that have a number of tasks between 7 and 22 with a number of resources between 10 and 30. They give a height that we use as the deadline and for each task j ,a width w_j and a height h_j that we keep as an area $e_j = w_h h_j$.

Starting from the NGCUT dataset we set a maximum for the number of repeats RMAX and a maximum for the number of resources assignable to a task as WMAX. The number of repeats of each task is generated with uniform distribution in the range $[1, RMAX]$. and for each task we generate at most WMAX configurations defined by a width of w_i with $w_i = 1, \dots w_i = WMAX$ and the correspondent height

h_i is given by e_j/w_j rounded up to the nearest integer. Rounding up can lead to dominated configurations that have a different number of assigned resources, but same height. These configurations are removed from the possible choices for a task.

4.1 Table shows the characteristics of the generated instances. Column name report the name of the original instance in the NGCUT dataset. W is the width of the strip, wmax and rmax are respectively the maximum number of assignable resources and the maximum number of repeats. Nitems is the total number of configurations and reff is the sum of repeats, that is, the effective number of configurations that must be scheduled while ub and lb report the upper and lower bound of the objective function for instance.

The upper bound is computed with a fast first fit heuristic based on the shelves formulation where for each task and repeat a random configuration is picked. The lower bound is obtained by dividing the sum of the area of the tasks by the width of the strip.

name	W	ntasks	rmax	wmax	nitems	reff	ub	lb
NGCUT01	10	10	05	05	48	23	51	43
				08	62	31	75	60
				10	68	29	59	56
			10	05	48	49	97	94
				08	62	39	75	73
				10	68	66	136	128
NGCUT02	10	17	05	05	77	49	89	81
				08	98	59	103	95
				10	104	53	86	79
			10	05	77	86	154	142
				08	98	118	191	180
				10	104	76	140	126
NGCUT03	10	21	05	05	90	66	97	85
				08	111	62	97	80
				10	116	56	91	75
			10	05	90	99	153	139
				08	111	89	141	125
				10	116	120	177	160
NGCUT04	10	7	05	05	34	14	34	31
				08	46	20	54	48
				10	49	21	57	50
			10	05	34	45	109	102
				08	46	39	105	95
				10	49	42	108	93
NGCUT05	10	14	05	05	67	42	112	100
				08	91	44	134	116
				10	99	42	119	105
			10	05	67	58	145	137
				08	91	63	184	162
				10	99	58	176	158
NGCUT06	10	15	05	05	70	38	72	66

Continued on next page

name	W	ntasks	rmax	wmax	nitems	reff	ub	lb
				08	92	45	100	86
				10	98	43	89	79
			10	05	70	71	145	136
				08	92	74	158	139
				10	98	87	178	170
NGCUT07	20	8	05	05	33	17	25	23
				08	42	22	35	31
				10	48	27	32	30
			10	05	33	57	63	58
				08	42	46	63	57
				10	48	41	44	41
NGCUT08	20	13	05	05	65	31	83	76
				08	96	37	95	88
				10	109	41	106	95
			10	05	65	71	176	163
				08	96	64	157	142
				10	109	65	171	158
NGCUT09	20	18	05	05	86	49	139	128
				08	127	60	165	153
				10	150	63	189	179
			10	05	86	87	258	243
				08	127	76	246	235
				10	150	99	320	304
NGCUT10	30	13	05	05	65	38	202	183
				08	101	40	186	155
				10	124	37	184	174
			10	05	65	56	294	269
				08	101	69	359	341
				10	124	60	221	210
NGCUT11	30	15	05	05	75	41	173	156
				08	117	42	166	145
				10	142	43	160	148
			10	05	75	92	311	294
				08	117	87	280	257
				10	142	75	265	256
NGCUT12	30	22	05	05	107	59	245	222
				08	161	77	320	299
				10	193	65	238	226
			10	05	107	113	421	403
				08	161	116	452	427
				10	193	91	343	329

TABLE 2.1: instances

As the table 2.2 shows, the shelf based model is able to solve almost all the instances to optimality while the coordinate based model is not able to find a provable optimal solution in the time given. Note that this model tends to produce a bad lower bound that is hard to close. Our algorithm was able to produce solutions as good or better than the ones of the shelf based model with lower running times.

name	rmax	wmax	shelves-init obj	shelves-init bound	shelves-init gap	shelves-init runtime	coord-init obj	coord-init bound	coord-init gap	coord-init runtime	matheuristic obj	matheuristic bound	matheuristic gap	matheuristic runtime
NGCUT01	05	05	44	44	0	0.11	43	43	0.00	46.50	44	-	-	0.23
		08	60	60	0	0.06	60	60	0.00	237.14	60	-	-	1.06
		10	57	57	0	0.08	56	56	0.00	141.30	56	-	-	29.99
	10	05	94	94	0	0.05	94	94	0.00	1098.47	94	-	-	0.90
		08	73	73	0	0.11	73	73	0.00	91.47	73	-	-	3.32
		10	128	128	0	0.06	128	128	0.00	1819.07	128	-	-	4.30
NGCUT02	05	05	82	82	0	0.26	81	81	0.00	1009.83	82	-	-	1.26
		08	95	95	0	0.08	95	95	0.00	1209.65	95	-	-	1.53
		10	80	80	0	0.49	79	79	0.00	838.89	80	-	-	1.12
	10	05	143	143	0	384.91	142	121	14.79	TL	143	-	-	12.32
		08	180	180	0	3.65	190	11	94.21	TL	180	-	-	9.99
		10	127	127	0	0.65	126	126	0.00	3491.61	127	-	-	3.41
NGCUT03	05	05	85	85	0	3.06	85	85	0.00	727.54	85	-	-	1.75
		08	80	80	0	0.09	80	80	0.00	891.54	80	-	-	0.70
		10	75	75	0	0.23	75	75	0.00	736.67	75	-	-	0.87
	10	05	139	139	0	0.17	139	116	16.55	TL	139	-	-	4.44
		08	125	125	0	0.26	126	78	38.10	TL	125	-	-	1.68
		10	160	160	0	0.27	171	11	93.57	TL	159	-	-	2.57
NGCUT04	05	05	31	31	0	0.03	31	31	0.00	3.22	31	-	-	0.27
		08	48	48	0	0.03	48	48	0.00	78.61	48	-	-	0.56
		10	50	50	0	0.02	50	50	0.00	65.37	50	-	-	0.47
	10	05	103	103	0	0.11	102	102	0.00	759.09	103	-	-	3.24
		08	96	96	0	0.15	95	95	0.00	1100.49	96	-	-	0.68
		10	93	93	0	0.05	93	93	0.00	764.85	93	-	-	1.49

Continued on next page

name	rmax	wmax	shelves-init obj	shelves-init bound	shelves-init gap	shelves-init runtime	coord-init obj	coord-init bound	coord-init gap	coord-init runtime	matheuristic obj	matheuristic bound	matheuristic gap	matheuristic runtime
NGCUT05	05	05	101	101	0	0.22	100	100	0.00	1066.89	101	-	-	1.49
		08	117	117	0	0.27	116	116	0.00	2964.34	117	-	-	4.29
		10	105	105	0	0.23	105	105	0.00	1615.33	105	-	-	1.54
	10	05	137	137	0	2.52	137	137	0.00	2076.21	137	-	-	1.94
		08	162	162	0	0.21	168	17	89.88	TL	162	-	-	4.91
		10	158	158	0	0.35	166	9	94.58	TL	158	-	-	3.05
NGCUT06	05	05	67	67	0	0.06	66	66	0.00	531.87	67	-	-	0.65
		08	86	86	0	0.12	86	86	0.00	1316.41	86	-	-	1.05
		10	80	80	0	0.20	79	79	0.00	635.82	80	-	-	0.59
	10	05	137	137	0	0.18	136	136	0.00	2753.32	137	-	-	1.15
		08	139	139	0	0.25	142	81	42.96	TL	139	-	-	2.40
		10	170	170	0	0.19	177	10	94.35	TL	170	-	-	2.69
NGCUT07	05	05	24	24	0	0.01	23	23	0.00	5.17	24	-	-	0.16
		08	31	31	0	0.04	31	31	0.00	75.10	31	-	-	0.88
		10	30	30	0	0.03	30	30	0.00	89.60	30	-	-	0.88
	10	05	59	59	0	0.13	58	58	0.00	762.99	59	-	-	2.79
		08	58	58	0	0.14	57	57	0.00	728.19	58	-	-	1.86
		10	41	41	0	0.04	41	41	0.00	301.83	41	-	-	0.88
NGCUT08	05	05	77	77	0	0.70	76	76	0.00	1444.89	77	-	-	2.35
		08	89	89	0	0.45	90	15	83.33	TL	89	-	-	8.74
		10	95	95	0	2.32	99	11	88.89	TL	95	-	-	22.93
	10	05	165	164	0.61	TL	175	21	88.00	TL	164	-	-	22.14
		08	142	142	0	1.74	156	14	91.03	TL	141	-	-	13.69
		10	159	158	0.63	TL	170	8	95.29	TL	158	-	-	26.41

Continued on next page

			shelves-init				coord-init				matheuristic			
name	rmax	wmax	obj	bound	gap	runtime	obj	bound	gap	runtime	obj	bound	gap	runtime
NGCUT09	05	05	129	129	0	1.65	135	18	86.67	TL	129	-	-	9.37
		08	154	154	0	2.36	164	13	92.07	TL	154	-	-	13.93
		10	180	180	0	2.77	188	8	95.74	TL	180	-	-	80.24
	10	05	244	244	0	10.22	257	6	97.67	TL	244	-	-	21.20
		08	236	236	0	31.47	245	13	94.69	TL	234	-	-	24.76
		10	304	304	0	5.33	318	10	96.86	TL	304	-	-	49.48
NGCUT10	05	05	186	186	0	2443.39	193	49	74.61	TL	184	-	-	889.66
		08	156	156	0	0.85	170	31	81.76	TL	156	-	-	141.67
		10	174	174	0	0.64	184	19	89.67	TL	174	-	-	174.76
	10	05	269	269	0	0.15	294	35	88.10	TL	269	-	-	72.19
		08	342	342	0	2.42	-	-	-	-	341	-	-	2301.01
		10	211	211	0	2.05	-	-	-	-	211	-	-	2753.18
NGCUT11	05	05	160	160	0	44.40	169	40	76.33	TL	159	-	-	35.69
		08	147	147	0	1007.62	162	26	83.95	TL	146	-	-	3620.07
		10	-	-	-	-	158	15	90.51	TL	150	-	-	103.44
	10	05	296	295	0.34	TL	311	23	92.60	TL	296	-	-	65.37
		08	258	258	0	2.26	-	-	-	-	258	-	-	114.09
		10	257	257	0	4.53	-	-	-	-	257	-	-	331.10
NGCUT12	05	05	226	225	0.44	TL	239	42	82.43	TL	226	-	-	337.87
		08	300	300	0	14.85	-	-	-	-	301	-	-	3811.87
		10	227	227	0	5.76	-	-	-	-	227	-	-	713.13
	10	05	406	405	0.25	TL	-	-	-	-	406	-	-	132.14
		08	428	428	0	48.76	-	-	-	-	428	-	-	495.74
		10	331	330	0.30	TL	-	-	-	-	331	-	-	3809.27

Continued on next page

name	rmax	wmax	shelves-init				coord-init				matheuristic			
			obj	bound	gap	runtime	obj	bound	gap	runtime	obj	bound	gap	runtime

TABLE 2.2: results

Chapter 3

Genetic Algorithms for Bin Packing Problem

3.1 Introduction

The bin packing problem consists in inserting a given set of rectangular box (called *items*), in a minimal number of rectangular containers (called *bins*) in such a way that every items is completely inscribed in a bin and there is no overlap between different items. Three-dimensional packing problems arise often in industrial application such as loading cargo into vehicles, container or pallet. In some application, additional constrains are necessary such as stability or cargo balance.

