UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea Magistrale in **Ingegneria Gestionale**

ESPERIMENTI COMPUTAZIONALI SU FORMULAZIONI NON-LINEARI DEL PROBLEMA DI BIN-PACKING

COMPUTATIONAL EXPERIMENTS ON NON-LINEAR FORMULATIONS FOR BIN-PACKING PROBLEM

Relatore: Chiar.mo

Prof. **Fabrizio Marinelli**

Tesi di Laurea di:

Riccardo Renzi

A.A. **2019 /2020**

Contents

1 Introduction **2**
- 1.1 Problem general caractheristic . . . 2
- 1.2 mathematical programming language AMPL . . . 4

2 Analysis of integer linear formulations **6**
- 2.1 Kantorovich model . . . 6
  - 2.1.1 bb and sim . . . 8
- 2.2 Dyckhoff model . . . 10
- 2.3 Arc Flow Formulation . . . 14
- 2.4 Column generation on CSP and BPP . . . 17
  - 2.4.1 Dantzig Wolfe decomposition on Kantorovich formulation . . . 17
  - 2.4.2 Column generation on Gilmore Gomory . . . 20
  - 2.4.3 Branch(and cut) and price . . . 22

3 Experimental evaluation on exact model **24**
- 3.1 Heuristic Upper Bound or Primal Bound . . . 24
- 3.2 Heuristic Lower Bound or Dual Bound . . . 25
- 3.3 Code in C++ for the bounds . . . 26
- 3.4 Solution of benchmark instances . . . 31
  - 3.4.1 Analysis of the results obtained . . . 33
  - 3.4.2 Dyckoff Hard istances . . . 35

4 From integer to continuous model **37**
- 4.1 binarization of the models . . . 37
  - 4.1.1 experimental on binary models . . . 39
- 4.2 Continuous Non-linear model . . . 43

5 Non linear solver **47**
- 5.1 how COUENNE works . . . 49
  - 5.1.1 Reformulation . . . 50

5.1.2 Linearization 51
5.1.3 Branching 54

6 Computation experiment on non-linear solver 56

To my mother, Patrizia,
In memory of my father, Giovanni,
In memory of my grandparents,

Abstract

This dissertation is the result of an internship period spent from April 2018 to July 2018 at the FEUP(Faculdade de Engenharia da Universidade do Porto), in Porto(Portugal). The domain of the work carried on is Operation Research.

The thesis concern about the analyzing of the Linear and exact formulation of the bin packing problem, the obtaining of the model in a mathematical programming language for performing computational experiments on given set of instances, the binarization of the variable and the use of non linear constraint to remove the binary bond.

The final purpose is to obtain continuous formulation without integer variable, to compute and compare with the achieved solution, in terms of time and optimality.

Extended summary (italian)

La tesi è frutto del periodo di tirocinio svolto all' estero tra aprile e luglio 2018, svolto alla facoltà **FEUP (Faculdade de Engenharia da Universidade do Porto)** a Porto in Portogallo, grazie al progetto europeo Erasmus+Traineeship.
Il problema che si è affrontato durante il periodo e in tutto il documento è quello del problema di bin-packing espansione del problema dello zaino e caso specifico del problema di Cutting Stock, in particolare l'obiettivo, dato un insieme di elementi aventi un certo valore di peso o ingombro a una dimensione e un insieme di contenitori con un valore di capacità massima, è quello di ridurre il numero di contenitori utilizzati per impacchettare tutti gli elementi.
Una volta analizzati i modelli a disposizione nella letteratura, si è sviluppato attraverso la programmazione matematica quelli che avevano una formulazione esatta (i modelli che restituivano il valore ottimo una volta risolti) e si sono analizzati i risultati ottenuti in termini di esattezza e tempo di risoluzione.
L'obiettivo a partire dall' analisi iniziale è stato quello di realizzare modelli matematici continui del problema di bin-packing e osservare quali erano i risultati e confrontarli con quelli ottenuti dagli esperimenti fatti.
Per raggiungere l'obiettivo desiderato si sono dovute modificare in primo luogo le caratteristiche delle variabili passando da intere a binarie, in secondo luogo si sono potuti eliminare aggiungendo dei vincoli che costringono la variabile interessata ad assumere valori tra 0 e 1, attraverso l'uguaglianza $x = x^2$ rendendo il problema Non lineare a causa della moltiplicazioni delle variabili.
Attraverso i modelli ottenuti si sono infine svolti degli esperimenti computazionali su solutori non lineari per osservare con quale efficienza ed efficacia gli stessi operano e i modelli si comportano.

Chapter 1

Introduction

Given a set N= {1,...,n} of object, everyone with weight $w_i > 0$ and a set M= {1,...,m} of bin with standard capacity W, the bin-packing problem consists in the allocation of all the object inside the bin without exceed the bin capacity and the minimization of the number of bin used.
The applications of the problem are many, from the basic need of filling up containers, the optimization of the truck's capacity for the reduction of the cost, management of file backup in media, to ICT with the mapping in Field-programmable gate array semiconductor chip design and the reduction of memory requirement by the virtual machines in a server.
When the number of part of a specific item is more than 1 and is expressed by the parameter of the instance the model should have a constraint to respect the request of item and that is called Cutting Stock Problem(CSP), the bin packing problem could be seen as a specific case of the CSP when the request of every item is 1.

1.1 Problem general caractheristic

The one dimensional BPP in the literature is classified as 1/V/I/M and the CSP 1/V/I/R.
The classification came from the study of Dyckoff in 1990 that give 4-field notation, trying to divide the different problem relying on the characteristic, the meaning of every space **1/2/3/4** are:

1. Dimensionality
(1) One-dimensional.
(2) Two-dimensional.
(3) Three-dimensional.

(N) N-dimensional with N $>$ 3.

2. Kind of assignment
(B) All objects and a selection of items.
(V) A selection of objects and all items.

3. Assortment of large objects
(O) One object.
(I) Identical figure.
(D) Different figures.

4. Assortment of small items
(F) Few items (of different figures).
(M) Many items of many different figures.
(R) Many items of relatively few different (non-congruent) figures.
(C) Congruent figures.

Meaning that the 1-D BPP is a one dimensional problem with a selection of objects and all item that are identical figure organized in many items of many different figure while CSP have many items of relatively few different figures.
The problem is NP-Complete, finding an admissible solution is possible if $w_k \leq W(\forall k \in N)$ and $m \geq n$, in this case the easiest solution is putting one object in every bin, but if m $<$ n the problem of founding a solution becomes NP-Hard. following some theorem that established the NP-hardness **(unless P = NP)**:

Theorem 1. *:*
deciding if BPP has a solution with 2 bins is NP-complete

Proof. Let $\{l_1, .., l_n\}$ be an instance of *PARTITION*. Consider an instance of BPP with items $\{l_1, .., l_n\}$ and 2 bins $\sum l_i/2$ long. The BPP instance is a yes-instance if the *PARTITION* instance is a yes-instance □

Corollary 1. *:*
BPP is NP-hard to approximate with ratio $< 3/2$

Proof. an approximation algorithm with ratio $\delta < 3/2$ should solve a yes-instance of *PARTITION* in polynomial time. □

Theorem 2. *:*
deciding if BPP has a solution with m bins is strongly NP-complete

Proof. exists a polynomial reduction from *3-DIMENSIONAL MATCHING* □

1.2 mathematical programming language AMPL

AMPL (A Mathematical Programming Language) is an algebraic modeling language to describe and solve high-complexity problems for large-scale mathematical computing (i.e., large-scale optimization and scheduling-type problems).
One advantage of AMPL is the similarity of its syntax to the mathematical notation of optimization problems. This allows for a very concise and readable definition of problems in the domain of optimization. Many modern solvers available on the NEOS Server accept AMPL input. According to the NEOS statistics AMPL is the most popular format for representing mathematical programming problems. AMPL supports a wide range of problem types, among them:

A Linear programming
B Quadratic programming
C Nonlinear programming
D Mixed-integer programming
E Mixed-integer quadratic programming with or without convex quadratic constraints
F Mixed-integer nonlinear programming
G Second-order cone programming
H Global optimization
I Semidefinite programming problems with bilinear matrix inequalities
J Complementarity theory problems (MPECs) in discrete or continuous variables
K Constraint programming

Declarations of model entities have the following common form:

entity, name, alias, indexing, body;

where name is an alphanumeric name that has not previously been assigned to an entity by a declaration, alias is an optional literal, indexing is an optional indexing expression, and entity is one of the keywords:

A set
B param
C var
D arc
E minimize
F maximize
G subject to
H node

In addition, several other constructs are technically entity declarations, but are described later; these include environ, problem, suffix and table. The entity may be omitted, in which case subject to is assumed. The body of various declarations consists of other, mostly optional, phrases that follow the initial part. Each declaration ends with a semicolon.

Chapter 2

Analysis of integer linear formulations

2.1 Kantorovich model

The Kantorovich compact model was published in 1939 by Leonid Vitaliyevich Kantorovich a Soviet Mathematician during the third Five Year Plan when the increasing of efficiency in the industry start to pass from improvement of technology to a better organization of planning and production, the model structure for 1-D bin-packing problem is :

$$min \quad \sum_{j=1}^{m} y_j \tag{2.1}$$

$$s.t. \quad \sum_{j=1}^{m} x_{i,j} \geq d_i \quad \forall i \tag{2.2}$$

$$\sum_{i=1}^{n} l_i * x_{i,j} \leq L_j * y_j \quad \forall j \tag{2.3}$$

$$y_j \in \{0,1\} \quad \forall j \tag{2.4}$$

$$x_{i,j} \geq 0, integer \quad \forall i, \forall j \tag{2.5}$$

The variable of the model are:

$y_j = \{$ 1 if the bin k is used, 0 otherwise $\}$
$x_{i,j} \geq 0$ *integer* number of item i allocated in bin k

The objective is the minimization of the number of bin used.
The constraints concern about the respect of the demand of the item with an established width (2) and the capacity of the bin used for the allocation of the item (3).

The AMPL codification to solve Kantorovich model to optimality can be:

1. **param n;**
2. **param m;**
3. **set roll := 1..n; # different roll**
4. **set item := 1..m; # item to obtain**
5. **param w {item} > 0; # leght of item i**
6. **param W {roll} > 0; # leght of roll k**
7. **var x {i in item,k in roll} >= 0 integer; # item cut in roll k**
8. **var y {k in roll} >= 0 binary; # utilization of roll y**
9. **minimize numb : sum{k in roll } y[k] ;**
10. **subject to dem {i in item}: sum {k in roll} x[i,k] >= 1;**
11. **subject to cap {k in roll}: sum i in item w[i]*x[i,k] <= W[k]*y[k];**

This formulation can be very weak, Valerio de Carvalho and Vance report that branch and bound algorithms based on this model failed to solve to optimality some instances that could be tackled with stronger formulation. To understand why is useful to describe the branch and bound algorithms and how it work.

2.1.1 Branch and bound and symmetries in kantorovich model

The branch and bound algorithm is defined on the idea of exploring the space of solution with exclusion of the not interesting one.
The original problem is decomposed (branch) in subproblems smaller and easier to solve, the subproblems that cannot reach the optimal solution are eliminated (bounding).

The branching must:
not exclude the optimal solution of the original problems, without this condition the method could not grant the resolution to optimality.
not match subproblem with the original problem, to make the procedure end in a finite number of step otherwise it could create cycling iteration in the algorithm.
To grant the branching accuracy, every subproblem is defined with the original objective function and the union of the eligible region is coincident with the original problem:
Defined (P) the original problem and $P^1, ..., P^i$ the subproblem,
$P = \cup_i P^i$ the method is applied recursively to the subproblem if they remain hard to solve, all the node that are generated in that way form a structure named enumeration tree.
For example if the problem has 3 variable with integer constraint, the method can generate a tree putting the first variable to 1 for the first subproblem and 0 for the second and iterate it for the other variable, the number at last level of the structure of subsubproblem will be $= 2^3 = 8$.
In general the number of all the subproblem in an enumeration tree is exponential.
The potential of this method is the possibility to cut the subproblem that aren't interesting for the optimal solution, implicitly solving a large number of subproblem depending in what level of the enumeration tree it is.
The process name is bounding, and it works with 2 elements: upper bound and lower bound, with those two elements is possible to analyze the subproblem generated and prune the problem that cannot improve the current solution.

The Kantorovich problem in the branch and bound algorithm in most cases explore a large number of subproblem before the end and considering that the tree is exponential it takes long time to find the optimal.
Following an example of what happens during the exploration of the tree.
Let's consider an instance with capacity of the bin L=12
and some item to pack with weight:
i^z=8, i^{z+1}=7, i^{z+2}=5, i^{z+3}=4
and the optimal solution is x = n+m and contain the pattern:

$y_n = 1$ with $y_n * L \geq x_{n,z} * i^z + x_{n,z+3} * i^{z+3}$
as $x_{n,1} = 0, ..., x_{n,z-1} = 0, x_{n,z} = 1, x_{n,z+1} = 0, x_{n,z+2} = 0, x_{n,z+3} = 1$
and
$y_{n+m} = 1$ with $y_{n+m} * L \geq x_{n+m,z+1} * i^{z+1} + x_{n+m,z+2} * i^{z+2}$
as $x_{n,1} = 0, ..., x_{n,z-1} = 0, x_{n,z} = 0, x_{n,z+1} = 1, x_{n,z+2} = 1, x_{n,z+3} = 0$

so in the optimal solution bin (n) is activated and completely filled with item of weight 8 and 4 since the capacity is 12, at the same way bin (n+m) is filled with item of weight 7 and 5.

Considering the optimal solution, it's easy to understand that permutation of the two patterns don't change the optimal;

$y_n = 1$ with $y_n * L \geq x_{n,z+1} * i^{z+1} + x_{n,z+2} * i^{z+2}$
as $x_{n,1} = 0, ..., x_{n,z-1} = 0, x_{n,z} = 0, x_{n,z+1} = 1, x_{n,z+2} = 1, x_{n,z+3} = 0$
and
$y_{n+m} = 1$ with $y_{n+m} * L \geq x_{n+m,z} * i^z + x_{n+m,z+3} * i^{z+3}$
as $x_{n,1} = 0, ..., x_{n,z-1} = 0, x_{n,z} = 1, x_{n,z+1} = 0, x_{n,z+2} = 0, x_{n,z+3} = 1$

It can be concluded that all the permutation of activated pattern ($y_n = 1$) are equivalent optimal solution spreading all over the leaf of enumeration tree generated on the branch and bound algorithm.
Since $z_L^i \leq x^* \leq z_L^i$ for each subproblem i, the node cannot be prune from the tree and is explored until the last level, symmetries make ineffective the bounding and the algorithm, independently from the exploration strategy adopted, execute a complete enumeration of the problem.

2.2 Dyckhoff model

The Dyckhoff model uses a different interpretation of the problem, in this case the decision variable corresponds to a cutting operation done on a single piece or roll, obtaining a roll with new width, so the roll is divided into two smaller rolls and one of them should have the same width of an ordered size. This operation can be performed on stock pieces or in roll already cut.
This interpretation is call Onecut model and the Dyckhoff one is structured as follows:
Initially are established 3 different sets S, D and R:
In S there are the different widths of the roll $p \in \{W_1, ..., W_m\}$ integer and non negative.
In D are allocated the order widths requested $q \in w_1, ..., w_n$ integer and non negative, and $D \cap S = \emptyset$.
Finally in R are released the residual roll obtained by the cut of p and not shorter than the smaller order width.

The variables are:

$z_k = \{$ 1 if the bin of size Wk is used, 0 otherwise
$x_{p,q} \geq 0$ *integer* With $p \in S \cap R, q \in D, q < p$

The last variable stands for the number of roll of width p divided in pieces of width q and in other pieces of width p - q.

The Dyckoff model published in 1981 for 1-D CSP is:

$$min \quad \sum_{k=1}^{K} W_k * z_k \tag{2.6}$$

$$s.t. \quad z_k + \sum_{p\in D,(p+q)\subset(S\cup R)} y_{(p+q,p)} \geq \sum_{p\in D,p<q} y_{q,p} \quad \forall q \in S \tag{2.7}$$

$$\sum_{p\in(S\cup R),p>q} y_{p,q} + \sum_{j\in D,(j+q)\in(S\cup R)} y_{(j+q,j)} \geq \sum_{i\in D,i<q} y_{q,i} + Nq \quad \forall q\in(D\cup R)-S \tag{2.8}$$

$$y_{p,q} \geq 0, integer \quad \forall(p \in (S \cup R), q \in D, q < p) \tag{2.9}$$

$$z_k \leq B_k \quad \forall k \tag{2.10}$$

$$z_k \geq 0, integer \quad \forall k \tag{2.11}$$

The parameter N_q is the demand of item of size q and B_k is the number of roll k with the same size.
The constraint (7) impose that the number of bin used with the same width q plus the number of rolls cut in that width q has to be equal or more of the number of the rolls cut from the with q, combined with (9) became a capacity constraint where all of the number of rolls cut of one width are less of the availability of them.
To obtain the relation in the modelling it became necessary the introduction of $q = W_k$ clause in (7)
While the demand constraint is obtained with (8), it concerns about the number of roll cut to fill the request of item, in fact the number of roll cut to obtain q plus the number of roll of with p+q cut to obtain p has to be more or equal the number of items q requested and the numbers of rolls of width q cut. In this constraint $N_q = 0$ if $q \in D$ else $N_q = b_i$ with $q = w_i$.

The AMPL codification of Dyckhoff model can be:

```
1. param n; # number of roll available
2. param m; # number of item needed
3. set roll := {1..n}; # different roll
4. set item := {1..m}; # item to obtain
5. param w {item} > 0; # leght of item i
6. param W {roll} > 0; # leght of roll k
7. param O {roll} default 1;
8. set S default {};
9. set D default {};
10. # if w[m] > w[1]
11. set R := {w[1]..(W[n]-w[1])};
12. # if w[1] > w[w]
13. #set R := {w[m]..(W[n]-w[m])};
14. set H := (D union R);
15. set A := (H diff S);
16. set B := (S union R);
17. set C := (D union R);
18. set I := (A diff D);
19. var x {k in roll} >= 0 integer; # utilization of roll
20. var y {p in B, q in A:q<p} >= 0 integer ; # piece of width q
    obtained from piece of width p
21. minimize width: sum {k in 1..n} W[k]*x[k];
22. subject to cap {q in S,k in roll : q=W[k] }: x[k] + sum {p in
    D:(p+q) in B} y[(p+q),p] >= sum {p in D: (p<q)} y[q,p];
23. subject to dem {q in A}: sum {p in B:p>q} y[p,q] + sum{j
    in D:(j+q) in B} y[(j+q),j] >= sum {i in D:i<q} y[q,i] +
    numberof q in ({i in item} w[i]);
24. subject to nroll {k in 1..n}: x[k] <= sum { i in roll :
    W[k]=W[i]} O[i];
25. subject to Agg {p in B, q in I:q<p}: y[p,q] = 0;
```

The model is for Cutting Stock Problem, for the bin packing it can be simplified as follow:

```
1. param n; # number of roll available
2. param m; # number of item needed
3. set roll := {1..n}; # different roll
4. set item := {1..m}; # item to obtain
5. param w {item} > 0; # leght of item i
6. param W {roll} > 0; # leght of roll k
7. param O {roll} default 1;
8. set S default {};
9. set D default {};
10. # if w[m] > w[1]
11. # set R := {w[1]..(W[n]-w[1])};
12. # if w[1] > w[m]
13. set R := {w[m]..(W[n]-w[m])};
14. set H := (D union R);
15. set A := (H diff S);
16. set B := (S union R);
17. set C := (D union R);
18. set I := (A diff D);
19. var x >= 0 integer; # utilization of roll
20. var y {p in B, q in A:q<p} >= 0 integer ; # piece of width q
    obtained from piece of width p
21. minimize width: W[1]*x;
22. subject to cap {q in S : q=W[1] }: x + sum {p in D:(p+q)
    in B} y[p+q,p] >= sum {p in D: (p<q)} y[q,p];
23. subject to dem {q in A}: sum {p in B:p>q} y[p,q] + sum{j
    in D:(j+q) in B} y[j+q,j] >= sum i in D:i<q y[q,i] +
    numberof q in ({i in item} w[i]);
24. subject to nroll : x <= n;
25. subject to Agg {p in B, q in I:q<p}: y[p,q] = 0;
```

2.3 Arc Flow Formulation

In 1999 Valério de Carvalho presented a different formulation of Cutting stock problem using a graph representation of an arc flow for a better comprehension of branch-and-price algorithm (see 2.36).
Given a different number of bins grouped in K classes with parameter W_k for the different capacities and B_k for the number of bin in each class, in symmetry with a set of item grouped in m classes with parameter w_i for capacities and b_i for the number of item in each class and both classes (bin and item) are indexed in order of decreasing value of capacity and size obtaining $W_{max} = max_k W_k = W_1$.
Consider a graph G=(V,A) formulate with vertices V={0,1,...,W_{max}} and arcs A={(d,e): $0 \leq d < e \leq W_{max}$ & $e - d = w_i$ $\forall (1 \leq i \leq m)$} with that representation in the graph exits a directed arc if there is an item of the size e-d and a directed arc between d and d+1 corresponding to unoccupied portion of the bin, concluding the number of arcs is O(mW_{max}).
The activation of a bin with capacity w_k is when exist a path between vertices 0 and w_k the length of arc represent the size of item packed.
the number of directed arc from w_k to 0 are the bin k used, the number of constraints in the model is O($mW_{max} + m + K$).
With the previous defined variable the model has a lot of symmetry in the optimal solution because there are many alternative solutions with same item in each bin, to reduce the symmetry in the solution space and the size of the model reducing number of arc in A, with the sort of the item in decreasing value, the following criteria could be used for the reduction:

Criterion 1.
Consider two arc and two item with length w_{i1} and w_{i2} with $w_{i1} \geq w_{i2}$, an arc designated by (d,d+w_{i2}) should have its tail at a node d that is node 0 or is the head of another arc (d−w_{i1},d), in particular,if a bin has any loss it must be last in the bin.

Criterion 2.
The variable corresponding to the arc, called $x_{d,e}$ fo arc (d,e), could be set to zero for d < w_n if is a loss arc variable $x_{d,d+1}$.

Criterion 3.
Considering item i_2 with w_{i2} the only valid arc for size w_{i2} could star from $d = 0$, another arc of size w_{i1} or an arc with size sw_{i2} with $s = 0, 1, ..., b_{i2} - 1$ and $d + sw_{i2} \leq W_{max}$ where b_{i2} is the demand of item i_2

After applying the previous criteria, the remaining set of arcs can be denoted with A'$\subset$ A.

For the mathematical formulation using the above consideration, adding the decision variable x_{de} as the arc and the item of size e − d, placed on distance d from the beginning of the bin and z_k the number of bin of size k used, seen as a feedback arc $x_{(Wk,0)}$.
Minimizing the weight flow, the model can be formulated as follows:

$$min \quad \sum_{k=1}^{K} W_k * z_k \tag{2.12}$$

$$s.t. \quad -\sum_{(d,e)\in A'} x_{de} + \sum_{(e,f)\in A'} x_{ef} = \begin{cases} \sum_{k=1}^{K} z_k & if \;\; e = 0, \\ -z_k & for \;\; e = W_k, \;\; k = 1, ..., K, \\ 0 & otherwise, \end{cases} \tag{2.13}$$

$$\sum_{(d,d+w_i)\in A'} x_{d,d+w_i} \geq b_i, \quad i = 1, ..., m, \tag{2.14}$$

$$z_k \leq B_k, \quad k = 1, ..., K, \tag{2.15}$$

$$x_{de} \geq 0 \;\; integer \;\; \forall (d, e) \in A' \tag{2.16}$$

$$z_k \geq 0 \;\; integer \;\; k = 1, ..., K. \tag{2.17}$$

The flow conservation constraint (13) guarantees that the items are placed at the border of after another item.
(14) is a demand constraint, it forces the model to satisfy the request of item, while constraint (15) reduces the amount of feedback arc to not exceed the number of bin of a certain length.
It is possible to understand from the flow decomposition properties that the model has finite set of cycle and paths and every cycle corresponds to a bin and the length of the arcs in the bin are the different item packed.