The problem is strongly NP-Hard so finding solution in reasonable time often require the use of some heuristic. In section 3.2 we give a overview of the BRKGA framework and it's specialization for the problem at hand. Finally in 3.3 we provide some computational result.

3.2 Solution approach

The proposed heuristic is based on the biased random-key genetic algorithm framework presented in Gonçalves and Resende, 2013. In this framework, each solution is encoded as a vector of random keys (that is, real numbers generated in the $[0-1]$ interval). Those value are used by ad constructive heuristic (called *decoder*) which build the corresponding packing solution and it's fitness value. At each iteration a population of p solution is constructed and the solutions are then partitioned in two disjoint subset: a small one called *elite* of p_e element and a bigger one of $p - p_e$ non elite elements. The subsequent generation is computed first by copying the p_e elite element, then p_m random value are produced for introduce some mutation in the process with the scope of exit eventual local minima. The remaining $p - p_e - p_m$ solutions are generated picking up a random element of the elite population (with repetition) and a random element of the total population and combine the two solution via *parameterized uniform crossover*: given a parameter chosen by the user $\rho_e \in [0-1]$ each element of the new offspring vector is inherited from the elite parent with probability ρ_e or from the the other one with probability $1 - \rho_e$.

This approach allowed a clear separation of the problem specific part of the heuristic (namely, the decoder and the specification of the encoding of the solution in random-key vectors) from the problem-independent part, namely the evolutionary process.

Another important aspect of the framework it's that it allowed the evaluation of fitness of the solution to be run in parallel, which enhances greatly the efficiency of the approach.

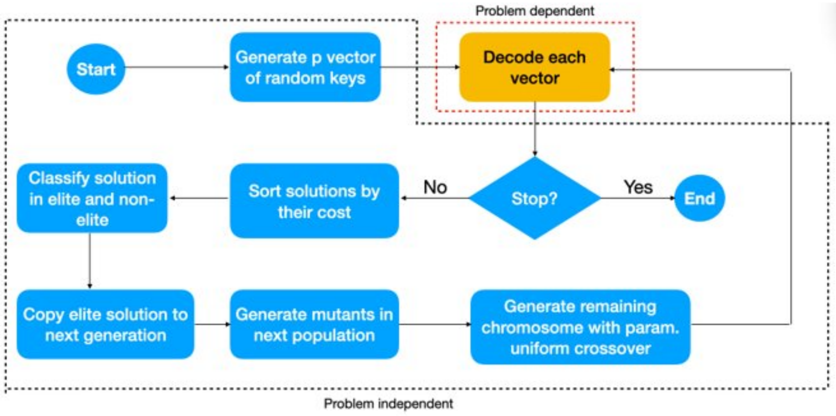

FIGURE 3.1: BRKGA scheme

For the decoder, the *empty-maximal space rapresentation* (EMS) Lai and Chan, 1997 is choosen. The free space in the bin is represented as a set of not disjoint rettagular shapes each represented by the coordinates of the left-bottom-back corner and the upper-right-top corner and not contained in a other space. Every time a item is inserted the list of free space is updated with the *difference process*.

difference process

Given the $i-th$ EMS of coordiante $[(x_i, y_i, z_i), (X_i, Y_i, Z_i)]$ and the $j-th$ item inserted at coordinate $[(x_j, y_j, z_j), (X_j, Y_j, Z_j)]$ and assumed:

$$x_i \leq x_j \leq X_j \leq X_i, \quad y_i \leq y_j \leq Y_j \leq Y_i, \quad z_i \leq z_j \leq Z_j \leq Z_i$$

The new spaces are generated considering the projections along the axis:

$$\begin{aligned}
&[(x_i, y_i, z_i), (X_i, Y_i, Z_i)] - [(x_j, y_j, z_j), (X_j, Y_j, Z_j)] = \\
&[[(x_i, y_i, z_i), (x_j, Y_i, Z_i)], \\
&[(X_j, y_i, z_i), (X_i, Y_i, Z_i)], \\
&[(x_i, y_i, z_i), (X_i, y_j, Z_i)], \\
&[(x_i, Y_j, z_i), (X_i, Y_i, Z_i)], \\
&[(x_i, y_i, z_i, (X_i, Y_i, z_j)], \\
&[(x_i, y_i, Z_j), (X_i, Y_i, Z_i)]]
\end{aligned}$$

This process must be repeated for each EMS that overlap the item.

After that, all the spaces with a null dimension or that are totally inscribed in a other space must filtered out. That is, the space must be removed from the list of EMS if:

$$(x_i \geq x_j) \wedge (y_i \geq y_j) \wedge (z_i \geq z_j) \wedge (X_i \leq X_j) \wedge (Y_i \leq Y_j) \wedge (Z_i \leq Z_j) \text{ dove } i \neq j \in S$$

or:

$$(x_i = X_i) \vee (y_i = Y_i) \vee (z_i = Z_i)$$

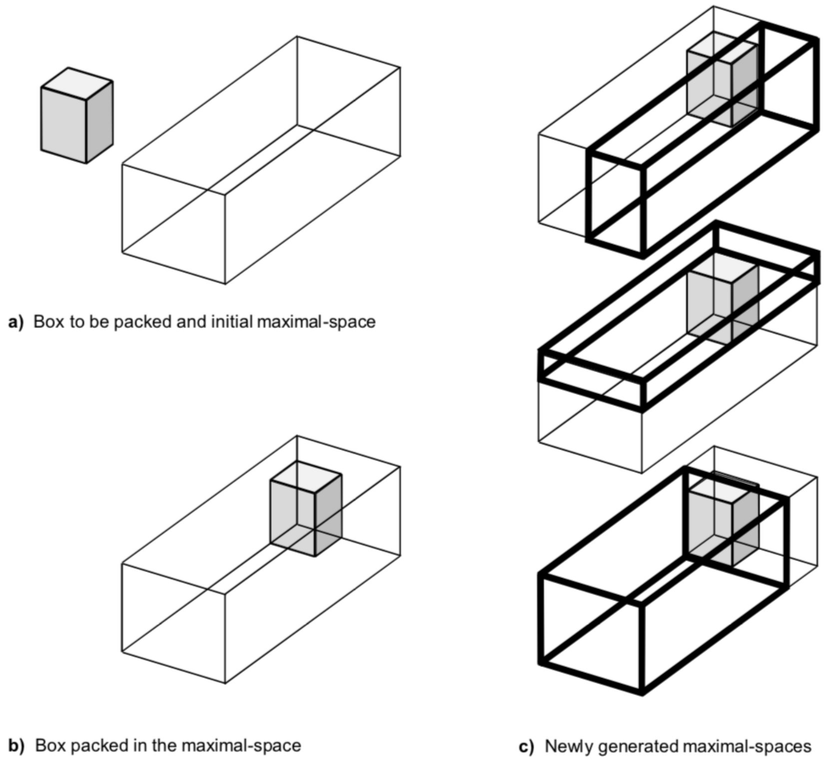

FIGURE 3.2: Example of difference process

TABLE 3.1: params

parameter	description	recommended value
p	population size	$p = an$, where $1 \leq a \in \mathbb{R}$
p_e	elite population size	$0.10p \leq p_e \leq 0.25p$
p_m	mutant population size	$0.10p \leq p_m \leq 0.30p$
ρ_e	probability of inherit from elite parent	$0.5 < \rho_e \leq 0.8$

Figure 3.2 show an example of difference process where an object is inserted at the origin of the EMS.

3.2.1 Parameters Selection

In the BRKGA algorithm there are a set of parameters that the user can set:

- genes number in the chromosome (n);
- population size (p);
- elite population size (p_e);
- mutant population size (p_m);
- probability of inherit from the elite parent (ρ_e);

Although there is not a precise way to select these parameters, in Gonçalves and Resende, 2013 some guidelines are suggested, we report such suggestion in table 3.1.

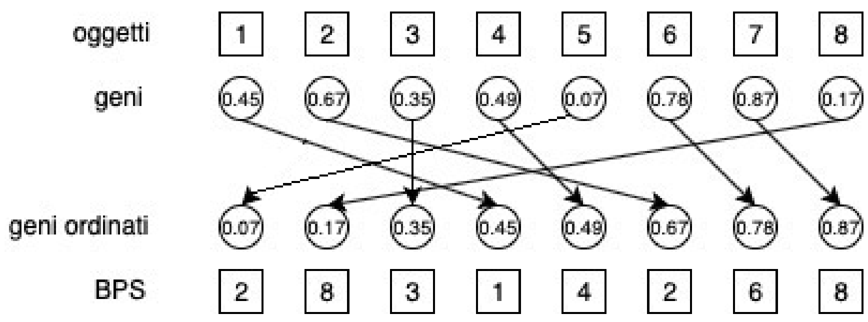

FIGURE 3.3: Insertion order

Decoder

For our problem, each solution is encoded as a vector of n random-keys, where n is the number of items in the instace of the problem. This vector define the insertion order of the items: sorting the random-keys in non-increasing order give us a permutation of the items as show in figure 3.3.

The EMS is selected based on a best-fit search, that is, the smaller one that can contain the item is choosen.

The item rotation is choosen so to minimize the residual space in the fitter dimension. That is, if P is the set of all the permutation of the dimensions of the item and (e_1, e_2, e_3) are the dimension of the selected EMS, the choosen rotation is the one that permute the dimension so that:

$$\min_{\sigma \in P} \min(e_1 - \sigma(x), e_2 - \sigma(y), e_3 - \sigma(z))$$

The pseudocode of the decoder is given in 2.

Algorithm 2 Decoder

procedure DECODER(BPS)
    let B be the set of open bins;
    for $i \leftarrow 0, n$ **do**
        $boxToPack \leftarrow BPS_i$;
        let $selEMS$ The EMS, for every bin in B,
        with minimal volume that can contain $boxToPack$;
        if $selEMS = null$ **then**
            $B \leftarrow B \cup \{newBin\}$;
            $selEMS = newBin$;
        end if

        select the rotation of $boxToPack$;
        that minimize the residual space;

        insert $boxToPack$ in the origin of $selEMS$;
        update the EMS list with difference process;
    end for
    compute fitness value;
end procedure

TABLE 3.2: instance types

Type 1:	$w_j \in [1, \frac{1}{2}W]$,	$h_j \in [\frac{2}{3}H, H]$,	$d_j \in [\frac{2}{3}D, D]$;
Type 2:	$w_j \in [\frac{2}{3}W, \frac{1}{2}W]$,	$h_j \in [1, \frac{1}{2}H]$,	$d_j \in [\frac{2}{3}D, D]$;
Type 3:	$w_j \in [\frac{2}{3}W, \frac{1}{2}W]$,	$h_j \in [\frac{2}{3}H, H]$,	$d_j \in [1, \frac{1}{2}D]$;
Type 4:	$w_j \in [\frac{1}{2}W, W]$,	$h_j \in [\frac{1}{2}H, H]$,	$d_j \in [\frac{1}{2}D, D]$;
Type 5:	$w_j \in [1, \frac{1}{2}W]$,	$h_j \in [1, \frac{1}{2}H]$,	$d_j \in [1, \frac{2}{2}D]$;

3.3 Results

3.3.1 Test instances

The algorithm is banchmarked on a set of 320 problems presented in Martello, Pisinger, and Vigo, 1998.