The AMPL codification of the problem could be.

```
1. param n;
2. param m;
3. set roll := 1..n; # different roll
4. set item := 1..m; # item to obtain
5. param w {item} > 0; # leght of item i
6. param W {roll} > 0; # leght of roll k
7. param O {roll} default 1;
8. set A := {0..(W[n])};
9. set B := {1..(W[n]-1)};
10. var x {i in A,k in A,i<k} >= 0 integer;
11. var y >= 0 integer;
12. minimize numb : y ;
13. subject to me0 : - sum {d in A} x[d,0] + sum {f in A:(0<f)}
    x[0,f] = y;
14. subject to mew : - sum {d in A:(d < W[n])} x[d,W[n]] +
    sum {f in A:(W[n] < f)} x[W[n],f] = - y;
15. subject to mew2 {e in B } :- sum {d in A:(d < e)} x[d,e] +
    sum {f in A:(e < f)} x[e,f] = 0;
16. subject to qb {i in item}: sum {d in A:(d+w[i]) in A}
    x[d,d+w[i]] >= numberof w[i] in ({q in item} w[q]);
17. subject to nroll : y <= n ;
```

2.4 Column generation on CSP and BPP

2.4.1 Dantzig Wolfe decomposition on Kantorovich formulation

The Dantzing-Wolfe decomposition is a tool used to obtain stronger (LP) relaxation from the reformulation of model for integer and combinatorial optimization. Normally the LP relaxation of a model resulting from the drop of the integrality constraints is very weak, a stronger model is obtained by restricting the set of point.
From a compact formulation.

$$z* = min\ c^T x \tag{2.18}$$
$$s.t.\ Ax \geq b \tag{2.19}$$
$$Dx \geq f \tag{2.20}$$

with:
$A(m_1\ x\ n)$
$D(m_2\ x\ n)$
and
$Q(D, f) = x \in R^n | Dx \geq f$
$Q(D, f)$ is a non-empty polytope

Theorem 3 (Weyl-Minkowski, 1936)**.** *:*
Let P be a non-empty polyhedron with at least one extreme point.
Then
$P = conv(ext(P)) + rec(P)$
A point $\boldsymbol{x}$ belongs to $Q(D, f)$ if and only if it can be written as a convex combination of the extreme points $\{v_1, ..., v_q\}$ of $Q(D, f)$:
$Q(D, f) \ni x = \sum_i \lambda_i V_i \quad$ *with* $\quad \sum_i \lambda_i = 1$ *and* $\lambda_i \geq 0$

Using the theorem and substituting for $x = \sum_i \lambda_i V_i$ in the compact formulation it could be obtained the *extensive formulation*:

$$z*_{DW} = \ min \ \sum_{i=1}^{q} (c^T v_i)\lambda_i \tag{2.21}$$

$$s.t. \sum_{i=1}^{q} (Av_i)\lambda_i \geq b \tag{2.22}$$

$$\sum_{i=1}^{q} \lambda_i = 1 \tag{2.23}$$

$$\lambda_i \geq 0 \quad 1 \leq i \leq q \tag{2.24}$$

Compact and *Extensive* formulation have the same optimal value $z*_{DW} = z*$ and the extensive formulation has fewer constraints $(m_1 + 1)$ but a huge number of variable $(\binom{n}{m_2})$.

Adding to the *compact* formulation (2.18) the constaint $x \in Z^n$ and the new polytope is $X = Q(D, f) \cap Z^n$.
the space $X = Q(D, f) \cap Z^n = conv(X) \cap Z^n$ result in a *Convexification* of the area and the model can be written as:

$$z*_c = min \ c^T x \tag{2.25}$$

$$s.t. \ Ax \geq b \tag{2.26}$$

$$x \in conv(X) \tag{2.27}$$

$$x \in Z^n \tag{2.28}$$

Probably the space $conv(X) \subset Q(D, f)$ resulting in a continuous relaxation stronger than the one from the original model.
Defined another theorem:

Theorem 4 (Nemhauser-Wolsey, 1988)**.** *:*
X is generated by a finite number of integer points $\{p_1, ..., p_q\}$ *of* X
$X \ni x = \sum_i \lambda_i \boldsymbol{p}_i \quad$ *with* $\quad \sum_i \lambda_i = 1$ *and* $\lambda_i \in \{0, 1\}$

It could be defined another reformulation that consists in the Discretization of the polytope, the difference between the two method is:
Convexification: it consider the integer vertex of the area defined as $\mathbf{v}$,
$X \ni x = \sum_i \lambda_i \mathbf{v}_i \quad$ with $\quad \sum_i \lambda_i = 1$ *and* $\lambda_i \in$ **[0,1]**

the branching must be performed on the original variable **x**
Discretization: it considers the integer inner point of the area defined as **p**,
$X \ni x = \sum_i \lambda_i \mathbf{p}_i$ with $\sum_i \lambda_i = 1$ *and* $\lambda_i \in$ **{0,1}**
the branching can be performed on the binary variable λ.

In the end the two methods are equivalent if the original variable are binaries and the set of integer point of $Conv(X)$ corresponds to the set of extreme points.

The discretization is used on the Kantorovich model to obtain the Vance formulation where Gilmore Gomory introduced the column generation, the passage are the following:
from the Kantorovich model (4.13), Each of the m polyhedra corresponding to the pattern feasibility constraints is discretized:

$X^j = \{p \in N^n | \sum_{i=1}^n l_i p_i \leq L\} \quad j = 1, ..., m$

and the pattern x from the discretization became:

$$x_{ij} = \sum_{k=1}^{q_j} p_k^j \lambda_k^j \tag{2.29}$$

$$\sum_{k=1}^{q_j} \lambda_k^j = 1 \tag{2.30}$$

$$\lambda_k^j \in \{0, 1\} \tag{2.31}$$

where $q_j = |X^j|$

The discretized complete model is:

$$z_D^* = min \sum_{j=1}^{m} \sum_{k=1}^{q_j} \lambda_k^j \tag{2.32}$$

$$\sum_{j=1}^{m} \sum_{k=1}^{q_j} \lambda_k^j (p_k^j)_i \geq d_i \quad 1 \leq i \leq n \tag{2.33}$$

$$\sum_{k=1}^{q_j} \lambda_k^j = 1 \quad 1 \leq j \leq m \tag{2.34}$$

$$\lambda_k^j \in \{0, 1\} \tag{2.35}$$

Considering that $X^1 =, ..., = X^m = X$
hence $p_k^1 =, ..., = p_k^m = p_k$ and $q_1 =, ..., = q_m = q$
$\sum_{k=1}^{q_j} \lambda_k^j$ could be replaced with $\lambda_k \geq 0, integer$ and
$\sum_{k=1}^{q_j} \lambda_k^j = 1 \quad 1 \leq j \leq m$ correspond to the sum $\sum_{j=1}^{m} \sum_{k=1}^{q_j} \lambda_k^j = m$
applying those replacement is obtained the Gilmore Gomory model:

$$z_D^* = min \sum_{k=1}^{q} \lambda_k \tag{2.36}$$

$$\sum_{k=1}^{q} \lambda_k (p_k)_i \geq d_i \quad 1 \leq i \leq n \tag{2.37}$$

$$\sum_{k=1}^{q} \lambda_k = m \tag{2.38}$$

$$\lambda_k \geq 0, integer \quad 1 \leq k \leq q \tag{2.39}$$

2.4.2 Column generation on Gilmore Gomory

The enumeration of all pattern of all combination of items that can fit into a bin can be described with a set covering formulation, using p to define both pattern and index, and the set of patterns and the set of indices with P.
For the cutting stock problem, an integer array $(a_{1p}, a_{2p}, ..., a_{mp})$ could describe a pattern p, the element a_{jp} is the number of copies of item j contained in pattern p, $a_{jp} \geq 0$ is integer and $\sum_{j=1}^{m} a_{jp} w_j \leq c$.
With an integer variable y_p that gives, for each p $\in$ P, the number of time the pattern p is used, we can describe the CSP in a set covering formulation:

$$min \sum_{p \in P} y_p \tag{2.40}$$

$$s.t. \sum_{p \in P} a_{jp} y_p \geq d_j \quad \forall j \tag{2.41}$$

$$y_p \geq 0, integer \tag{2.42}$$

Similar at the last formulation obtained at the previous section (2.36)
From the model it could be seen that the first constraint fulfills the demand requests and makes the subset of pattern contain at least d_j copies of item j.
For the BPP the problem can be simplified as following, the element of the array a_{jp} is 1 if contained in pattern p, $a_{jp} \geq 0$ is binary, $y_p \in \{0, 1\}$ and $\sum_{j=1}^{m} a_{jp} y_p \geq 1$.
the number of the patterns is exponential to the numbers of items so it could create an enumeration of all pattern returning to an increment of the solving time, in this case column generation techniques are adopted.
For the CSP basic technique is the definition of a continuous relaxation by removing the integrality constraint and initialize it with a feasible solution provided by the reduced set of patterns P' $\subseteq$ P.
The resulting problem called Resticted Master Problem (RMP) is:

$$min \quad \sum_{p \in P'} y_p \tag{2.43}$$

$$s.t. \quad \sum_{p \in P'} a_{jp} y_p \geq d_j \quad (j = 1, ..., m) \tag{2.44}$$

$$y_p \geq 0 \quad (p \in P') \tag{2.45}$$

Solved the RMP, let's consider π_j the dual variable of the jth constraint, it could be find a reduction of the objective function value for a column p $\notin$ P' solving the *reduced costs* $\overline{c}_p = 1 - \sum_{j=1}^{m} a_{jp} \pi_j \quad (p \notin P')$.
Solving an *unbonded knapsack problem*, in witch the dual variable π_j is the profits, could determine the column with the most negative reduced cost.
the problem is the following:

$$max \quad \sum_{j=1}^{m} \pi_j v_j \tag{2.46}$$

$$s.t. \quad \sum_{j=1}^{m} w_j v_j \leq c \tag{2.47}$$

$$v_j \geq 0 \;\; integer \;\; (j = 1, ..., m) \tag{2.48}$$

If the solution has value greater than 1, the column can be added to the RMP. Iterating the process provides the optimal solution of the continuous relaxation of the set covering formulation.
A pattern could contain more than d_j copies of an item j, imposing the constraint $a_{jp} \leq d_j \;\; \forall j$ it could be obtained a formulation that consists of a *bounded knapsack* defined by (2.46) and $v_j \leq d_j$, the result provided by the model is a stronger lower bound since the numbers of feasible pattern becomes smaller.
Set Partitioning can be a different formulation for an alternative column generation, from the definition of partitioning the sign'=' replace '$\geq$'.
To accelerate the column generation Valèrio de Carvalho proposed dual cuts, adding in the RMP some "extra" columns finds in a fast way accelerating the convergence to the continuous optimal solution reducing the number of standard columns generated by the *slave problem*.

2.4.3 Branch(and cut) and price

At the end of the column generation method the solution obtained can be fractional, the last step is the searching for a feasible integer solution, the direct solution of the master problem with the generation of all possible patterns could be an option only for small instances, other, non exact, method use rounding heuristics but the efficiency depends on the instances.
A better approach could be the using of lower bound from the column generation L_{LP} into the enumeration tree obtaining a branch and price algorithm. The problem of this approach is that the decision taken during the enumeration should be collected to exclude some redundances and avoid generation of columns, already excluded in branch decision, in the master and slave problem.
In the literature were proposed different methods, in this section are reviewed

the principal.
Vance,Barnhart, Johnson and Nemhauser proposed the first B&P for BPP in 1994, and it works at every decision node, where the algorithm considers the bins in which the decision variable y_p is fractional and select the items fractionally packed with largest total weight.
In 1995 Scheithauer and Terno works on the CSP with an hybrid strategy: they first reduce the instance solving the continuous relaxation rounded down obtaining a partial integer and residual instance and the last are attacked by heuristic algorithm.
Years later Vance gets the set covering model applying the Dantzing-wolfe decomposition, discussed in previous section (2.4.1), implementing tailored brancing rules.
In 1999 Valério de Carvalho proposed a different column generation approach based on the arc flow formulation (2.3), the slave in this specifically case is a longest path in *acyclic directed graph*.
In the same year Vanderbeck introduces an algorythm that branch in a set of columns, thus adding a constraint in the master problem imposing that the sum $\mathbf{s}$ of the variable associated with such set are either not smaller than $\lceil s \rceil$ or not grater than $\lfloor s \rfloor$. The convergence of the algorithm is improved by cut generation at the decision node, the method is called *Branch-and-cut-and-price* algorithm.
Other study are applied after the develop of the algorithm, about 600 papers between 1995 and 2004, at the state of the art the most efficient algorithm produced is from Scheithauer, Terno, Muller and Belov that solve CSP with the use of cutting plane generation.

Chapter 3

Experimental evaluation on exact model

3.1 Heuristic Upper Bound or Primal Bound

To improve the solution of the problem can be used some bound, the upper or primal bound reduce the number of roll used and applied on data reduce the number of variable generated by Kantorovich model, the simplest algorithm that can be applied is The First-Fit algorithm, for each item of the problem it attempts to place the object in the first bin that can accommodate it. It was established that the number of bin find with the algorithm is no more than twice of the optimal number. The *Firs-Fit Decreasing* is an improvement of the *Firs-Fit* just with the introduction of a decreasing order in the width of item.
A better upper bound than the previously found is provided by the Modified first-fit decreasing, the first operation done by the algorithm is classifying the item in 4 set : Large (w $>$ 1/2 W), Medium (1/2 W $>$ w $>$ 1/3 W), Small (1/3 W $>$ w $>$1/6 W), Tiny (1/6 W $>$ w). After the classification the algorithm proceed as it follow:

1. Is used 1 bin for each large item.
2. For every bin used, in forward direction, if the smallest medium item fit, place the largest medium that fit otherwise skip the bin.
3. For every bin used with no medium item inside, in backward direction, if the two smallest remaining small item fit, place the smallest and the largest small item that fit otherwise skip the bin.

4. For every bin used, in forward direction, if the smallest item fit, place the largest item that fit and stay otherwise skip the bin.

5. Use the FFD to pack the remaining item.

It was established that the number of bin find with the MFFD algorithm is: MFFD $>$ 71/60 OBJ+1

3.2 Heuristic Lower Bound or Dual Bound

The evaluation of an heuristic lower bound is provided from the study of Martello and Toth they provide a lower bound for the problem and it is proved that the worst case performance ration of the bound obtained is $R(LBmt) = 2/3$ from the optima solution and can be obtained in a polynomial amount of time O(n).
follow the calculation of a weaker bound(LBw) and the Martello and Toth bound (LBmt):

Weak lower bound:

Considered a positive integer C bigger than any item size $s_i \leq C \ \ \forall i = 1..n$ the first bound is:

$$LBw = \left\lceil \sum_{i=1}^{n} s_i/C \right\rceil \tag{3.1}$$

This bound is proved to have a ratio of $R(LBw) = 1/2$

The Martello and Torh Bound is obtained as follow:

Considered an integer K, $0 \leq K \leq 1/2C$ and defined 3 set of item where:

$$N1 = i \in I : s_i > C - K$$
$$N2 = i \in I : C - K \geq s_i > 1/2C$$
$$N3 = i \in I : 1/2C \geq s_i \geq K$$

then

$$LBmt = |N1| + |N2| + max\left(0, \left\lceil \frac{\sum_{i \in N3} s_i - (|N2|C - \sum_{i \in N2} s_i)}{C} \right\rceil\right) \tag{3.2}$$

that is a valid lower bound for the problem.

3.3 Code in C++ for the bounds

The following code was written using the C++ language, the FFD method and the lower bound alghorithm of the previous section:

```
#include <iostream>
#include <fstream>
#include <time.h>
#include <stdlib.h>
#include "complex.h"
#include <string>
#include <math.h>
using namespace std;

 int leggi_file(double data[], int max)
{
    int num = 0;
    ifstream f("File-istance.dat");

    while(num < max)
    {
        f >> data[num];
        if(f.eof())
            break;
        if(data[num]==0)
          break;
        num++;
    }
```

```
    return num;
}

void binPacking(int *a, int size, int n)
{
    int binCount = 0;
    int binValues[n];
    for (int i = 0; i < n; i++)
        binValues[i] = size;

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
        {
            if ( binValues[j] - a[i] >= 0)
            {
                binValues[j] -= a[i];
                break;
            }
        }

    for (int i = 0; i < n; i++)
        if (binValues[i] != size)
            binCount++;

    cout << "Primary bound: "
            << binCount;
}

int* sort(int *sequence, int n)
{
    // Bubble Sort descending order
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n - 1; j++)
            if (sequence[j] < sequence[j + 1])
            {
                sequence[j] = sequence[j] + sequence[j + 1];
                sequence[j + 1] = sequence[j] - sequence[j + 1];
                sequence[j] = sequence[j] - sequence[j + 1];
            }
    return sequence;
}
```

```
int main(int argc, char **argv)
{

    double data[100000];
    int num = leggi_file(data, 100000);

        int n;
        int m;
        int a[100000];
        double w[100000];
        int size;
        int c = 3;
        int W;
        int sum1 = 0;

        m = data[0];
        n = data[0];
        size = data[3 + 2*n];
        W = data[3 + 2*n];
        for(int i = 0; i < n; i++ )
        {
            a[i] = data[c];
            w[i] = data[c];
            c = c + 2;
            sum1 = sum1 + w[i];
    }

    float N1 = 0;
    float LB = 0;
    float L2 = 0;
    float in3 = 0;
    float in2 = 0;
    float sumN2 = 0;
    float sumN3 = 0;
    float LB1 = 0;
        LB1 = ceil(sum1/W);
        cout << "dual bound weak: " << LB1 << endl;
    for ( int K = (W/2); K > 0; K--)
    {
         N1 = 0;
```

```
 L2 = 0;
 in3 = 0;
 in2 = 0;
 sumN2 = 0;
 sumN3 = 0;
for (int i1 = 0; i1 < m + 1; i1++)
{
          if(w[i1] > (W-K))
              {
                   N1 = N1 + 1;
          }
                   else
                   {
                   if(w[i1] > (W/2))
                   {
                            in2 = in2 + 1;
                            sumN2=sumN2 + w[i1];
                   }
                   else
                   {
                   if(w[i1]>= K)
          {
                            in3 = in3 + 1;
                            sumN3=sumN3 + w[i1];
                             }
                        }
                   }

}
if((sumN3 - in2*W + sumN2)/W > 0)


          L2=N1+in2+ceil((sumN3 - in2*W  + sumN2)/W);

          else

          L2=N1+in2;

          if(L2 > LB)
          LB = L2;
```

```
        }
    cout << "Dual Bound: " << LB << endl;

    int *sequence = sort(a, n);
    binPacking(sequence, size, n);
}
```

3.4 Solution of benchmark instances

For a first comparison of the exact model could be used the number of variable and constraints generated and the time to obtain the solution of a specific set of object.
Following some solution about different instances with increasing number of item for 1-D bin packing problem.

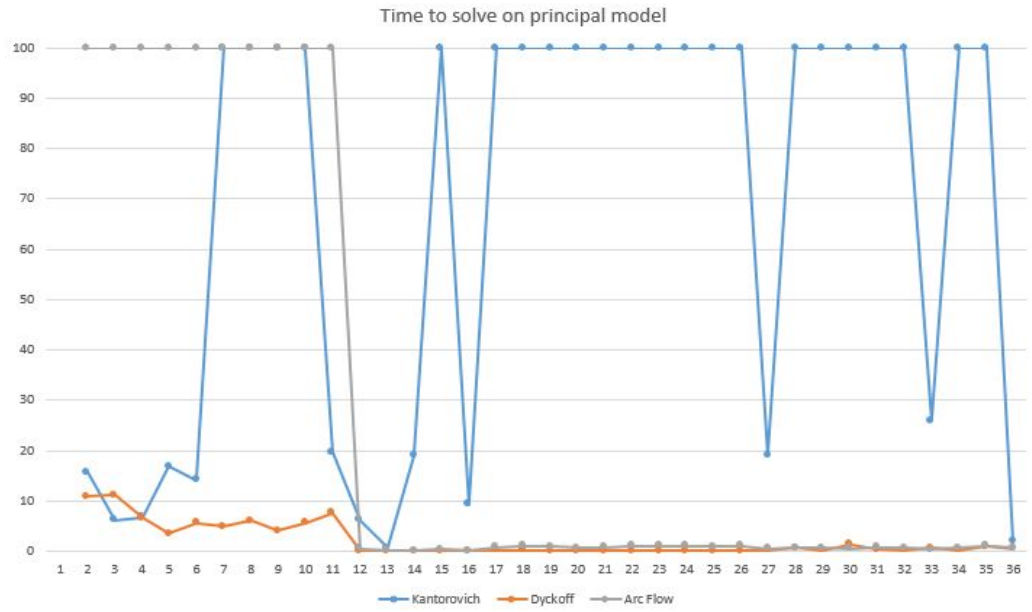

Figure 3.1: Different time to solve on different model

instances	problem charatestic			time to solve to optimality		
	N° item	Wmax bin	optimal	**Kantorovich**	**Dyckoff**	**Arc Flow**
bpp001	100	1000	18	15,64	10,84	>100
bpp002	100	1000	18	6,24	11,12	>100
bpp003	100	1000	18	6,81	6,64	>100
bpp004	100	1000	18	16,77	3,50	>100
bpp005	100	1000	18	14,13	5,60	>100
bpp006	100	1000	18	>100	4,90	>100
bpp007	100	1000	18	>100	5,98	>100
bpp008	100	1000	18	>100	3,99	>100
bpp009	100	1000	18	>100	5,59	>100
bpp010	100	1000	18	19,70	7,54	>100
N1C1W1C	50	100	20	6,20	0,11	0,48
N1C1W1E	50	100	26	0,62	0,07	0,20
N1C1W1H	50	100	31	18,94	0,04	0,07
N1C1W1I	50	100	25	>100	0,11	0,30
N1C1W1J	50	100	26	9,27	0,09	0,12
N3C2W4_A	200	120	113	>100	0,15	0,78
N3C2W4_B	200	120	112	>100	0,19	1,02
N3C2W4_C	200	120	132	>100	0,11	0,99
N2C2W4_D	200	120	114	>100	0,12	0,70
N2C2W4_E	200	120	110	>100	0,12	0,80
N2C2W4_F	200	120	115	>100	0,12	1,07
N2C2W4_G	200	120	122	>100	0,11	1,06
N2C2W4_H	200	120	113	>100	0,14	1,09
N2C2W4_I	200	120	115	>100	0,07	0,91
N2C2W4_J	200	120	120	>100	0,13	1,14
N3C3W1_A	200	150	66	19,15	0,24	0,49
N3C3W1_B	200	150	71	>100	0,66	0,67
N3C3W1_C	200	150	69	>100	0,20	0,66
N3C3W1_D	200	150	63	>100	1,39	0,57
N3C3W1_E	200	150	68	>100	0,36	0,84
N3C3W1_F	200	150	69	>100	0,18	0,66
N3C3W1_G	200	150	65	25,86	0,67	0,42
N3C3W1_H	200	150	69	>100	0,15	0,69
N3C3W1_I	200	150	68	>100	0,95	1,08
N3C3W1_J	200	150	65	1,92	0,62	0,56
N° solved to opt			**35**	13	35	25

Figure 3.2: Time to solve confrontation

3.4.1 Analysis of the results obtained

The first analysis was about the number of variable and constraints generated by cplex. In Kantorovich model the N° of variable is exactly (n*m+m) where n is the number of item and m is the number of roll available while the constraints are (n+m), so using an heuristic for the reduction of roll available could decrease the structure of the model. In Dyckoff model the variable and the constraints are related to the maximum roll width, they are respectively O(m*Wmax) and O(K + Wmax),as a mix of the previous two models the arc flow generate a fixed number of constraint and variable that depends, like Dyckoff model, on the Wmax of the bin, the number of variable are ((Wmax^2+Wmax)/2) and the constraints (n+Wmax), whereas the number of item and the number of roll are the same on problem it can be obtained the following graph:

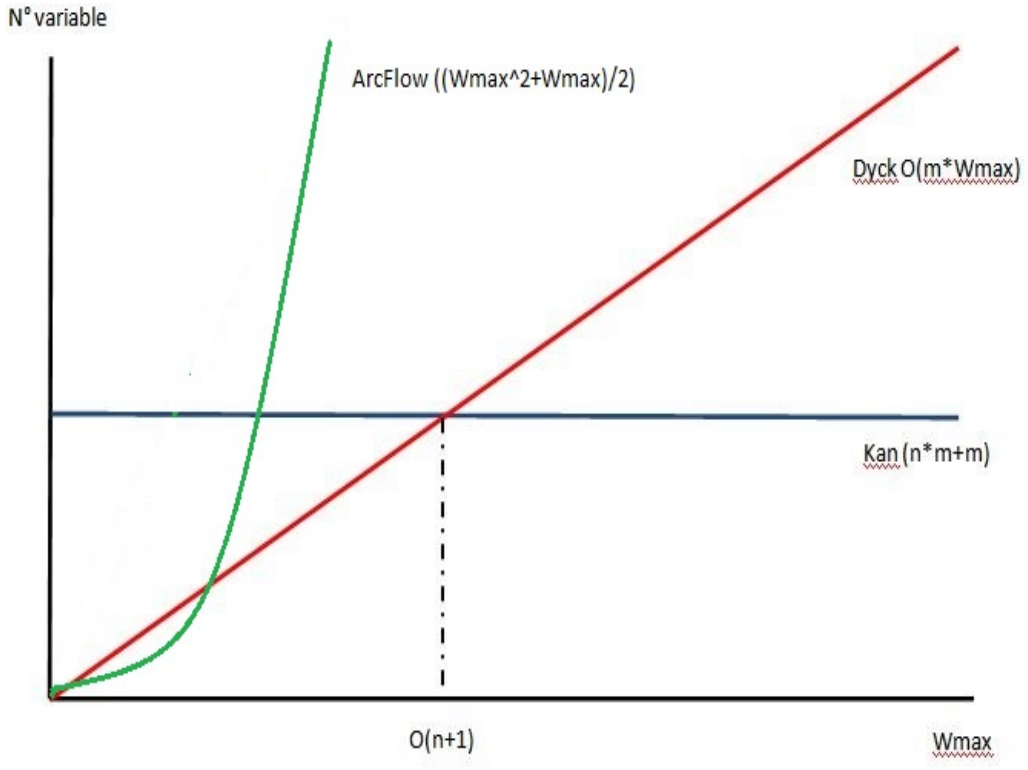

Figure 3.3: Graph N°variable - Wmax

To find the intersection point with a fixed m and n,
$n * m + m = m * Wmax => (n * m + m)/m = Wmax => n + 1 = Wmax$
In fact with 500 items and 150 Wmax Nv(Dyck) < Nv(Kant).
with 200 items and 1000 Wmax Nv(Dyck) > Nv(kant).
At the same situation is the growth of the variable in the Arc Flow model, it only depends on the parameter Wmax but it is exponential on the value so the break even with the kantorovich number of variable is with a lower value than the Dyckoff one.
Another important analysis was about the solutions found, The Kantorovich model from a small number of item have difficult to find the solution because start to explore the tree and the structure have a large number of symmetries, for Dyckoff model a first consideration is that the difficult of solve a set of instances is related at the number of different item and the maximum width of the roll, problem with a small number of different item and a small maximum weight is simpler to solve than a problem with every item different and bigger maximum weight, the number of item doesn't seems have an impact in the solution.
At the same level the Arc flow Model have huge resolution problem for instances with an hight Wmax, and the explanation is on the exponentially number of variable depending on the weight of bill factor but until a certain level where the solver start to have difficulty to compute the variable it takes the same amount of time of the Dyckoff model even the differences on the variable number.
In some case the Dyckoff model provides an optimum in short while, but in another case for the same number of item and Wmax it starts to enumerate all the node of branch and bound tree, that could depends on the small number of possible solution compared with the high number of variable.