The instances are grouped into 8 class with 40 instances each, 10 for each value of $n \in \{50, 100, 150, 200\}$. For the classes 1-5, the bins dimensions $W = D = H = 100$ and there are 5 types of items with dimensions (d_j, w_j, h_j) uniformely generated in the intervals shown in table 3.2. For each class k, an item is of type k with probability 60% and one of the others 4 classes with probability 10% each.

The classes 6-8 are defined in the following way:

- class 6: $W = H = D = 10$; $w_j, h_j, d_j \in [1, 10]$;
- class 7: $W = H = D = 40$; $w_j, h_j, d_j \in [1, 35]$;
- class 8: $W = H = D = 100$; $w_j, h_j, d_j \in [1, 100]$;

3.4 Parameter selection

As discussed in 3.2.1, the BRKGA algorithm requires the specifications of some initial parameters. Following the suggestion given in table 3.1, the algorithm was tested on a set of 5 challenging instances using all the possible combinations of the following values for the parameters:

- $p_e \in \{0.10, 0.15, 0.20\}$;
- $p_m \in \{0.10, 0.15, 0.25\}$;
- $\rho_e \in \{0.70, 0.75, 0.80\}$;
- population of $10, 20$ o 30 times the number of the items in the instance;

The configuration that produced the best solutions of the instances tested is $p_e = 0.1$, $p_m = 0.1$, $\rho_e = 0.7$ with a population of 30 times the items count.

TABLE 3.3: Computational Results

Class	Bin size	n	L_2	Goncalves et al. $6r$	NB	aNB	$Time(s)$	$6r$	NB	aNB	$Time(s)$
1	100	50	12.5	11.5	13.4	13.4	2.1	11.5	13.4	13.4	0.46
	100	100	25.1	22.9	26.7	26.6	17.8	22.9	26.6	26.7	12.09
	100	150	34.7	32.0	36.6	36.3	45.2	31.6	36.4	36.8	74.79
	100	200	48.4	43.7	51.0	50.7	69.1	43.0	50.7	51.1	145.67
2	100	50	12.7	11.7	13.9	13.8	3.9	11.7	13.9	13.8	0.52
	100	100	24.1	22.5	25.7	25.5	20.5	22.4	25.8	25.5	10.95
	100	150	35.1	32.2	37.0	36.7	39.2	31.5	37.1	36.6	65.27
	100	200	47.5	42.9	49.6	49.4	91.6	42.2	50.0	49.4	145.29
3	100	50	12.3	11.6	13.3	13.3	4.1	11.5	13.3	13.3	0.56
	100	100	24.7	22.7	26.2	25.9	21.2	22.5	26.4	25.9	9.61
	100	150	36.0	32.4	37.6	37.5	43.6	32.0	37.6	37.5	46.19
	100	200	47.8	43.0	50.1	49.8	78.2	42.4	50.3	49.8	155.23
4	100	50	28.7	28.9	29.4	29.4	5.0	28.9	29.4	29.4	0.038
	100	100	57.6	58.4	59.0	59.0	26.4	58.4	59.0	59.0	0.17
	100	150	85.2	86.4	86.8	86.8	40.7	86.4	86.8	86.8	0.47
	100	200	116.3	118.3	118.8	118.8	60.3	118.3	118.8	118.8	0.98
5	100	50	7.3	7.5	8.3	8.3	6.9	7.5	8.4	8.3	0.26
	100	100	12.9	13.7	15.0	15.0	15.0	13.7	15.1	15.0	8.93
	100	150	17.4	18.5	20.1	19.9	33.7	18.6	20.1	19.9	194.0
	100	200	24.4	25.3	27.1	27.1	67.5	25.3	27.1	27.0	356.1
6	10	50	8.7	8.9	9.8	9.8	5.4	8.9	9.9	9.8	0.35
	10	100	17.5	17.9	19.0	18.8	25.9	17.9	19.0	18.9	4.80
	10	150	26.9	27.6	29.2	29.2	42.3	27.5	29.2	29.2	14.39
	10	200	35.0	35.5	37.2	37.2	75.0	35.5	37.2	37.2	47.01
7	40	50	6.3	6.4	7.4	7.4	6.5	6.4	7.4	7.4	0.99
	40	100	10.9	10.8	12.3	12.2	12.6	10.8	12.4	12.3	18.43
	40	150	13.7	13.7	15.5	15.2	27.2	13.7	15.5	15.3	115.45
	40	200	21.0	21.6	23.4	23.4	72.5	21.6	23.4	23.4	189.37
8	100	50	8.0	8.3	9.2	9.2	11.7	8.3	9.4	9.2	0.27
	100	100	17.5	17.5	18.9	18.9	21.4	17.5	19.0	18.9	14.78
	100	150	21.3	22.0	23.6	23.5	48.2	21.8	23.8	23.7	98.54
	100	200	26.7	27.5	29.4	29.2	64.0	27.3	29.4	29.2	299.39
total bin			9242	9038	9805	9772		8995	9831	9776	

3.5 Computational Results

Table 3.3 Provide the numerical results obtained on the tested instances. The algorithm was implemented in C++ and compiled using clang. The experiments were performed on a Intel processor i5-8259U @2.3 GHz with 8GB of RAM.

In column L_2, we reported the lower bound as defined by Martello, Pisinger, and Vigo, 1998, the column $6r$ is the result with all rotation allowed, NB is the result without rotation and with the fitness function that only count the number of open bins, aNB the result with the aNB fitness function and lastly column $Time$ is the average run time of the three runs.

Fig. 3.4 show how our algorithm manage to produce better solution when the rotation of items is permitted while maintaining competitive result in the other cases.

The figures 3.6 and 3.7 compare, respectively, the results obtained by the two heuristics with rotations and fitness function aNB. The new procecure produces solution that are better or equivalent to the ones obtained by Gonçalves and Resende, 2013.

Finally, table 3.4 ed figure 3.5 show the comparison of the results with other algorithms in the literature.

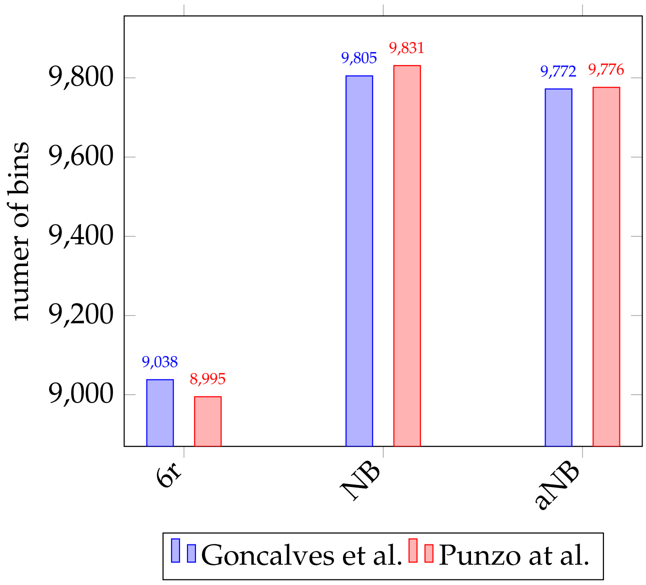

FIGURE 3.4: Heuristics results

The figure shows the total of used bins (excluding the instances for class 2 and class 3, for which there are no solution for the alghoritms TS^{2}PACK e GLS).

The *rot* bar represent the total number of bin with rotations and with fitness function aNB, bars NB e aNB are the results without rotations and with the corresponding fitness function.

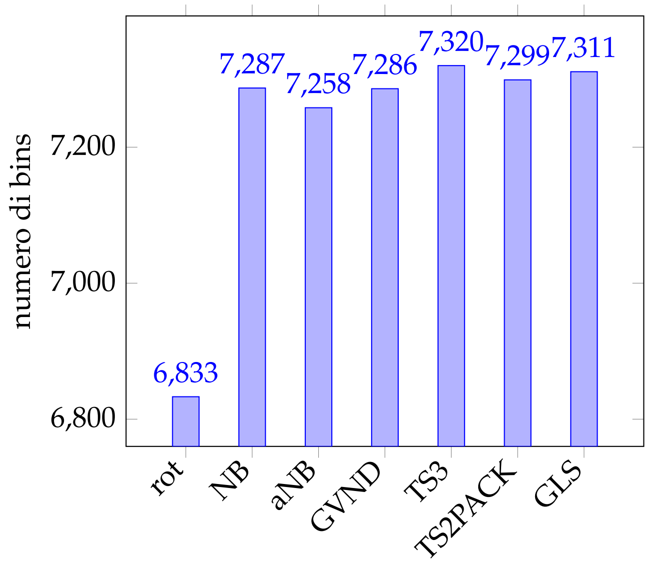

FIGURE 3.5: Comparison with literature's algorithms

TABLE 3.4

Class	Bin size	n	$6r$	NB	aNB	$Time(s)$	$TS3$	$GVND$	TS^2PACK	GLS
1	100	50	11.5	13.4	13.4	0.46	13.4	13.4	13.4	13.4
	100	100	22.9	26.6	26.7	12.09	26.6	26.6	26.7	26.7
	100	150	31.6	36.4	36.8	74.79	36.7	36.4	37.0	37.0
	100	200	43.0	50.7	51.1	145.67	51.2	50.9	51.1	51.2
2	100	50	11.7	13.9	13.8	0.52	13.8	13.8	-	-
	100	100	22.4	25.8	25.5	10.95	25.7	25.7	-	-
	100	150	31.5	37.1	36.6	65.27	37.2	36.9	-	-
	100	200	42.2	50.0	49.4	145.29	50.1	49.4	-	-
3	100	50	11.5	13.3	13.3	0.56	13.3	13.3	-	-
	100	100	22.5	26.4	25.9	9.61	26.0	26.0	-	-
	100	150	32.0	37.6	37.5	46.19	37.7	37.6	-	-
	100	200	42.4	50.3	49.8	155.23	50.5	50.0	-	-
4	100	50	28.9	29.4	29.4	0.038	29.4	29.4	29.4	29.4
	100	100	58.4	59.0	59.0	0.17	59.0	59.0	58.9	59.0
	100	150	86.4	86.8	86.8	0.47	86.8	86.8	86.8	86.8
	100	200	118.3	118.8	118.8	0.98	118.8	118.8	118.8	119.9
5	100	50	7.5	8.4	8.3	0.26	8.4	8.3	8.3	8.3
	100	100	13.7	15.1	15.0	8.93	15.0	15.0	15.2	15.1
	100	150	18.6	20.1	19.9	194.0	20.4	20.1	20.1	20.2
	100	200	25.3	27.1	27.0	356.1	27.6	27.1	27.4	27.2
6	10	50	8.9	8.9	9.8	0.35	9.9	9.8	9.8	9.8
	10	100	17.9	19.0	18.9	4.8	19.1	19.0	19.1	19.1
	10	150	27.5	29.2	29.2	14.39	29.4	29.2	29.2	29.4
	10	200	35.5	37.2	37.2	47.01	37.7	37.4	37.7	37.7
7	40	50	6.4	7.4	7.4	0.99	7.5	7.4	7.4	7.4
	40	100	10.8	12.4	12.3	18.43	12.5	12.5	12.3	12.3
	40	150	13.7	15.5	15.3	115.5	16.1	16.0	15.8	15.8
	40	200	21.6	23.4	23.4	189.4	23.9	23.5	23.5	23.5
8	100	50	8.3	9.4	9.3	0.27	9.3	9.2	9.2	9.2
	100	100	17.5	19.0	18.9	14.78	18.9	18.8	18.9	18.9
	100	150	21.8	23.8	23.7	98.54	24.1	24.1	23.9	23.9
	100	200	27.3	29.4	29.2	299.4	30.3	29.8	30.0	29.9
Total bin			8995	9831	9776		9863	9813		