3.4.2 Dyckoff Hard istances

For the analysis of the Hard instances of the Dyckoff model were plotted the histograms and were calculated averages and standard deviations.
The first comparison is between the instance "Nirup04" and "Nirup05, the first is Hard to solve.

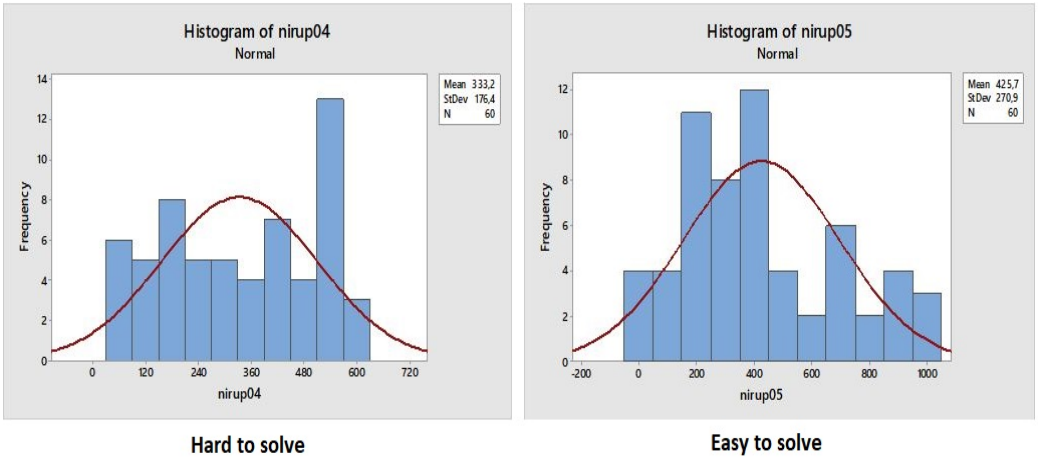

Figure 3.4: Different distribution nirup04-nirup05

Following other graphics for different instances:

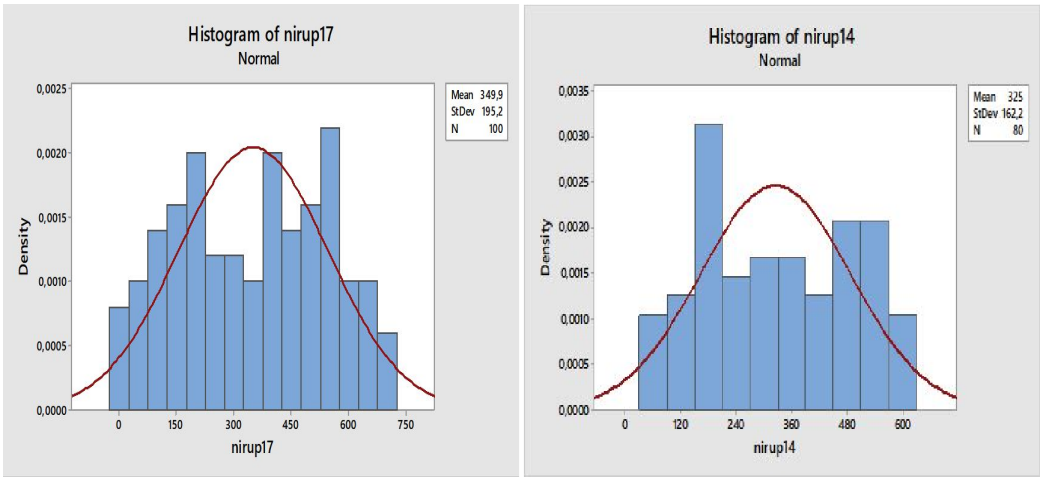

Figure 3.5: Hard to solve.

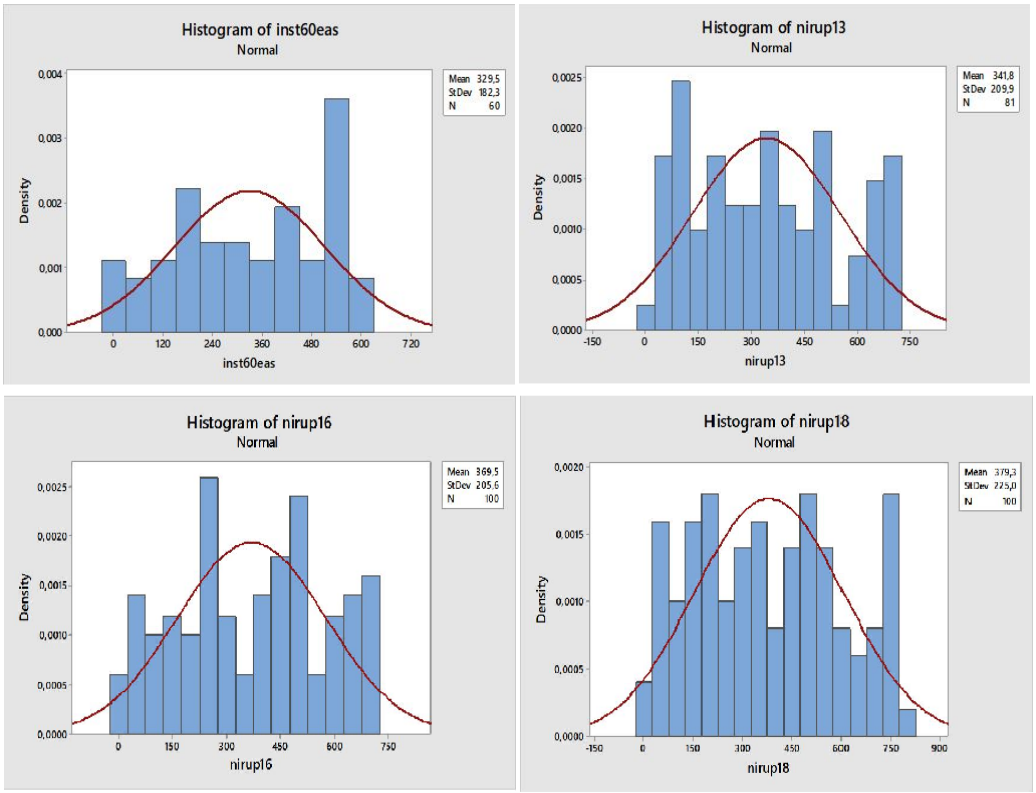

Figure 3.6: Easy to solve.

The difference of distribution didn't give evidence of difference between easy and hard instances and this was confirmed by a second analysis about problem with the same number of item (200):

N°Item	Dev	Avg	Diff
Hard 20	198,6	365	Easy
Hard 23	198,8	374,9	Medium
Hard 21	200	360	Hard
Hard 19	200,5	379,9	Easy
Hard 22	213,5	355	Hard
Hard 24	224,5	414,6	Hard
Hard 25	224,8	399,6	Easy
Hard 27	225,9	414,2	Easy
Hard 26	229,5	399,8	Medium
Hard 20	238,3	404,1	Easy

Chapter 4

From integer to continuous model

For the evaluation of the efficiency of continuous model, the first step is the definition of the strategies to obtain the model:
The easiest way to have the conversion is start from a binary model; since the Kantorovich model start from binary variable the method is applied only on Dyckhoff and Arc Flow model.

4.1 binarization of the models

The challenge in this part of the study is to not change the model in the structure and have the same optimal results of the mixed integer model.
The rule applied on the two models considered is the same for both but have different meaning in term of the structure obtained:

In the Dyckhoff model the binarization is achieved adding a parameter to the variable representing the piece of width q obtained from piece of width p, becoming $y_{q,p,k}$ meaning that piece is obtained from roll k, since it's impossible to cut 2 piece of the same length in a roll the the variable could be used as binary.
The model is the following:

$$min \quad \sum_{k=1}^{K} W_k * z_k \tag{4.1}$$

$$s.t. \quad z_k + \sum_{p\in D,(p+q)\subset(S\cup R)} y_{(p+q,p,k)} \geq \sum_{p\in D,p<q} y_{q,p,k} \quad \forall q \in S \quad \forall k \tag{4.2}$$

$$\sum_{k=1}^{K} \sum_{p\in(S\cup R),p>q} y_{p,q,k} + \sum_{k=1}^{K} \sum_{j\in D,(j+q)\in(S\cup R)} y_{j+q,j,k} \quad \geq$$

$$\sum_{k=1}^{K} \sum_{i\in D,i<q} y_{q,i,k} + Nq \quad \forall q\in(D\cup R)-S \tag{4.3}$$

$$z_k \geq z_{k+1}, \quad k := 1..n-1 \tag{4.4}$$

$$y_{p,q,k} \geq 0, binary \quad \forall(p \in (S \cup R), q \in D, q < p) \quad \forall k \tag{4.5}$$

$$z_k \geq 0, binary \quad \forall k \tag{4.6}$$

The constraint (4.4) is added for the symmetry that can be generated from the model choosing witch bin should be activated.
Displaing the variable y at the resolution of the model is easy to see the structure of the solution and the cut pattern to use to achieve the optimal result: the variable is composed by a k number of matrix and every matrix have the composition of a cutting pattern.
In a specific case with size of bin$W = 20$,size of generic item
$i_x = 8, i_{x+1} = 7, i_{x+2} = 5$
there will be a matrix with parameter k=a with the variable
$x_{8,20,a} = 1, x_{7,12,a} = 1, x_{5,5,a} = 1$ and 0 in the others.

In a similar way the Arc flow binary model is obtained adding a parameter k to the variable x meaning that every bin represent a flow from 0 to the weight of the bin, from the structure and the flow model it's impossible to violate the flow and with a parameter to represent every flow it could be generated the value of variable must be binary.
The model is the following:

$$min \quad \sum_{k=1}^{K} W_k * z_k \tag{4.7}$$

$$s.t. \quad -\sum_{k=1}^{K} \sum_{(d,e)\in A'} x_{d,e,k} + \sum_{k=1}^{K} \sum_{(e,f)\in A'} x_{e,f,k} = \begin{cases} \sum_{k=1}^{K} z_k & if \;\; e = 0, \\ \sum_{k=1}^{K} -z_k & if \;\; e = W_k, \\ 0 & otherwise, \end{cases} \tag{4.8}$$

$$\sum_{k=1}^{K} \sum_{(d,d+w_i)\in A'} x_{d,d+w_i,k} \geq b_i, \quad i = 1, ..., m, \tag{4.9}$$

$$z_k \leq n, \quad k = 1, ..., K, \tag{4.10}$$

$$x_{d,e,k} \geq 0 \;\; binary \;\; \forall (d,e) \in A' \;\; \forall k \tag{4.11}$$

$$z_k \geq 0 \;\; binary \;\; k = 1, ..., K. \tag{4.12}$$

Displaing the variable x at the resolution of the model is easy to see the structure of the solution and the flow used to achieve the optimal result: the variable is composed by a k number of matrix and every matrix represent a flow from 0 to the length of the bin.
In a specific case with size of bin$W = 20$,size of generic item
$i_x = 8, i_{x+1} = 7, i_{x+2} = 5$
there will be a matrix with parameter k=a with the variable
$x_{0,8,a} = 1, x_{8,15,a} = 1, x_{15,20,a} = 1$ and 0 in the others.

4.1.1 experimental on binary models

For the passage to the non linear model is important to check if the binary model work and if it give an optimal solution even if the number of variable increase in an exponential way.
Follow computational experiment on the model:

instances	problem char		Dyckoff			Bynary Formulation		
	N° item	Wmax bin	Time to solve (s)	Var	Con	Time to solve (s)	Var	Con
bpp001	100	1000	10,84	24085	549	5,54	1107910	639
bpp002	100	1000	11,12	24625	550	10,76	1157375	642
bpp003	100	1000	6,64	22393	547	8,13	1011632	633
bpp004	100	1000	3,50	21851	545	7,19	917742	627
bpp005	100	1000	5,60	22900	547	9,63	1007600	633
bpp006	100	1000	4,90	25230	551	15,70	1211040	645
bpp007	100	1000	5,98	22331	546	7,05	960233	630
bpp008	100	1000	3,99	22954	547	19,64	1009376	633
bpp009	100	1000	5,59	23390	546	6,61	1052550	634
bpp010	100	1000	7,54	21188	544	5,42	868708	624
N1C1W1C	50	100	0,11	2131	94	0,89	103550	192
N1C1W1E	50	100	0,07	2030	97	0,53	101500	195
N1C1W1H	50	100	0,04	1839	100	0,60	94650	198
N1C1W1I	50	100	0,11	2134	98	0,65	106700	196
N1C1W1J	50	100	0,09	2162	100	0,58	108100	198
N3C2W4_A	200	120	0,15	2501	72	3,11	500200	470
N3C2W4_B	200	120	0,19	2246	70	2,66	449200	468
N3C2W4_C	200	120	0,11	2283	70	3,19	457800	468
N2C2W4_D	200	120	0,12	2441	72	1,89	488200	470
N2C2W4_E	200	120	0,12	2362	70	1,70	472400	468
N2C2W4_F	200	120	0,12	2420	71	1,79	484000	469
N2C2W4_G	200	120	0,11	2434	72	3,21	486800	470
N2C2W4_H	200	120	0,14	2388	71	2,29	477600	469
N2C2W4_I	200	120	0,07	2416	70	2,95	483200	468
N2C2W4_J	200	120	0,13	2199	71	1,91	439800	469
N3C3W1_A	200	150	0,24	8889	150	6,89	1777800	548
N3C3W1_B	200	150	0,66	7940	150	11,55	1588000	548
N3C3W1_C	200	150	0,20	8254	150	7,90	1650800	548
N3C3W1_D	200	150	1,39	8610	150	7,61	1722000	548
N3C3W1_E	200	150	0,36	8855	150	13,98	1771000	548
N3C3W1_F	200	150	0,18	8857	150	6,76	1771400	548
N3C3W1_G	200	150	0,67	8926	150	7,41	1785200	548
N3C3W1_H	200	150	0,15	8347	150	8,75	1669400	548
N3C3W1_I	200	150	0,95	8543	150	12,91	1708600	548
N3C3W1_J	200	150	0,62	8518	148	7,31	1703600	546
N° solved to opt			35			35		

Figure 4.1:

instances	problem char		Arc Flow			Bynary Formulation		
	N° item	Wmax bin	Time to solve (s)	Var	Con	Time to solve (s)	Var	Con
bpp001	100	1000	>100	500501	1101	>100	23023046	1101
bpp002	100	1000	>100	500501	1101	>100	23023046	1101
bpp003	100	1000	>100	500501	1101	>100	23023046	1101
bpp004	100	1000	>100	500501	1101	>100	23023046	1101
bpp005	100	1000	>100	500501	1101	>100	23023046	1101
bpp006	100	1000	>100	500501	1101	>100	23023046	1101
bpp007	100	1000	>100	500501	1101	>100	23023046	1101
bpp008	100	1000	>100	500501	1101	>100	23023046	1101
bpp009	100	1000	>100	500501	1101	>100	23023046	1101
bpp010	100	1000	>100	500501	1101	>100	23023046	1101
N1C1W1C	50	100	0,48	5051	151	0,88	252550	151
N1C1W1E	50	100	0,20	5051	151	0,87	252550	151
N1C1W1H	50	100	0,07	5051	151	0,79	252550	151
N1C1W1I	50	100	0,30	5051	151	0,66	252550	151
N1C1W1J	50	100	0,12	5051	151	0,51	252550	151
N3C2W4_A	200	120	0,78	7261	321	11,35	1452200	321
N3C2W4_B	200	120	1,02	7261	321	19,27	1452200	321
N3C2W4_C	200	120	0,99	7261	321	17,05	1452200	321
N2C2W4_D	200	120	0,70	7261	321	40,13	1452200	321
N2C2W4_E	200	120	0,80	7261	321	9,19	1452200	321
N2C2W4_F	200	120	1,07	7261	321	8,79	1452200	321
N2C2W4_G	200	120	1,06	7261	321	20,72	1452200	321
N2C2W4_H	200	120	1,09	7261	321	13,46	1452200	321
N2C2W4_I	200	120	0,91	7261	321	14,14	1452200	321
N2C2W4_J	200	120	1,14	7261	321	14,23	1452200	321
N3C3W1_A	200	150	0,49	11326	351	4,56	2265200	351
N3C3W1_B	200	150	0,67	11326	351	6,18	2265200	351
N3C3W1_C	200	150	0,66	11326	351	7,27	2265200	351
N3C3W1_D	200	150	0,57	11326	351	7,74	2265200	351
N3C3W1_E	200	150	0,84	11326	351	8,52	2265200	351
N3C3W1_F	200	150	0,66	11326	351	7,30	2265200	351
N3C3W1_G	200	150	0,42	11326	351	5,11	2265200	351
N3C3W1_H	200	150	0,69	11326	351	8,29	2265200	351
N3C3W1_I	200	150	1,08	11326	351	6,84	2265200	351
N3C3W1_J	200	150	0,56	11326	351	7,84	2265200	351
N° solved to opt			25			25		

Figure 4.2:

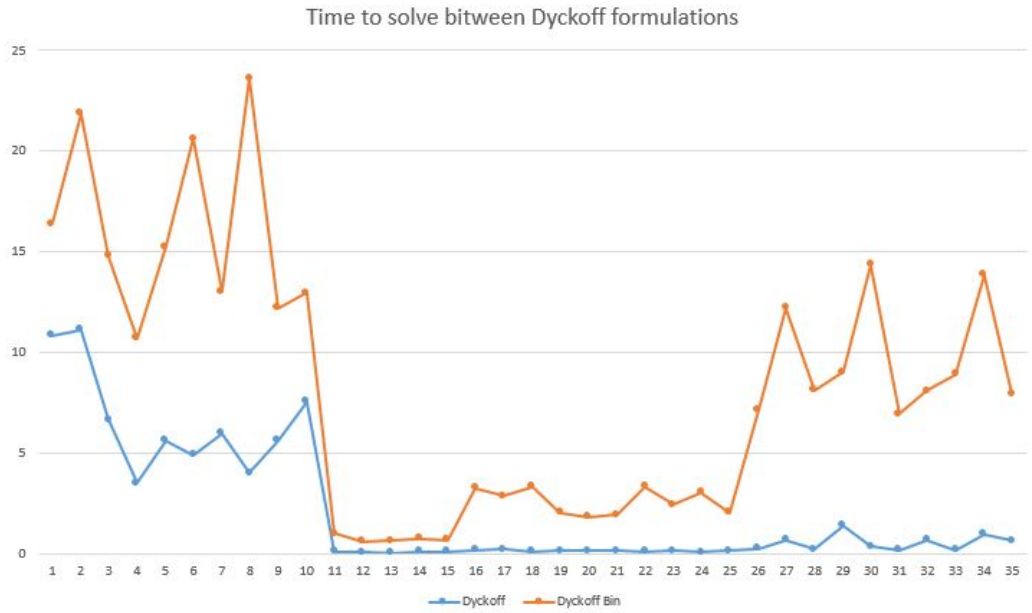

Figure 4.3:

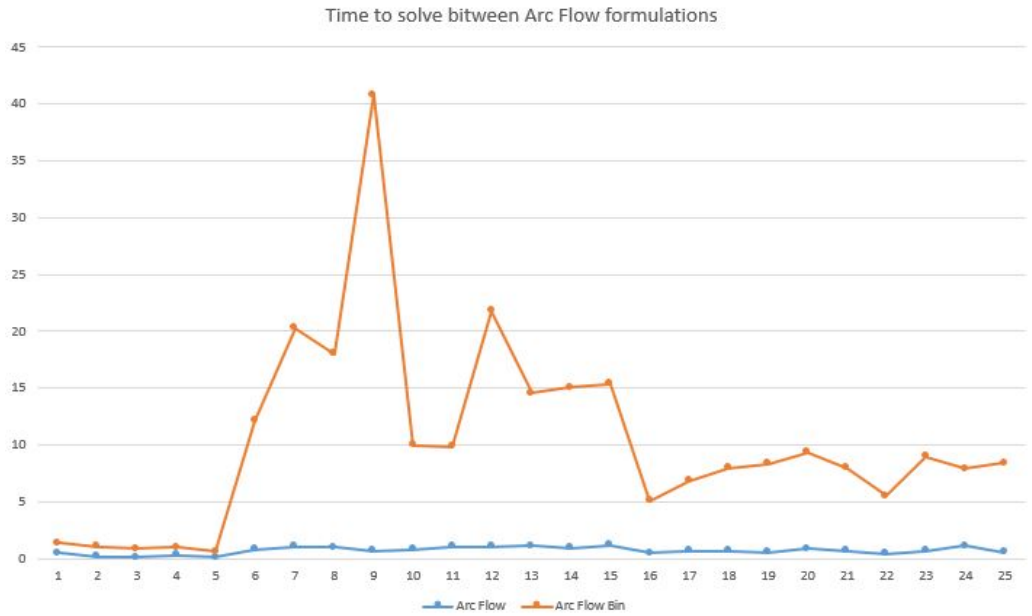

Figure 4.4:

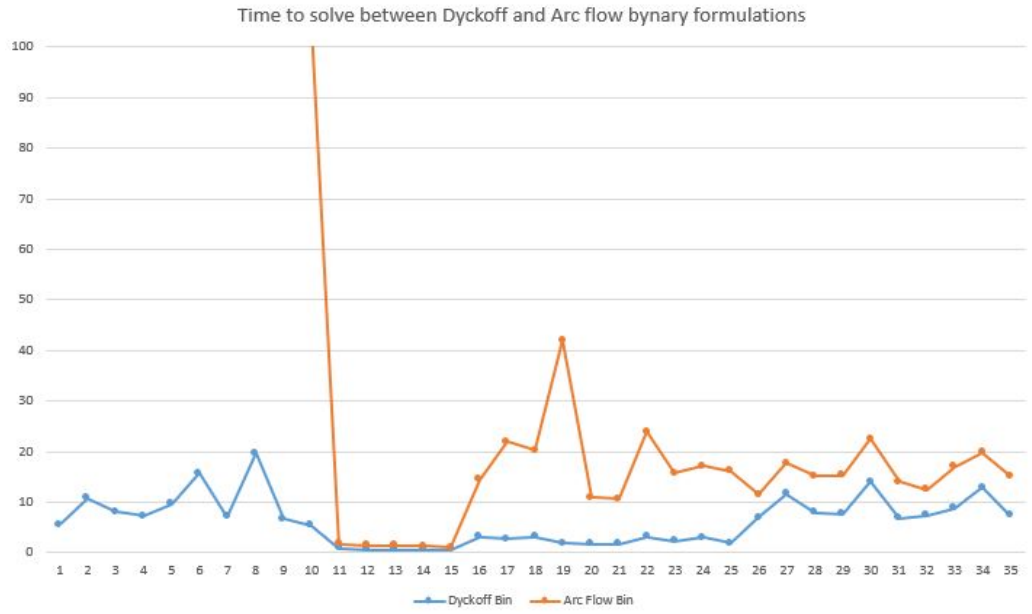

Figure 4.5:

The results show how the time to solve the problem change, even if the number of variable explode in the binary formulation the time remain stable under the 100 second to obtain the optimal value, in the set of instances tested every instance solved by the lecture model is solved even by the binary formulation, that's is important for the continue of research because change in the solution time could have a big impact to the resolution of the non linear formulation.