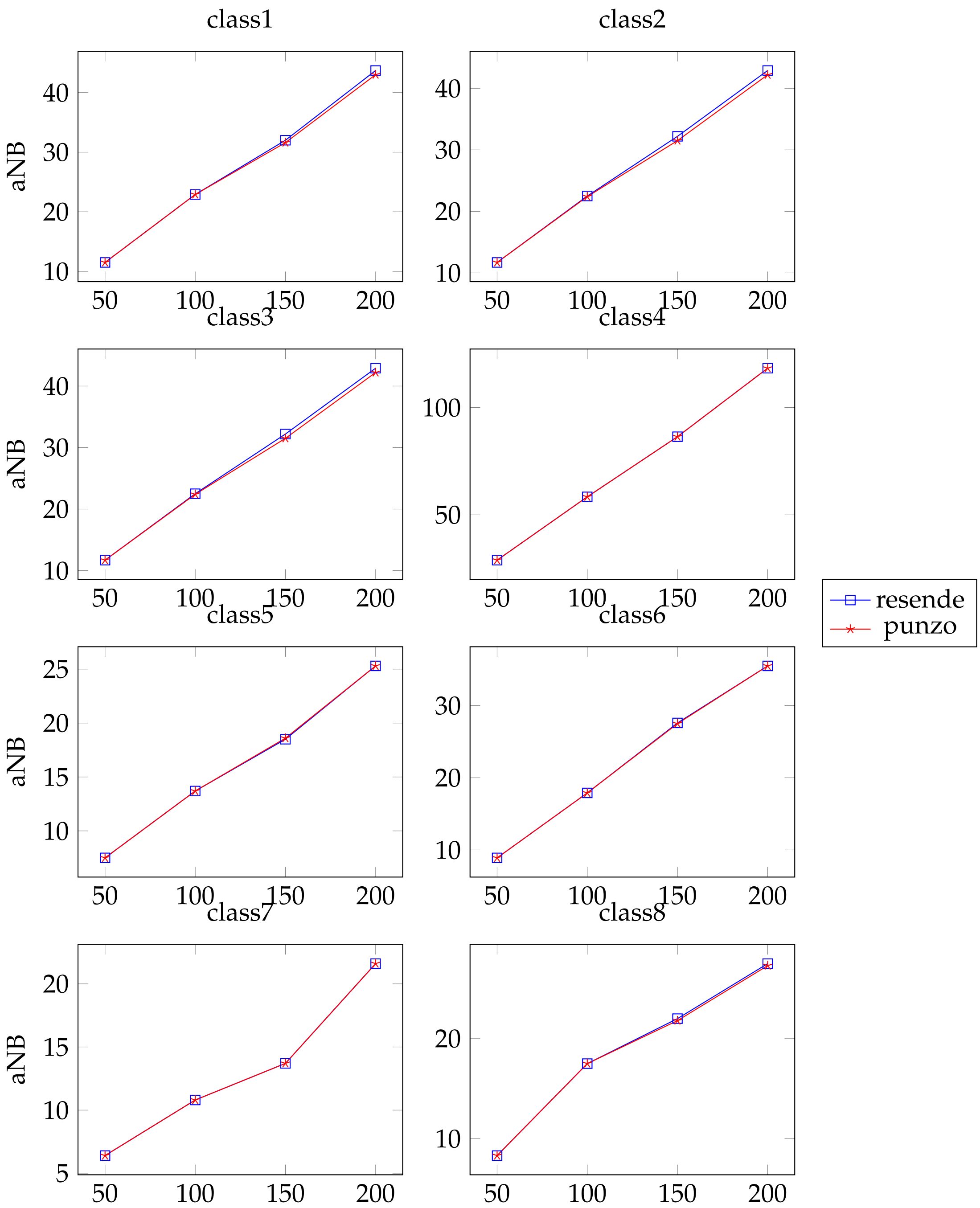

FIGURE 3.6: Comparison quality of solution

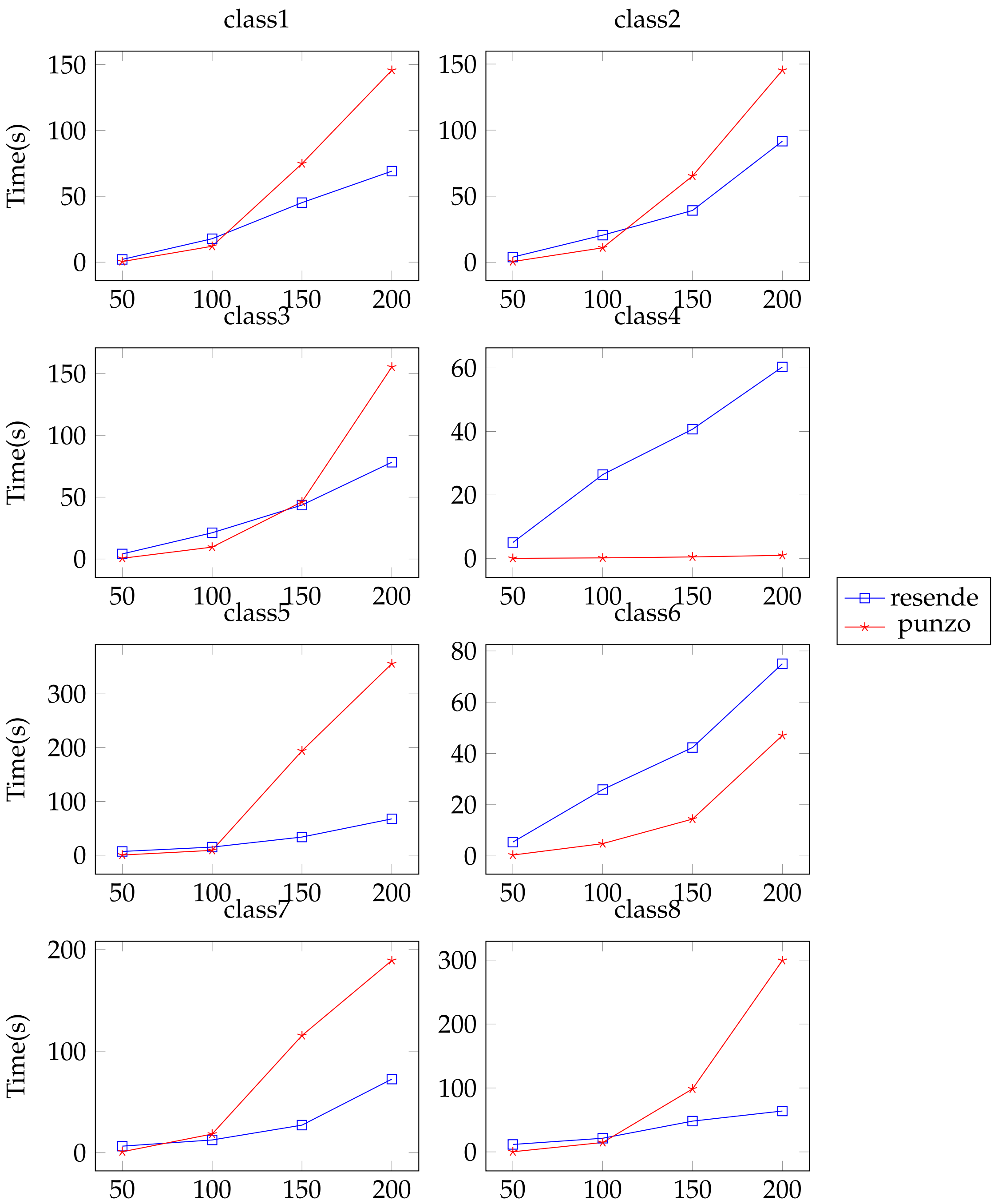

FIGURE 3.7: Comparison runtime

TABLE 3.5: support and balance

Class	Bin size	n	Unsupported	time(s)	Support	time(s)	Load Balance	time(s)
1	100	50	11.5	0.5	12.2	8.5	11.8	2.3
	100	100	22.9	26.0	24.5	147.1	23.8	211.5
	100	150	31.6	116.3	35.0	622.4	33.9	465.0
	100	200	43.0	245.7	48.5	971.9	46.2	1041.7
2	100	50	11.7	1.13	12.7	6.7	11.9	8.8
	100	100	22.4	17.0	23.8	123.9	22.8	217.6
	100	150	31.5	114.1	35.4	559.9	33.9	441.16
	100	200	42.2	266.2	47.6	1132.5	45.4	1174.7
3	100	50	11.5	1.6	13.3	10.1	11.7	4.0
	100	100	22.5	22.9	26.2	115.9	22.6	208.0
	100	150	32.0	110.8	38.2	559.5	33.6	484.3
	100	200	42.4	249.2	49.5	1675.5	44.5	1501.0
4	100	50	28.9	0.0	28.9	0.2	28.9	0
	100	100	58.4	0.2	58.4	1.0	58.5	0
	100	150	86.4	0.5	86.7	66.8	86.4	0
	100	200	118.3	1.1	118.3	15.6	118.3	0
5	100	50	7.5	0.2	8.2	4.1	7.5	1.3
	100	100	13.7	15.1	15.4	120.9	13.7	180.0
	100	150	18.6	178.9	21.6	904.6	18.8	1499.8
	100	200	25.3	275.9	28.9	2061.4	25.5	3541.4
6	100	50	8.9	0.1	10.0	7.8	8.9	1.2
	100	100	17.9	2.5	20.0	101.4	17.9	41.5
	100	150	27.5	20.4	30.8	509.7	27.6	144.2
	100	200	35.5	35.2	40.5	1327.0	35.5	457.4
7	100	50	6.4	0.8	7.1	13.2	6.5	28.3
	100	100	10.8	21.4	13.7	92.8	11.0	100.7
	100	150	13.7	168	18.3	928.6	14.1	698.8
	100	200	21.6	204.5	27.1	1989.3	26.6	1041.5
8	100	50	8.3	0.3	8.8	12.3	8.3	4.2
	100	100	17.5	12.2	18.9	106.7	17.7	66.4
	100	150	21.8	152.2	25.1	854.5	22.3	1070.0
	100	200	27.3	287.0	31.6	2500.0	27.9	1947.0
total			8995		9852		9463	

3.6 Results with additional constraints

Lastly, in table 3.5 are reported the results obtained with support and balance constraints.