4.2 Continuous Non-linear model

The compact models obtained presents only binary variable and the resolution of big instances result hard to obtain, in order to explore the solution and obtaining a continuous model it is proposed the linearization of model through the use of quadratic constraints for the bin-packing model. The resulting linearization for the Kantorovich model is:

$$min \quad \sum_{j=1}^{m} y_j \tag{4.13}$$

$$s.t. \quad \sum_{j=1}^{m} x_{i,j} \geq d_i \ \ \forall i \tag{4.14}$$

$$\sum_{i=1}^{n} l_i * x_{i,j} \leq L_j * y_j \ \ \forall j \tag{4.15}$$

$$y_j^2 = y_j \ \ \forall k \tag{4.16}$$

$$x_{i,j}^2 = x_{i,j} \ \ \forall i,k \tag{4.17}$$

$$y_j \geq 0 \ \ \forall j \tag{4.18}$$

$$x_{i,j} \geq 0 \ \ \forall i, \forall j \tag{4.19}$$

The models presents non linear constraints for the representation of integrality clause, the only way known for this substitution is the use of $y_k^2 = y_k$ quadratic equation. From the literature it is specified that this quadratic equation is non-convex and the problem become a QCNCP (Quadratic-Constrained Non Convex Programming) problem. since integer programming is NP-Hard, the problem remains as NP-Hard.

following the AMPL codification of the models after the binarization and the linearization:

Kantorovich Non linear model:

1. **param n;**
2. **param m;**
3. **set roll := {1..n}; # different roll**
4. **set item := {1..m}; # item to obtain**
5. **param w {item} > 0; # leght of item i**
6. **param W {roll} > 0; # leght of roll k**
7. **var x {i in item,k in roll} >= 0 ; # item cut in roll k**
8. **var y {k in roll} >= 0 ; # utilization of roll y/n**
9. **minimize numb : sum{k in roll } y[k] ;**

10. subject to dem {i in item}: sum {k in roll} x[i,k] >= 1;
11. subject to cap {k in roll}: sum {i in item} w[i]*x[i,k] <= W[k]*y[k];
12. subject to NL1 {k in roll}: y[k] = y[k]^2;
13. subject to NL2 {i in item,k in roll}: x[i,k] = x[i,k]^2;

Dyckhoff Non linear model:

1. param n; # number of roll available
2. param m; # number of item needed
3. set roll := {1..n}; # different roll
4. set item := {1..m}; # item to obtain
5. param w {item} > 0; # leght of item i
6. param W {roll} > 0; # leght of roll k
7. param G {roll} > 0;
8. param O {roll} default 1;
9. param N {1..w[m]} >= 0;
10. set S default {};
11. set D default {};
12. set R := {w[m]..(W[n]-w[m])};
13. set H := (D union R);
14. set A := (H diff S);
15. set B := (S union R);
16. set C := (D union R);
17. set I := (A diff D);
18. var x {k in roll}>= 0; # utilization of roll
19. var y {p in B, q in A,k in roll:q<p} >= 0; # piece of width q obtained from piece of width p in roll k
20. minimize width: sum {k in roll} x[k];
21. subject to cap {k in roll, q in S : q=W[1] }: x[k] + sum {p in D:(p+q) in B} y[p+q,p,k] >= sum {p in D: (p<q)} y[q,p,k];
22. subject to dem {q in A}: sum {k in roll, p in B:p>q} y[p,q,k] + sum{k in roll, j in D:(j+q) in B} y[j+q,j,k] >= sum {k in roll, i in D:i<q} y[q,i,k] + numberof q in ({i in item} w[i]);
23. subject to nroll : sum {k in roll} x[k] <= n;
24. subject to Agg {p in B, q in I,k in roll:q<p}: y[p,q,k] = 0;
25. subject to sym {k in 1..(n-1)} : x[k] >= x[k+1];
26. subject to NL1 {k in roll}: x[k] = x[k]^2;

27. **subject to NL2 {p in B, q in A,k in roll:q<p}: y[p,q,k] = y[p,q,k]^2;**

Arc Flow Non linear model:

1. **param n; # number of roll available**
2. **param m;**
3. **set roll := {1..n}; # different roll**
4. **set item := {1..m}; # item to obtain**
5. **param w {item} > 0; # leght of item i**
6. **param W {roll} > 0; # leght of roll k**
7. **set A := {0..(W[n])};**
8. **set B := {1..(W[n]-1)};**
9. **var x {i in A,k in A,j in roll:i<k} >= 0 binary;**
10. **var y {k in roll}>= 0 binary; # utiliyation of roll y**
11. **minimize numb :sum {k in roll} y[k] ;**
12. **subject to me0 : sum {k in roll} sum {f in A:(0<f)} x[0,f,k] = sum {k in roll} y[k];**
13. **subject to mew : sum {k in roll} sum {d in A:(d<W[1])} x[d,W[1],k] = sum {k in roll} y[k];**
14. **subject to mew2 e in B : sum {k in roll} sum {d in A:(d<e)} x[d,e,k] - sum {k in roll} sum {f in A:(e<f)} x[e,f,k] = 0;**
15. **subject to qb {i in item}: sum {k in roll} sum {d in A:(d+w[i]) in A} x[d,d+w[i],k] >= numberof w[i] in ({q in item} w[q]);**
16. **subject to nroll : sum {k in roll} y[k] <= n ;**
17. **subject to NL1 {i in A,k in A,j in roll:i<k}: x[i,k,j] = x[i,k,j]^2;**
18. **subject to NL2 {k in roll}: y[k] = y[k]^2;**

Chapter 5

Non linear solver

For testing the non linear formulation could be used different type of solver, the one tested during the computational experiment are the following:

IPOPT: is a software used for large-scale nonlinear optimization which implements an algorithm of primal-dual internal points, uses a search method of filter lines to ensure global convergence.

CONOPT: is a solver for non-linear programming models with non-linear constraints. It is based on the GRG algorithm (generalized reduced gradient) and is a quick way to find an initial workable solution for problems with a few degrees of freedom. *Conopt* deletes the intermediate variables used to define the terms of the objective function by moving the constraints to the objective function.

Filter: implements a sequential quadratic scheduling solution for large non-linear problems with a discrete number of degrees of freedom. Implements a SQP trust-region algorithm with a filter to foster global convergence. A new step is accepted whenever the target or constraints on the filter are better.

Lancelot: is used for large, nonlinear problems. The basic algorithm combines the objective function and all the most complex constraints into simple limitations on variables in an augmented Lagrangian function. The function is approximately minimized within the region defined by the simple bounds.

LOQO: is based on an infeasible primal-dual interior-point method and

solves both convex and non-convex optimization problems. If definition functions are smooth, *loqo* can handle problems: linear and non-linear, convex or non-convex, bound or unconstrained. For the convex, it finds a global optimal solution, otherwise it would be interested from the point of departure specified to find a local optimal solution.

MINOS is designed to solve "smooth" nonlinear programming (NLP) problems. Smooth nonlinear functions can be accommodated in both the objective and the constraints; nonlinear equation systems may also be solved by omitting an objective. *Minos* is suitable for problems with large constraints with a linear or non-linear objective function and a combination of linear and non-linear constraints. It's more efficient if constraints are linear and there are not too many degrees of freedom. For non-linear lens functions and linear constraints, it uses a low gradient algorithm.

SNOPT: is a solver for non-linear optimization problems. It is suitable for large-scale linear and quadratic programming, linear limited optimization, and non-linear general programs. *SNOPT* locates best-of-breed solutions and implements a sequential square programming algorithm (SQP). Research directions are derived from quadratic programming subproblems that minimize a quadratic model of the Lagrangian function subject to linearized constraints. In order to ensure convergence from any starting point, an increased Lagrangian merit function is reduced along each research direction. It is particularly effective for non-linear programs whose functions and gradations are expensive to evaluate. Functions must be smooth but must not be convex.

Knitro: offers solution of:

1. general (convex or nonconvex) nonlinear problems (NLP),
2. linear problems (LP),
3. convex or nonconvex quadratic problems (QP/QCQP/SOCP),
4. linear or nonlinear least-squares problems,
5. mathematical programs with complementarity constraints (MPEC),
6. mixed-integer nonlinear problems (MIP/MINLP),
7. derivative-free optimization problems.

Key features of Knitro include:
(i) a large set of well-documented user options;
(ii) (parallel) multi-start for global optimization;
(iii) derivatives approximation and checker;
(iv) internal presolver.

BARON: is a computational system for solving non-convex optimization problems in global optimization.
Can solve simple nonlinear problems, simple integers and with combined integers. Its name derives from the combination of constraint propagation, interval analysis and duality in arsenal reduction with advanced branch and bound concepts as it allows to overcome the lower and upper peak points of complex optimization problems in search of global solutions.

COUENNE: a global optimizator widely described in the next section.

5.1 how COUENNE works

COUENNE is an open source global optimizator that allows to find global optimum solution for non linear problems even if they are non-convex. It was developed by IBM in collaboration Carnegie Mellon University and in particular, among the developers, are quoted Pietro Belotti and Leo Liberti. In the solving process, couenne act in 4 fundamental steps:

1. Reformulation and linearization;
2. Branching;
3. Heuristic to find feasible solution;
4. bound reduction;

For the exploration of nodes it is used the Spatial Branch-and-Bound method that creates a hierarchy of nodes represented by a binary tree, in the creation of subproblem the solver follows 4 essential steps:

- computation of lower bound of the subproblem;
- computation of upper bound or feasible solution;
- branching to partition to new subproblem;
- bounds tightening to reduce the feasible space;

In the following section, are proposed the COUENNE procedures in general.

5.1.1 Reformulation

The first step for the COUENNE solution is the reformulation and it consists in the introduction of auxiliary variables for the factorization of the function of model, in general:

$$\begin{array}{lcl} min\ f(x) & & min\ x_{n+q} \\ s.t\ g_j(x) \leq 0\ \ \forall j \in M & & s.t\ x_i = \vartheta_i(x)\ \ \forall i \in Q \subseteq N \\ x_i^l \leq x_i \leq x_i^u\ \ \forall i \in N_0 & \Longleftrightarrow & x_i^l \leq x_i \leq x_i^u \forall i \in N \\ x_i \in Z\ \ \forall i \in N_0^I \subseteq N_0 & & x_i \in Z\ \ \forall i \in N_I \subseteq N \end{array}$$

Where $\vartheta_i(x)$ contains all of the auxiliary variable associate with operators $\Theta =$ {sum,product,quotient,ecc} for the factorization of the function. The reformulation is carried out to create a simple structure than the original one with more variable and nonlinear constraints

In the case of the Kantorovich model, the reformulation increases the number of variables and decreases the number of constraints, the reformulation is the follow:

$$min\ w_1 \tag{5.1}$$
$$s.t\ w_i = 0\ \ \forall\ 1 < i \leq n * m + n \tag{5.2}$$

variable:

$$x_{i,j} \in (0,1)\ \ \forall\ 0 \leq i \leq n\ \ \forall\ 0 \leq i \leq m \tag{5.3}$$
$$w_{(n*m+n)+j} \geq \sum_i (x_{i,j} * \frac{w_i}{W})\ \ \forall j \tag{5.4}$$
$$w_1 = \sum_{j=1}^{n} w_{(n*m+n)+j} \tag{5.5}$$
$$w_{k=(n*i+j)} = x_{i,j} - x_{i,j}^2\ \ \forall i,j \tag{5.6}$$

From this reformulation it can be easily obtained the number of new constraint and variable for every instance of the problem.
in specific:
if n is the number of bin usable and m is the number of objects, the number of new variable will be:
$Nnv = 2 * (n * m + n) + n + m - m + (mxn) + 1$
$Nnv = 3 * (n * m) + 3 * n + 1$
while the number of new constraints is the same of the old variable that it is $Nv = (n * m + n)$ for Kantorovich non-reformulated model.
The option to display the reformulation is '**problem_print_level 7**'.

Following is proposed the output of the solver for an instance of 4 bin and 10 objects:

Problem size before reformulation: 44 variables (0 integer), 58 constraints.
Problem size after reformulation: 133 variables (0 integer), 44 constraints.

5.1.2 Linearization

The first step that COUENNE realizes, consists on the reformulation and linearization of the problem to obtain its linear relaxation. In fact, through the formulation of a linear relaxation of a problem, any admissible solution of the starting problem P_0 is also an admissible solution of its linear relaxation LP_0, on the other hand an optimal solution $\overline{x}$ of the latter represents a lower bound valid for the optimum of P_0.
Firstly this step is analysed in a general type of problem and then it is study in what way it is realized in bin-packing specific model.
Reformulation is used to represents the problem in an equivalent way that is easier to deal with from a symbolic viewpoint and it is executed only in the starting problem, then in the initial node of the branching process, while linearization allows to realize the linear relaxation of the problem and it is performed at all nodes of the branching process.
To analyse the process of linearization from a general point of view, consider the follow structure of the problem P, obtained after the reformulation

$$min\ x_{n+q} \tag{5.7}$$
$$s.t\ x_i = \vartheta_i(x) \quad \forall i \in Q \subseteq N \tag{5.8}$$
$$x_i^l \leq x_i \leq x_i^u \quad \forall i \in N \tag{5.9}$$
$$x_i \in Z \qquad \forall i \in N_I \subseteq N \tag{5.10}$$

From which is relevant the constraint $x_i = \vartheta_i(x)$, with $\vartheta_i \in \theta$, and $x \in [x^l, x^u]$. The inequality $ax \geq b$ can be a linearization inequality for $x_i = \vartheta_i(x)$ if it is satisfied by all points $x \in [x^l, x^u]$ such that $x_i = \vartheta_i(x)$. Therefore, it can be said that a linearization for that constraint is a system of linear inequalities, like the aforementioned one, and it is represented with $A^i x \geq b^i$. Proceeding to define a linearization for each variable $x_i = \vartheta_i(x)$, with $i \in Q \subseteq N$, the linear relaxation of the initial problem P_0 can be defined as $LP_0 : min\{x_{(n+q)} : Ax \geq b\}$.
In particular, if it is considered that the variable x_i depends on a single one x_j, where $x_j^l \leq x_j \leq x_j^u$, through the equality $x_i = \vartheta_i(x_j)$, its linearization is given by the set of points (x_j, x_i) such that the inequality $a_h x_j + b_h x_i \geq c_h$, with $h = 1, ..., H$ is satisfied.
In this step the goal of COUENNE is to develop a tight linearization of the problem, trying to use a small number of linear inequalities. Below it is analysed in what way a constraint can be linearized, considering a univariate function $x_i = \vartheta_i(x_j)$, because in the particular model that will be analysed, there will be constraints in which every variable depends only on a unique auxiliary variable.
If ϑ_i is convex, valid linear inequalities, associated to the constraint $x_i = \vartheta_i(x_j)$, is given by the equations of its tangent lines in each point $\widetilde{x}_j \in [x_j^l, x_j^u]$ in this way:

$$x_i \geq \vartheta_i(\widetilde{x}_j) + \frac{\delta \vartheta_i}{\delta x_j}(\widetilde{x}_j)(x_j - \widetilde{x}_j) \tag{5.11}$$

COUENNE, initially, adds a linearization inequality for each x_j in a discretization of the interval $[x_j^l, x_j^u]$, so that a reasonable approximation is obtained.