The column *Unsupported* show the result without additional constraints, while the column *Support* show the result with the support constraint and the column *load Balance* with both support and balance.

For the support constraint, a items is considered supported if at least 70% of its base in in direct contact with the underlying items or bin floor while the cargo is considered balanced if the mass center is distant from the geometrical center of less than the 10% of the size of the bin.

Differently from the previous tables, this one report only run time for the run with permitted rotations.

Chapter 4

Transmission Expansion Problem

4.1 Introduction

The general objective of the project is the development of an integrated planning tool for multi-energy systems on a European scale. To reach the COP21 goals concerning a stepwise reduction of energy-related greenhouse gas (GHG) emissions in a cost effective way, the decarbonization of multiple energy sectors is necessary. The projected increase in the power load in the near future and the shift towards a increased share of renewable resources, that are usually placed in remote areas, far away from the major centers of energy consumption, require a substantial addition of transmission capacity.

The scope of the Transmission Expansion Planning (TEP) is the minimization of the investment and operational costs necessary to expand the transmission grid in a way that allow to meet the future demand, security and environmental requirements.

The rest of this chapter is organized as follow: in Section 4.2 we formally describe the TEP problem. Section 4.3 gives an overview of the basic assumptions for the DC power flow model and introduces a mathematical formulation of the problem. In 4.4 we describe an exact solution method based on Benders decomposition and a metaheuristic algorithm based on a genetic algorithm. Finally, computational experiments on real instances are given in Section 4.5.

4.2 Problem Description

Given an energy network and a profile for demand and generation for each node along a given time horizon, the objective of the Transmission Expansion Planning (TEP) problem is to determine an optimal extension of the network, i.e, the selection of a set of new lines and other exapnsion measures, so that all demands are satisfied at minimum investment and operational cost.

To obtain a robust solution, the network is analyzed at different instants of time (called *grid snapshots*), each one characterized by a specific load/generation pattern. Also, different *outages situations* are taken into account to ensure the capacity to meet demands in case of failure of some power line.

The starting topology of the transmission grid is defined by a set of nodes, corresponding to electrical busses and by a set of transmission corridors. Nodes are associated with a a power load for each grid snapshot and with a set of generation units, including renewable plants that provide a certain power output at each grid snapshot.

Transmission corridors have multiple position in which parallel circuits may be installed. While some of these positions are already occupied by pre-existing lines, the remaining ones are available for new lines. We consider the possibility to install both AC lines and HVDCs (A high-voltage direct current (HVDC) electric power transmission system).

Some of the existing lines can be upgraded in different ways, namely by installing a Phase Shift Transformer (PST) or a Thyristor Controller Series Compensator (TCSC), or by upgrading the voltage level, or by rewiring the circuits with a conductor having an increased transmission capacity. Some of these expansion operations require the definition of additional parameters, such as, for example, the voltage angle for the PST.

The construction of a new line or the extension of an existing one are modelled by a set of binary variables.

In case of power grids, several representations exist, each one with a different trade off between accuracy and computational complexity. The more accurate model is the AC power flow model.

In this model the active power flow through a lossless transmission line is given by

$$P_L = \frac{|V_N||V_Q|}{X_L} \sin(\delta_N - \delta_Q) \tag{4.1}$$

where V_N and V_Q are the voltage amplitude at node N and node Q, respectively, δ_N and δ_Q are the associated voltage angles, and X_L is the reactance of the line.

Given the non-linearity of this formulation, the model is seldom used in practice. Indeed a grid of N nodes result in a system of $2N$ non-linear equations, i.e., a computationally intractable model in practice. A more practical representation of the system can however be obtained using the so-called DC power flow model, which is obtained by a linearization of the AC model, and is based on three assumptions:

1. line resistances are negligible compared to line reactances, i.e.,:

$$B_L = \frac{-X_L}{R_L^2 + X_L^2} \approx -\frac{1}{X_L}$$

2. the voltage profile is flat, i.e., the amplitude is equal for all nodes (p.u is the unit value):

$$|V_N| \approx 1 \ p.u \tag{4.2}$$

3. the voltage angles differences are small. This assumption allows to approximate the sin of this difference with the difference itself in the following way:

$$\sin(\delta_N - \delta_Q) \approx \delta_N - \delta_Q \tag{4.3}$$

It follows from the previous assumptions that equation (4.1) simplifies to

$$P_L = B_L(\delta_N - \delta_Q), \tag{4.4}$$

where B_L is the susceptance of the line.

The resulting model includes linear constraints only. However, some decision variables are forced to be binary as they model the possibility of selecting the expansions measures, which yields an MILP formulation. This model will be detailed in the next section.

4.3 Mathematical Model

4.3.1 Input sets

The structure of the network is defined by a set Ω_K of node/busses, and a set of Ω_T transmission corridors. Each node comprises a set Ω_{G_k} of generation units, including traditional power plant (Ω_{PP_k}) and renewable (Ω_{RES_k}). Each transmission corridor t is characterized by a set of available voltage levels (Ω_{V_t}) and multiple position (Ω_{N_t}) for the installation of parallel circuits. Some of these positions are occupied by pre-existing circuits ($\Omega_{N_{t,0}}$), whereas the remaining ones are free for installation of some candidate new circuit ($\Omega_{N_{t,c}}$).

Both AC and HVDC lines (set Ω_{DC}) are available as new circuits, and a wide portfolio of expansions measures can be used to upgrade some of the existing circuits. In particular, set $\Omega_{N_{t,0}}$ includes the following subsets:

- $\Omega_{N_{t,0,pst}}$ is the set of circuits in which a Phase Shifting Transformers can be installed in series with the circuit;
- $\Omega_{N_{t,0,tcsc}}$ represents the set of circuits in which a Thyristor Controlled Series Compensator can be installed;
- $\Omega_{N_{t,0,vu}}$ is the set of circuits whose voltage level can be upgraded; and
- $\Omega_{N_{t,0,rew}}$ denotes the set of circuits that can be rewired with a conductor characterised by an increased transmission capacity.

Note that an existing line may belong to more than one of these subsets.

Each line is identified by the triplets (t, n, v) where, $t \in \Omega_T, \quad n \in \Omega_{N_t}, \quad v \in \Omega_{V_t}$

Lastly, the grid is analyzed to different grid snapshots (set Ω_U) and at different outage situations (set Ω_{cs}).

4.3.2 Variables

The construction of new lines or the reinforcement of the pre-existing ones are controlled by the introduction of the following binary variables:

- $y^{AC}_{t,n,v}$: 1 if AC line of voltage level v is installed in corridor t and position n, 0 otherwise; $\quad t \in \Omega_T; n \in \Omega_{N_{t,c}}; v \in \Omega_{V_t}$
- y^{REW}_{t,n,v_0}: 1 if pre-existing line in transmission corridor t and position n is rewired, 0 otherwise; $\quad t \in \Omega_T; n \in \Omega_{N_{t,0,rew}}$
- y^{VU}_{t,n,v_0}: 1 if pre-existing line in transmission corridor t and position n voltage level is upgraded, 0 otherwise;
  $t \in \Omega_T; n \in \Omega_{N_{t,0,vu}}$
- y^{DC}_{t}: 1 if HVDC line in corridor t is installed, 0 otherwise; $\quad t \in \Omega_T$
- y^{PST}_{t,n,v_0}: 1 if a phase shift transformer is installed serially to the pre-existing line in corridor t and position n, 0 otherwise; $t \in \Omega_T; n \in \Omega_{N_{t,0,pst}}$
- y^{TCSC}_{t,n,v_0}: 1 if Thyristor-controlled series capacitor is installed on line in corridor t and position n, 0 otherwise;
  $t \in \Omega_T; n \in \Omega_{N_{t,0,tcsc}}$

As explained in 4.2, the power flowing along line (t,n,v) at grid snapshot u and outage situation cs ($f^{AC,u,cs}_{t,n,v}$) depends to the difference of the voltage phase angles at starting and ending node of the line: ($\theta^{u,cs}_{kf_{t,n,v_0}}$ and $\theta^{u,cs}_{kt_{t,n,v_0}}$, respectively).

These θ values are represented as continuous variables that are bounded by a maximum and minimum value.

From the Kirchhoff first law, nodal power balance equations are derived. Redispatch variables ($\Delta p^{+,u}_{g}$ and $\Delta p^{-,u}_{g}$) are introduced to model the possibility for the transmission system operator for instructing the power plant operators to adjust the active power feed-in, so as to avoid congestions (or to solve them, if they happen). Additional slacks variables ($r^{u,cs}_{g_k}$and$r^{u,cs}_{d_k}$) are introduced for modelling the impossibility to satisfy the demand of the node with the current infrastructure.

Finally, we introduce further continuous variables that are used to model the control parameters of some of the expansions expansion measures, namely:

- $\theta^{PST,u}_{t,n,v_0}$ is the voltage angle of the Phase Shift Transformer that can be installed in series with the circuit in corridor t and position n and grid snapshot u;
- $\theta^{TCSC,u}_{t,n,v_0}$ is the equivalent voltage angle of the Thyristor Controlled Series Compensator that can be installed on the circuit in corridor t and position n and grid snapshot u.

4.3.3 Objective function

The TEP formulation aims at minimising overall costs resulting from the expansion and operation of the electrical transmission grid.

$$\min \quad IC + OC \tag{4.5}$$

The investment costs IC include the costs for the installation of new assets as well as costs related with the operation of these assets. Operational costs for installed assets are modelled as a percentage of corresponding investment costs.

where the investment cost is defined by:

$$IC = (1 + C^{OP}(1 + \frac{1}{\alpha})) \sum_{t \in \Omega_T} IC_t \tag{4.6}$$

C^{OP} is the operational cost associated with the installment of new assets, while α is the annual discount factor.

The investment of a single transmission corridors t is given by:

$$\begin{aligned} IC_t = & \sum_{n \in \Omega_{N_{t,c}}} \sum_{v \in \Omega_{V_t}} C^{AC}_{t,n,v} y^{AC}_{t,n,v} + C^{DC}_{t} y^{DC}_{t} + \\ & \sum_{n \in \Omega_{N_{t,0,vu}}} C^{VU}_{t,n,v_0} y^{VU}_{t,n,v_0} + \sum_{n \in \Omega_{N_{t,0,rew}}} C^{REW}_{t,n,v_0} y^{REW}_{t,n,v_0} + \\ & \sum_{n \in \Omega_{N_{t,0,pst}}} C^{PST}_{t,n,v_0} y^{PST}_{t,n,v_0} + \sum_{n \in \Omega_{N_{t,0,tcsc}}} C^{TCSC}_{t,n,v_0} y^{TCSC}_{t,n,v_0} \\ & \qquad \forall t \in \Omega_T \end{aligned} \tag{4.7}$$

Where the parameters C are the costs associated with the correspondent expansion measures or the installations of new lines.

Operational costs arise at a single grid snapshot of a single year. Hence, the appropriate analysis of expansion and operational costs requires weighting operational costs within the objective function. This weighting is done by calculating operational costs as perpetual annuity to compare operational costs of one single year with long-term expansion costs.

The operational costs OC contain costs for congestion management interventions ($OC^{CM,u}$) as well as load shedding and generation curtailment ($OC^{Slack,u}$).

$$OC = (1 + \frac{1}{\alpha}) \sum_{u \in \Omega_U} W^{CM,u} (OC^{CM,u} + OC^{Slack,u}) \tag{4.8}$$

Congestion management interventions are distinguished in those for redispatch of conventional power plants and curtailment of renewable energies.

$$OC^{CM,u} = \sum_{g \in \Omega_{PP}} C_g^{PP} (\Delta p_g^{+,u} - \Delta p_g^{-,u}) + C^{RES} \sum_{g \in \Omega_{RES}} \Delta p_g^{-,u} \qquad \forall u \in \Omega_U \tag{4.9}$$

($OC^{Slack,u}$) are modelled as node-specific slack variables to ensure the solvability of the optimisation problem.

$$OC^{Slack,u} = C_r \sum_{cs \in \Omega_{CS}} \sum_{k \in \Omega_K} (r_{g_k}^{u,cs} + r_{d_k}^{u,cs}) \qquad \forall u \in \Omega_U \tag{4.10}$$

4.3.4 Investment constraints

The investment problem deals with restrictions limiting the construction of new assets due to mutual interdependencies between different measures. On the one hand, the expansion costs can depend on the order in which the assets are placed and, on the other hand, the number of measures which can be realised per circuit is limited.

In context of constructing new AC circuits, it is differentiated between the reinforcement of existing and the development of new transmission corridors. Developing new transmission corridors requires the installation of new line towers whereas reinforcing existing ones requires only an upgrade of existing line towers. Furthermore, costs for upgrading an existing one depend on the number of circuits being already installed within the corridor taking all available voltage levels into account. Therefore, it has to be ensured that per line tower place only one circuit of the available voltage levels can be installed:

$$\sum_{v \in \Omega_{V_t}} y_{t,n,v}^{AC} \leq 1 \qquad \forall t \in \Omega_T, \forall n \in \Omega_{N_{t,c}} \tag{4.11}$$

Furthermore, the order in which parallel circuits can be constructed has to be restricted. A circuit n can only be placed when the circuit $n-1$ is already installed.

$$\sum_{v \in \Omega_{V_t}} y_{t,n,v}^{AC} - \sum_{v \in \Omega_{V_t}} y_{t,n-1,v}^{AC} \leq 0 \qquad \forall t \in \Omega_T, \forall n \in \Omega_{N_{t,c}}, n \geq 1 \tag{4.12}$$

It is assumed that each circuit can only be expanded or reinforced by one technological measure. Hence, either a parallel circuit can be installed, the voltage level can be upgraded, the circuit can be re-wired, a PST can be placed in series or a TCSC can be installed serially. Nevertheless, the restriction allows the parallel placement of more than one parallel measure.

$$y^{AC}_{t,n_c,v_0} + y^{VU}_{t,n,v_0} + y^{REW}_{t,n,v_0} + y^{PST}_{t,n,v_0} + y^{TCSC}_{t,n,v_0} \leq 1 \qquad \forall t \in \Omega_T, \forall n_c \in \Omega_{N_{t,c}}, \forall n \in \Omega_{N_{t,0}} \tag{4.13}$$

4.3.5 Operational constraints

The operational variables are constrained by physical law, investment choices and functional limit of the expansion measures.

Kirchhoff's first law imposes that the power injected into a node is equal to the power ejected at the same node:

$$\sum_{t\in\Omega_{T_k}} \sum_{n\in\Omega_{N_t}} \sum_{v\in\Omega_{V_t}} f^{AC,u,cs}_{t,n,v} + \sum_{t\in\Omega_{T_k}} f^{DC,u}_{t} + P^u_{g_k} - r^{u,cs}_{g_k} + \sum_{g\in\Omega_{PP_k}} \Delta p^{+,u}_g = P^u_{d_k} - r^{u,cs}_{d_k} + \sum_{g\in\Omega_{G_k}} \Delta p^{-,u}_g \qquad \forall k \in \Omega_K, \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.14}$$

Here, Ω_{T_k} is the set of transmission corridors incident on node k, $P^u_{g_k}$ is the power produced by generation unit g_k at grid snapshot u while $P^u_{d_k}$ is the power demand of the node at grid snapshot u. Signs of $f^{AC,u,cs}_{t,n,v}$ and $f^{DC,u}_{t}$ are taken with the usual sign convention.

The power flowing in a line is formulated separately for existing and candidate lines as well as for lines those voltage level can be upgraded

$$f^{AC,u,cs}_{t,n,v_0} - \gamma^{AC}_{t,n,v_0}(\theta^{u,cs}_{kf_{t,n,v_0}} - \theta^{u,cs}_{kt_{t,n,v_0}} + \theta^{PST,u}_{t,n,v_0} + \theta^{TCSC,u}_{t,n,v_0}) = 0 \qquad \forall t \in \Omega_T, \forall n \in \Omega_{N_{t,0}} \setminus \Omega_{N_{t,0,vu}}, \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.15}$$

Define the flow for the existing line minus the ones that can be voltage upgraded. γ^{AC}_{t,n,v_0} is the susceptance of the line, $\theta^{u,cs}_{kf_{t,n,v_0}}$ and $\theta^{u,cs}_{kt_{t,n,v_0}}$ are the voltage phase angle of the starting and ending node of the line, respectively. $\theta^{PST,u}_{t,n,v_0}$ and $\theta^{TCSC,u}_{t,n,v_0}$ take into account the possibility to install voltage phase transformer and thyristor controlled series compensators in series to the circuit.

For candidate lines, the flow is defined using a big M value so to limit the flow only if the line is constructed.

$$|f^{AC,u,cs}_{t,n,v} - \gamma^{AC}_{t,n,v}(\theta^{u,cs}_{kf_{t,n,v}} - \theta^{u,cs}_{kt_{t,n,v}})| \leq M(1 - y^{AC}_{t,n,v}) \qquad \forall t \in \Omega_T, \forall n \in \Omega_{N_{t,c}}, \forall v \in \Omega_{V_t}, \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.16}$$

Upgrading the voltage level of a line requires the construction of a new line with an increased voltage level and different reactance as well as the deconstruction of the existing one. Both measures are indicated by the same decision variable y^{VU}_{t,n,v_0}.

$$|f_{t,n,v_0}^{AC,u,cs} - \gamma_{t,n,v_0}^{AC}(\theta_{kf_{t,n,v_0}}^{u,cs} - \theta_{kt_{t,n,v_0}}^{u,cs} + \theta_{t,n,v_0}^{PST,u,cs} + \theta_{t,n,v_0}^{TCSC,u,cs})| \leq My_{t,n,v_0}^{VU}$$
$$\forall t \in \Omega_T, \forall n \in \Omega_{N_{t,0,vu}}, \forall u \in \Omega_U, \forall cs \in \Omega_{CS}$$
$$|f_{t,n,v_{vu}}^{AC,u,cs} - \gamma_{t,n,v_{vu}}^{AC}(\theta_{kf_{t,n,v_0}}^{u,cs} - \theta_{kt_{t,n,v_0}}^{u,cs})| \leq M(1 - y_{t,n,v_0}^{VU})$$
$$\forall t \in \Omega_T, \forall n \in \Omega_{N_{t,0,vu}}, \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.17}$$

If variable y_{t,n,v_0}^{VU} is set to 0, the first equation limit the power flow of the pre-existing line, if is set to 1, the the power of the new line is limited.

To compute the voltage phase angle differences, a node is choosen as reference node and the correspondent angle is set to 0:

$$\theta_{k_{ref}}^{u,cs} = 0 \qquad \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.18}$$

The voltage angle of each node is limited by an maximum voltage angle:

$$|\theta_k^{u,cs}| \leq \theta^{max} \qquad \forall k \in \Omega_K, \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.19}$$

The flow on a line, which can't be re-wired, is limited by the maximum transmission capacity:

$$|f_{t,n,v_0}^{AC,u,cs}| \leq f_{t,n,v_0}^{AC,max}$$
$$\forall t \in \Omega_T, \forall n \in \Omega_{N_{t,0}} \setminus \Omega_{N_{t,0,rew}}, \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.20}$$

The maximum flow on AC circuits, which can be re-wired, is formulated under consideration of the binary variable indicating the re-wiring status. In the case of re-wiring the circuit, the maximum capacity is increased, otherwise it is restricted to the original transmission capacity.

$$|f_{t,n,v_0}^{AC,u,cs}| \leq f_{t,n,v_0}^{AC,max}(1 - y_{t,n,v_0}^{VU}) + (f_{t,n,v_0}^{REW,max} - f_{t,n,v_0}^{AC,max})y_{t,n,v_0}^{REW}$$
$$\forall t \in \Omega_T, \forall n \in \{\Omega_{N_{t,0,rew}} \cup \Omega_{N_{t,0,vu}}\} \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.21}$$

The first term on the right hand-side ensure that the power flow on the existing line is set to 0 in case the line is deconstructed and replaced by the line with the increased voltage level.

The power flowing on the new constructed lines is limited to the maximum one taking the corresponding construction status into account:

$$|f_{t,n,v}^{AC,u,cs}| \leq f_{t,n,v}^{AC,max} y_{t,n,v}^{AC}$$
$$\forall t \in \Omega_T, \forall n \in \Omega_{N_{t,c}}, \forall v \in \Omega_{V_t}, \forall u \in \Omega_U, \forall cs \in \Omega_{CS}$$
$$|f_{t,n,v_{vu}}^{AC,u,cs}| \leq f_{t,n,v_{vu}}^{AC,max} y_{t,n,v_0}^{VU}$$
$$\forall t \in \Omega_T, \forall n \in \Omega_{N_{t,0,vu}}, \forall u \in \Omega_U, \forall cs \in \Omega_{CS} \tag{4.22}$$
$$|f_t^{DC,u}| \leq f_t^{DC,max} y_t^{DC} \qquad \forall t \in \Omega_T, \forall u \in \Omega_U$$

The outage of a line is simulated by reproducing the grid snapshot and forcing the power flow of the line to 0:

$$f^{AC,u,cs}_{t_{cs},n_{cs},v_{cs}} = 0 \qquad \forall t \in \Omega_T, \forall n \in \Omega_{N_t}, \forall v \in \Omega_{V_t}, \forall u \in \Omega_U \tag{4.23}$$

The voltage phase angle for the PST and the TCSC, if construted, are limited by an upper bounds:

$$\begin{aligned} |\theta^{PST,u}_{t,n,v_0}| &\leq \theta^{PST,max} y^{PST}_{t,n,v_0} && \forall t \in \Omega_T, \forall n \in \Omega_{N_{t,0,pst}}, \forall u \in \Omega_U \\ |\theta^{TCSC,u}_{t,n,v_0}| &\leq \theta^{TCSC,max}_{t,n,v_0} y^{TCSC}_{t,n,v_0} && \forall t \in \Omega_T, \forall n \in \Omega_{N_{t,0,tcsc}}, \forall u \in \Omega_U \end{aligned} \tag{4.24}$$

Positive redispatch for traditional power plant is caped by the difference between the maximum power output of the plan and the power output at a given grid snapshot:

$$0 \leq \Delta p^{+,u}_g \leq P^{max}_g - P^u_g \qquad \forall g \in \Omega_{PP}, \forall u \in \Omega_U \tag{4.25}$$

In a similar way, the negative redispatch for the traditional power plants is caped by the maximum between the difference of the power output at a given grid snapshot and the minimum power output and 0:

$$0 \leq \Delta p^{-,u}_g \leq \max(P^u_g - P^{min}_g, 0) \qquad \forall g \in \Omega_{PP}, \forall u \in \Omega_U \tag{4.26}$$

For renewable resources, the negative redispatch is caped by the power output at a given grid snapshot:

$$0 \leq \Delta p^{-,u}_g \leq P^u_g \qquad \forall g \in \Omega_{RES}, \forall u \in \Omega_U \tag{4.27}$$

4.4 Solution Method

The mathematical formulation given in the previous section can be solved using a general purpose MILP solver. Though nowadays effective commercial solvers, that include highly sophisticated tools (e.g., preprocessing, heuristics, cut generation procedures, . . .) are available on the market, the direct application of a solver to this formulation seems to be unpractical as the number of variables and constraints in the model is typically very large for real instances. For this reason, we developed a solution approach, described in the next section, that is based on a Benders decomposition. To improve the performances of the approach, we also use a metaheuristic algorithm based (see Section 4.4.3) that proved to be extremely effective in practice.

4.4.1 A Benders decomposition approach

Benders decomposition is a solution approach that is typically used to attack large scale optimization problems by exploiting the structure of the constraint matrix. In particular, the method is effective for problems that can be easily decomposed into subproblems when the value of a limited number of *complicating variables* has been estabilished.

The general scheme of a Benders decomposition consists of the iterative approach in which, at each iteration:

- a *master problem* is solved and determines the candidate value for the complicating variables; and
- given the current value of the complicating variables, one or more *subproblems* are solved, to check feasibility and cost of the proposed master solution.

The solution of the subproblems may produce additional cuts to be added to the master problem, in which case the process is iterated. The algorithm halts when an iteration is encountered in which the solution of the subproblems produces no cuts to be added. A key aspect is that the master problem works in the space of the complicating variables only, i.e., it is much smaller than the original problem. For this reason, the master can therefore be solved efficiently, and the cuts must be expressed in terms of the complicating variables only.

In the TEP model, the complicating variables are the the strategic variables y that control the installation of new assets. By fixing a value for each strategic variable, the remaining problem, involving operational variables only, reduces to a pure LP problem that can be easily resolved through decomposition. In particular, one can identify $|\Omega_U|$ subproblems, each associated with a specific snapshot.

As subproblems determine the cost of the solution, it is convenient to introduce in the master additional non-negative variables z_i ($i \in \Omega_U$), to take into account the cost of each subproblem. Accordingly, the master problem is defined by objective function

$$\min \quad IC + \sum_{i \in \Omega_U} z_i$$

under constraints (4.11) - (4.13). The initial formulation of the master includes no constraints that involves the z variables (but for their sign). The correct value of these variables will be estabilished by the cuts that are dynamically generated by the separation phase. Finally, note that the master problem includes integer (actually, binary) variables. This integrality requirement will be considered later, i.e., assume that one is interested in solving the LP relaxation of the master problem only.

Each subproblem is defined by objective function (4.8) and by constraints (4.9), (4.10) and (4.14) - (4.27). Note that, subproblems are solved for a given tentative value of the complicating variables. As these variables appear on the right-hand side of the constraints only, the subproblems are purely LP, i.e., they can be solved efficiently in practice. Moreover, as subproblems are always feasible (though, possibly, having a very large cost), all cuts that are generated are *optimality cuts*, i.e., inequalities that are used to correctly define the cost of the subproblems. In particular, denote by

$$\min\{g^i x : E^i x \geq b^i - D^i \bar{y}\}$$

the i-th subproblem with optimal value q_i. If $q_i > z_i$, we generate and add to the master the following cut

$$z_i \geq \pi^i (b^i - D^i y) \tag{4.28}$$

where π^i denotes the vector of dual variables associated with the constraints of the i-th subproblem.

Our solution method is described in Algorithm 3. Observe that, for the correctness of the method, the separation phase is required only for integer $\bar{y}$ solutions. For this reason, and to avoid the generation of a very large number of cuts, we execute

Algorithm 3 Benders decomposition algorithm

```
function BENDERS
    solve the continuous relaxation of the master problem;
    let (ȳ, z̄) an optimal solution of the relaxation;
    if ȳ is not integer then
        return (ȳ, z̄, infeasible)
    end if
                                                    ▷ separation phase
    feasible ← true
    for all i ∈ Ω_U do
        fix y := ȳ and update the right-hand side;
        solve subproblem i and let q_i its optimal value;
        if q_i > z̄_i then
            add z_i ≥ π^i (b^i − D^i y) to master;
            feasible ← false
        end if
    end for
    if feasible = true then
        return (ȳ, z̄, feasible)
    end if
    goto 2
end function
```

separation only in case $\bar{y}$ is integer. The procedure returns an optimal solution of the current LP relaxation, and a status indicating whether this solution is feasible or not.

As already mentioned, our Benders decomposition method solves the LP relaxation of the problem, in which the integrality requirement of the y variables has been dropped. To restore these constraints, the scheme is embedded into a branch-and-cut algorithm built on top of the general-purpose MILP solver Gurobi. At each node of the branching tree, a callback function is executed to possibly produce additional cuts that can be added to the master. To avoid the addition of a very large number of cuts, the separation phase is applied only to integer solutions. In particular, if $\bar{y}$ is not integer, a branching is performed to define two descendant nodes that will be explored later; the choice of the branching variable as well as the node selection strategies are left to the solver. If instead $\bar{y}$ is integer, each subproblem is solved to possibly separate new cuts. If some violated cuts have been detected, the master problem is solved again, possibly inducing a branching or another execution of the separation phase. If instead the cost of each subproblem matches with the value of corresponding z variable, the solution is accepted and the incumbent may be updated.

The pseudo-code of the callback function is given in Algorithm 4, that makes use of a parameter z^* denoting the incumbent best solution found so far.

4.4.2 Relaxation

Our computational experiments, that will be discussed in Section 4.5, showed that the direct solution of the mathematical formulation of the problem using a general-purpose MILP solver is very challenging from a computational viewpoint. This is due to the size of the model and to the fact that the objective function includes several costs, that have different relevance in the definition of the total solution cost.

Algorithm 4 Node callback

function CALLBACK(z^*)
  $(\bar{y}, \bar{z}, \text{status}) \leftarrow$ Benders
  set L be the cost of solution $(\bar{y}, \bar{z})$
  if $L \geq z^*$ **then**
    fathom the node;
  end if
  if status = infeasible **then**
    select a fractional $\bar{y}$ value and perform branching
  else
    update the incumbent z^*
  end if
end function

To ease the solution of the model, we consider a relaxation of the problem in which only the slacks variables $r^{u,cs}_{g_k}$ and $r^{u,cs}_{d_k}$ are minimized in the objective function. On the one hand, these variables are associated with the largest by far cost in the objective function. On the other hand, an optimal solution value of this problem is very useful from a practical viewpoint, as it provides two relevant information:

- its value is a valid lower bound on the optimal solution value;
- it provides a tight indication on the minimum amount of demand that cannot be satisfied.

finally, note that an optimal solution of this model is a feasible solution for the complete problem. Plugging all cost terms back in the objective function, one can evaluate the cost of this solution, thus producing an upper bound on the optimal value.

4.4.3 Biased Random Key Genetic Algorithm

As it typically happens for enumerative algorithms, the performances of the method are strongly affected by the availability of primal heuristics able to determine tight upper bounds on the optimal solution value. For this reason, we implemented a metaheuristic based on the Biased Random-Key Genetic Algorithm (BRKGA) framework, to be applied as a warm-start for the solution approach.

BRKGA is a variant of genetic algorithms in which each solution is encoded by a vector of random-keys, each having a value in the interval $[0, 1]$. To evaluate feasibility and cost of an element of the population, the associated vector of random keys is decoded using a problem-specific heuristic, called the *decoder*. This allow to clearly separate the heuristic procedure used to define a solution (which is typically dependent on the problem at hand) from the evolutionary process, which is instead absolutely general.

The initial population is generated with p random vectors. At each iteration, the fitness of each solution in the population is computed by means of the decoder. Then, the population is partitioned in two disjoint subset: the first one contains a small fraction of solutions, namely the p_e elements with the best fitness *elite solutions*, whereas the second one includes the remaining $p - p_e$ non elite solutions. The population for the next iteration is obtained as follows:

- including all elite solutions from the current iteration;

- defining a small number of p_m *mutants* to the population;
- generating the remaining $p - p_e - p_m$ elements by means of a crossover.

Mutation is used to avoid of being trapped in local minima. The *parameterized uniform crossover* is applied to a pair of randomly selected elements (one from each subset) as works as follows: given a parameter $\rho_e \in [0, 1]$, each element of the new solution is inherited from the elite parent with probability ρ_e and from the the other parent with probability $1 - \rho_e$. At each iteration, the fitness of each solution is computed and possibly used to update the incumbent solution value. Note that this framework allows for an efficient implementation in which the fitness of the solutions are computed in parallel. The process is halted after a maximum of iterations or in case the incumbent has not been updated for a given number of iterations.

Algorithm 5 Decoder

function DECODER(vector, subs, n_{ac})
    ▷ **step1:** define the strategic variables
    for all $t \in \Omega_T$ **do**
        extract entries A_t and B_t from vector;
        $prev \leftarrow true$
        for $n = 1$ **to** $|\Omega_{N_{t,c}}|$ **do**
            if $prev = true$ **then**
                let a^{nt} be the set of A^t entries associated with position n
                $val \leftarrow \max a^{nt}$
                $v \leftarrow \arg\max a^{nt}$
                if $val \geq \theta$ **then**
                    $\bar{y}^{AC}_{t,n,v} \leftarrow 1$
                    $new\_line = true$
                else
                    $prev = false$
                end if
            end if
            if $new\_line = false$ **then**
                let b^{nt} be the set of B^t entries associated with position n
                $val \leftarrow \max b^{nt}$
                $u \leftarrow \arg\max b^{nt}$ ▷ $u \in \{VU, REW, PST, TCSC\}$
                if $val \geq \theta$ **then**
                    $\bar{y}^{u}_{t,n,v_0} \leftarrow 1$
                end if
            end if
        end for
    end for
    ▷ **step2:** complete the solution
    for all $s \in subs$ **do**
        use $\bar{y}$ to fix s right-hand side
        solve(s) **return** $c^T y + \sum_{s \in subs} s.obj$
    end for
end function

In our implementation for solving TEP, each solution is encoded using a vector of n elements, each corresponding to a strategic variable y. A decoding procedure, described in Algorithm 5, is used to define a feasible solution associated with a given key vector. The procedure operates in two phases: in the first phase, it determines the value of each y variable, taking into account the investment constraints described in Section 4.3.4. In the second phase these value are used to define the right-hand side of the constraints that appear in the subproblems, that are solved using a MILP solver.

Note that, in the first phase the procedure makes use of an input a parameter θ (to be discussed later), and exploits the fact that each investment constraint refers to one transmission corridor. Accordingly, the procedure considers one corridor at a time and partitions the key vector in $|\Omega_T|$ parts, each composed of $|\Omega_{V_t}| \times |\Omega_{N_{t,c}}| + \alpha|\Omega_{N_{t,0}}|$ entries, where α is the number of possible upgrades for the existing lines ($\alpha = 4$ in our formulation).

To easy the notation, we let t be the current transmission corridor, and denote by $A^t \in [0,1]^{|\Omega_{V_t}|\times|\Omega_{N_{t,c}}|}$ the set of entries associated to corridor t and variables that model the construction of new AC circuits, and by $B^t \in [0,1]^{\alpha\times|\Omega_{N_{t,0}}|}$ the remaining values. Constraints (4.12) impose that a new AC circuit can be installed in position $n \in \Omega_{N_{t,c}}$ only if a circuit has been installed in position $n-1$ as well. For this reason, circuits are considered one at a time by increasing position index, and the installation of the current circuit is evaluated only in case the previous circuit has been installed. In this case, as constraints (4.11) impose that at most one available voltage level is used, we evaluate all the A^t entries associated with the current transmission corridor and circuit, and consider the one with maximum value. The corresponding variable is set to 1 if and only if this value is not below the threshold; all the other variables associated with transmission corridor t and circuit n are set to 0. Finally, constraints (4.13) allow at most one variable associated with an entry in B^t be one, only in case no new line is constructed in the same transmission corridor. In this case as well, we scan the corresponding entries and possibly set to one the y variable associated with the entry having maximum value, provided it is not below the threshold.

Once the strategic variables have been fixed, the problem can be decomposed into subproblems, each one being an LP. Solving these subproblems one at a time allows to produce a complete solution that can be used to possibly update the incumbent.

4.5 Computational Results

Computational test were performed using 4 real-world instances of different size that we received from our partners within PlaMES. In Table 4.1 we report the characteristics of each instance, in terms of number of nodes, corridors, branches, snapshots, and outages; in addition, we give the number of constraints and variables of the associated complete formulation.

TABLE 4.1: Characteristic of the instances

name	nodes	corridors	branches	snapshot	outages	constraints	variables
inst1	120	254	439	6	3	18318	10235
inst2	120	477	778	11	149	335254	146886
inst3	120	477	764	20	340	725575	319216
inst4	1589	1702	6379	20	812	12401547	6558991

All experiments were executed an AMD ryzen 3700x 8c/16t running at @3.6Ghz and equipped with 32GB of RAM.

4.5.1 Exact methods

For what concerns the exact solution of the problem, we consider the following algorithms:

- single model: the direct application of solver Gurobi to the complete formulation;
- Benders: the exact method based on Benders' decomposition described in Section 4.4.1;
- Benders*: this is a variant of the previous scheme in which, at the root node, separation is carried out also for non-integer solutions.

Each method was run in single-thread mode with a time limit equal to 10,000 seconds per instance. Table 4.2 reports the corresponding results and provides, for each algorithm, the value of the best solution found, the best lower bound, the associated percentage gap, the computing time and the number of nodes.

The table shows that, though Gurobi is a state-of-the-art solver for MILPs, it can successsfully be used for solving the first instance only. On the other hand, Benders decomposition allows to compute an optimal solution for other two instances within the given time limit. Separating Benders cuts in a more aggressive way at the root node gives some improvement as it reduces the internal gap of the algorithm before enumeration starts. Nevertheless, all methods run out of memory (without even computing a feasible solution) for the last instance, in which the large number of outages produces a very large formulation.

4.5.2 Heuristics

Our second set of experiments concern the approximate solution of the TEP problem. To this aim, we consider the following algorithms:

- only delta: the cost of the heuristic solution produced solving the relaxation introduced in Section 4.4.2;
- BRKGA: this is the result computed with the biased random key genetic algorithm;

When comparing heuristics, we used a reduced time limit equal to 600 seconds. Table 4.3 reports, for each algorithm, the value of the best solution found, the associated percentage gap with respect to the optimal value (best known solution for the last instance), and the required computing time.

TABLE 4.2: Results for exact algorithms. Time limit = 10,000 seconds, * = out of memory

	single model					Benders					Benders*				
name	obj	lb	%gap	time	#node	obj	lb	%gap	time	#node	obj	lb	%gap	time	#node
inst1	188.31	188.31	0.00	25	1672	188.31	188.31	0.00	18	3537	188.31	188.31	0.00	9	3227
inst2	8.31e+12	-334.45	100.00	limit	0	2.50e+9	2.50e+9	0.00	8437	29937	2.50e+9	2.50e+9	0.00	6214	81892
inst3	1.81e+13	8.87e+8	99.99	limit	0	3.21e+10	3.21e+10	0.00	3768	24851	3.21e+10	3.21e+10	0.00	1655	1923
inst4	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

TABLE 4.3: Results for heuristic algorithms. Time limit = 600 seconds, * = out of memory

	only delta			BRKGA		
name	obj	%gap	time	obj	%gap	time
inst1	188.31		25	3.02e+11		1248
inst2	8.31e+12		limit	1.57e+14		limit
inst3	1.81e+13		limit	2.35e+14		limit
inst4	*		*	*		*

Acknowledgements

This research is part of the project PlaMES (Integrated Planning of Multi- Energy Systems). PlaMES has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 863922.

Bibliography

Beasley, J. E. (1985). "An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure". In: *Operations Research* 33.1. Publisher: INFORMS, pp. 49–64. ISSN: 0030-364X. URL: https://www.jstor.org/stable/170866 (visited on 03/04/2023).

Chen, C. S., S. M. Lee, and Q. S. Shen (Jan. 1995). "An analytical model for the container loading problem". en. In: *European Journal of Operational Research* 80.1, pp. 68–76. ISSN: 0377-2217. DOI: 10.1016/0377-2217(94)00002-T. URL: https://www.sciencedirect.com/science/article/pii/037722179400002T (visited on 02/22/2023).

Crainic, Teodor Gabriel, Guido Perboli, and Roberto Tadei (June 2009). "TS2PACK: A two-level tabu search for the three-dimensional bin packing problem". en. In: *European Journal of Operational Research* 195.3, pp. 744–760. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2007.06.063. URL: https://www.sciencedirect.com/science/article/pii/S0377221707010995 (visited on 01/15/2023).

Faroe, Oluf, David Pisinger, and Martin Zachariasen (Aug. 2003). "Guided Local Search for the Three-Dimensional Bin-Packing Problem". In: *INFORMS Journal on Computing* 15.3. Publisher: INFORMS, pp. 267–283. ISSN: 1091-9856. DOI: 10.1287/ijoc.15.3.267.16080. URL: https://pubsonline.informs.org/doi/abs/10.1287/ijoc.15.3.267.16080 (visited on 01/15/2023).

Fekete, Sandor P. and Joerg Schepers (Oct. 2003). *A combinatorial characterization of higher-dimensional orthogonal packing*. arXiv:cs/0310032. DOI: 10.48550/arXiv.cs/0310032. URL: http://arxiv.org/abs/cs/0310032 (visited on 01/15/2023).

Fekete, Sandor P., Joerg Schepers, and Jan C. van der Veen (Apr. 2006). *An exact algorithm for higher-dimensional orthogonal packing*. arXiv:cs/0604045. DOI: 10.48550/arXiv.cs/0604045. URL: http://arxiv.org/abs/cs/0604045 (visited on 01/15/2023).

Franken, Marco et al. (June 2019). "Transmission Expansion Planning Considering Detailed Modeling of Expansion Costs". In: *2019 IEEE Milan PowerTech*, pp. 1–6. DOI: 10.1109/PTC.2019.8810437.

Gilmore, P. and Ralph Gomory (Feb. 1965). "Multi-Stage Cutting Stock Problems of Two or More Dimensions". In: *Operations Research* 13. DOI: 10.1287/opre.13.1.94.

Gilmore, P. C. and R. E. Gomory (Dec. 1961). "A Linear Programming Approach to the Cutting-Stock Problem". In: *Operations Research* 9.6. Publisher: INFORMS, pp. 849–859. ISSN: 0030-364X. DOI: 10.1287/opre.9.6.849. URL: https://pubsonline.informs.org/doi/10.1287/opre.9.6.849 (visited on 03/27/2023).

Gonçalves, José Fernando and Mauricio G. C. Resende (Oct. 2013). "A biased random key genetic algorithm for 2D and 3D bin packing problems". en. In: *International Journal of Production Economics* 145.2, pp. 500–510. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2013.04.019. URL: https://www.sciencedirect.com/science/article/pii/S0925527313001837 (visited on 01/30/2023).

Lai, K. K. and Jimmy W. M. Chan (Jan. 1997). "Developing a simulated annealing algorithm for the cutting stock problem". en. In: *Computers & Industrial Engineering* 32.1, pp. 115–127. ISSN: 0360-8352. DOI: 10.1016/S0360-8352(96)00205-7. URL:

https://www.sciencedirect.com/science/article/pii/S0360835296002057 (visited on 01/31/2023).

Lodi, Andrea, Silvano Martello, and Daniele Vigo (Sept. 2002). "Heuristic algorithms for the three-dimensional bin packing problem". en. In: *European Journal of Operational Research* 141.2, pp. 410–420. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(02)00134-0. URL: https://www.sciencedirect.com/science/article/pii/S0377221702001340 (visited on 01/30/2023).

— (Sept. 2004). "Models and Bounds for Two-Dimensional Level Packing Problems". en. In: *Journal of Combinatorial Optimization* 8.3, pp. 363–379. ISSN: 1573-2886. DOI: 10.1023/B:JOCO.0000038915.62826.79. URL: https://doi.org/10.1023/B:JOCO.0000038915.62826.79 (visited on 02/07/2023).

Lodi, Andrea and Michele Monaci (Jan. 2003). "Integer linear programming models for 2-staged two-dimensional Knapsack problems". en. In: *Math. Program., Ser. B* 94.2, pp. 257–278. ISSN: 1436-4646. DOI: 10.1007/s10107-002-0319-9. URL: https://doi.org/10.1007/s10107-002-0319-9 (visited on 02/07/2023).

Mahdavi, Meisam et al. (Sept. 2019). "Transmission Expansion Planning: Literature Review and Classification". en. In: *IEEE Systems Journal* 13.3, pp. 3129–3140. ISSN: 1932-8184, 1937-9234, 2373-7816. DOI: 10.1109/JSYST.2018.2871793. URL: https://ieeexplore.ieee.org/document/8482504/ (visited on 01/15/2023).

Martello, Silvano, David Pisinger, and Daniele Vigo (Feb. 1998). "The Three-Dimensional Bin Packing Problem". In: *Operations Research* 48. DOI: 10.1287/opre.48.2.256.12386.