The same process ca be used to linearize the constraints in which the function ϑ_i is concave, considering that in this case the graph of the function is all below the tangent in every points, so valid linear inequalities are:

$$x_i \leq \vartheta_i(\widetilde{x}_j) + \frac{\delta\vartheta_i}{\delta x_j}(\widetilde{x}_j)(x_j - \widetilde{x}_j) \tag{5.12}$$

Linearization process starts at the initial problem P_0 creating its linear relaxation LP_0 and it continues also when the various subproblems of P_0 is analysed, creating an sBB tree with k nodes. Thus, linearization is not done only at the root node, but it must also be improved in every k node. This is necessary to improve the lower bound of the problem and to find a solution, so it is important to explain in what way linearization can be refining in each node.
If the solution $\overline{x^k}$ of the linear relaxation LP_k is infeasible for the subproblem P_k, the lower bound $\overline{x}^k_{(n+q)}$ (that is given by the optimal solution of LP_k, can be improved by branching, or by a refinement procedure of the linear relaxation obtained by modifying or appending some linear inequality.
In this sense a separation problem can be used: if $\overline{x^k}$ is a solution for LP_k and not for P_k, a new linear relaxation LP'_k ,in which $\overline{x^k}$ is an infeasible solution, can be realized and solved for P_k. LP' is obtained appending a linear inequality to LP_k that separate $\overline{x^k}$ from solutions. However, if $\overline{x^k}$ is feasible for any possible linearization of P_k, the separation process cannot be used and branching is necessary.
If a refinement process is possible, it can be realized for a given number of iteration , until a feasible solution for P_k is found, or until there no other possibility to refine the linear relaxation and so branching becomes necessary. Refining a linearization, when it is possible, let to improve the lower bound without creating too many nodes.
COUENNE uses the refining process realizing the separation problem for each auxiliary variable $x_i = \vartheta_i(x)$.
In particular, in the case of convex univariate constraint $x_i = \vartheta_i(x_j)$, if the point $(\overline{x}^k_j, \overline{x}^k_i)$ satisfies the inequalities $\overline{x}^k_i > \vartheta_i(\overline{x}^k_j)$, it is contained in any linearization of ϑ_i and the separation problem cannot be used, so branching is the only possible solution. On the other hand if the point $(\overline{x}^k_j, \overline{x}^k_i)$ satisfies the inequalities $\overline{x}^k_i < \vartheta_i(\overline{x}^k_j)$, there is a linear inequality that is

violated by $\overline{x^k}$ and that can be appended to the linear relaxation, and it is the tangent line to ϑ_i in $(\overline{x}_j^k, \vartheta_i(\overline{x}_j^k)$ that is,

$$x_i \geq \vartheta_i(\overline{x}_j^k) + \frac{\delta \vartheta_i}{\delta x_j}(\overline{x}_j^k)(x_j - \overline{x}_j^k) \tag{5.13}$$

Specifically, COUENNE finds the linear inequalities that realizes the deepest cut that is obtain using the line tangent ϑ_i at the point of the curve that is closest to $(\overline{x}_j^k, \overline{x}_i^k)$.
This is obtained by solving a one-dimensionale, convex optimization problem, using a Newton method, through which it improves the quality of linearization.
Consider the situation in which separation problem cannot be used, branching process is analized in a general problem.

5.1.3 Branching

Branching is a technique used with bounds tightening in the Branch-and-Bound method to partition a problem P_0 into easier subproblems obtained by partitioning the original solution space.
Each subproblem is solved separately and, if it is necessary, they could be partitioned into other subproblems with a recursively process of partitioning, until a satisfactory solution is found. This process is represented through a tree with a series of nodes, in which any node represent a particular subproblem. In a branching strategy a node k is partitioned to improve the lower bound obtained by solving the subproblems created, or to create subproblem with similar difficulty, or to eliminate a portion of the solution space of the initial problem to reduce the set of solutions.
The goals of branching is to minimize the number of nodes generated. It is analysed the case of branching on a variable. In this branching strategy a variable x_j is selected to do the partitioning process and for this variable it is necessary to locate a branching point, in particular a lower branching point b_l and a upper branching point b_u, with $b_l \leq b_u$, which are used to create the two subproblem, one created appending the linear constraint $x_j \leq b_l$, and the other appending the linear constraint $x_j \geq b_u$. Thus, in summary, there are two steps in the branching process on a I

- The selection of the branching variable
- The selection of the branching point.

These two steps are easy in the case of problems with integer variables. In fact, in this case, the method chooses as a branching variable for the problem P_k a value x_j^k that is fractional. Then, the branching points are chosen placing $b_l = \lfloor x_j^k \rfloor$ and $b_u = \lceil x_j^k \rceil$, so the two constraint used to create two new subproblems of P_k are: $x_j \leq \lfloor x_j^k \rfloor$ and $x_j \geq \lceil x_j^k \rceil$.
When branching have to be applied to a continue variable, the situation becomes more complex, and some methods have to be introduced to explain what criteria are used to realize branching. In this case, a value b have to be selected as the branching point to cut point $(\overline{x}_j^k, \overline{x}_i^k)$ from the solutions. This point is a feasible solution for the linearization of $x_i = \vartheta_i(x_j)$ but it is not separable with a linearization cut, so branching is necessary. In this way two subproblems are created through the constraints $x_j \leq b$ and $x_j \geq b$, where both of them exclude the point $(\overline{x}_j^k, \overline{x}_i^k)$ from solutions.
COUENNE for the choice of branching variables uses a priority criterion: firstly it selects integer variable with a fractional value, and, if and only if there are no integer variable, it selects a continues one. A simple method to select a branching variable could be br-plain. Consider the constraint $x_i = \vartheta_i(x)$, to select the branching variable this method uses the linear combination:

$$\begin{aligned} \Omega_j^N(\overline{x}^k) = & \mu 1 \sum_{i \in E(j)} \delta_i(\overline{x}^k) + \\ & + \mu 2 max_{i \in E(j)} \delta_i(\overline{x}^k) + \\ & + \mu 3 min_{i \in E(j)} \delta_i(\overline{x}^k), \end{aligned} \tag{5.14}$$

Since $\Omega_j^N(\overline{x}^k)$ is hardly correlated with the improvement of the lower bound is important to define right strategies, in the solver are implemented: the *Violation Transfer* and the *reliability branching*.

Chapter 6

Computation experiment on non-linear solver

The first analysis for the Non linear problems is how they work on the different solver available in NEOS server discussed in the previous chapter, after the first analysis experiment will be done on the one that gives the optimal value or a good approximation.

solver	problem charatestic				Message displayed or obj value		
	N° item	n/max bir	optimal		Kantorovich		
			A	B	A	B	
COUENNE	10	19	3	4	0,031 (s) 3	2,87 (s) 4	
CONOPT	10	19	3	4			Infeasibility in pre-triangular part of model
Filter	10	19	3	4			Nonlinear constraints locally infeasible
IPOPT	10	19	3	4			Converged to a locally infeasible point.
Lancelot	10	19	3	4	2,41	3,07	could not find a feasible solution.
LOQO	10	19	3	4	2,36	3,04	iteration limit reached (500 iteration)
MINOS	10	19	3	4			infeasible problem
SNOPT	10	19	3	4	2,68	4,31	Nonlinear infeasibilities minimized
Knitro	10	19	3	4			Convergence to an infeasible point
BARON	10	19	3	4	2,99	3,99	CPU time limit reached

solver	problem charatestic				Message displayed or obj value		
	N° item	n/max bir	optimal		Dyckhoff		
			A	B	A	B	
COUENNE	10	19	3	4	3 upper bound	4 upper bound	
CONOPT	10	19	3	4			Infeasibility in pre-triangular part of model
Filter	10	19	3	4			Nonlinear constraints locally infeasible
IPOPT	10	19	3	4			Converged to a locally infeasible point.
Lancelot	10	19	3	4	1,29	1,87	could not find a feasible solution.
LOQO	10	19	3	4	2,59	3,42	iteration limit reached (500 iteration)
MINOS	10	19	3	4			infeasible problem
SNOPT	10	19	3	4	3	5	Nonlinear infeasibilities minimized
Knitro	10	19	3	4			Convergence to an infeasible point
BARON	10	19	3	4	2,99	3,99	stopped at 100 (s)

solver	problem charatestic				Message displayed or obj value		
	N° item	n/max bir	optimal		Arc flow		
			A	B	A	B	
COUENNE	10	19	3	4			no solution after 100 s
CONOPT	10	19	3	4			Infeasibility in pre-triangular part of model
Filter	10	19	3	4			Nonlinear constraints locally infeasible
IPOPT	10	19	3	4			Problem has too few degrees of freedom
Lancelot	10	19	3	4	2,57	3,04	could not find a feasible solution.
LOQO	10	19	3	4	0,64	2,03	iteration limit reached (500 iteration)
MINOS	10	19	3	4			infeasible problem
SNOPT	10	19	3	4	3	5	Nonlinear infeasibilities minimized
Knitro	10	19	3	4			Convergence to an infeasible point
BARON	10	19	3	4	2,99	4	stopped at 100 (s)

Figure 6.1:

solver	problem charatestic			Message displayed or obj value	
	N° item	Wmax bin	optimal		Kantorovich
COUENNE	50	100	20	19,84 lower bound	
LOQO	50	100	20	18,75	iteration limit reached (500 iteration)
SNOPT	50	100	20	20,32	Nonlinear infeasibilities minimized
BARON	50	100	20	21	CPU time limit reached

solver	problem charatestic			Message displayed or obj value	
	N° item	Wmax bin	optimal		Dyckhoff
COUENNE	50	100	20		no value
LOQO	50	100	20	49,01	iteration limit reached (500 iteration)
SNOPT	50	100	20		no value
BARON	50	100	20	20	optimal

solver	problem charatestic			Message displayed or obj value	
	N° item	Wmax bin	optimal		Arc flow
COUENNE	50	100	20		no value
LOQO	50	100	20	49,01	iteration limit reached (500 iteration)
SNOPT	50	100	20		no value
BARON	50	100	20		no iteration

Figure 6.2:

From the solution obtained it could be seen that, even if the instance is small respectively the one tested in the other experiment, some solvers are not indicated to solve the model, in particular: **CONOPT, Filter, IPOPT, Lancelor, MINOS** and **Knitro** don't have numerical value for the object functions or is distant more than 1 of the optimal solution, for that reason the experiment continues in the remain solver that can give different lecture on the value obtained and can reach the optimal.
In particular, **COUENNE** obtain the optimal solution for the Kantorovich Non linear model and gives good upper bound for the Dyckhoff one.
Overall in the first test the better solver is **BARON** it isn't stopped by a criteria but by a time option in the AMPL command, anyway the solution obtained are strictly close to the optimal (<0,0001) the only problem is that the model continues to try rounding continuous value close to the integer value without success, from the partial solution it could be easily obtained the optimal rounding the continuous value to the optimal.
LOQO and **SNOPT** are choses for the bound given minor to one from the optimal value.
Following other test on bigger instance:

solver	problem charatestic			Message displayed or obj value	
	N° item	Wmax bin	optimal		**Kantorovich**
COUENNE	20	1000	10	10 upper bound	
LOQO	20	1000	10	8,12	iteration limit reached (500 iteration)
SNOPT	20	1000	10	10,77	Nonlinear infeasibilities minimized
BARON	20	1000	10	9,99	CPU time limit reached

solver	problem charatestic			Message displayed or obj value	
	N° item	Wmax bin	optimal		**Dyckhoff**
COUENNE	20	1000	10	9 lower bound	
LOQO	20	1000	10	15,12	iteration limit reached (500 iteration)
SNOPT	20	1000	10	9,1	Nonlinear infeasibilities minimized
BARON	20	1000	10		no iteration

solver	problem charatestic			Message displayed or obj value	
	N° item	Wmax bin	optimal		**Arc flow**
COUENNE	20	1000	10		exceeded the maximum memory for a job
LOQO	20	1000	10		exceeded the maximum memory for a job
SNOPT	20	1000	10		exceeded the maximum memory for a job
BARON	20	1000	10		Time limit exceeded in parsing phase

Figure 6.3:

The latest solution provides a lecture about the work done, brings to the continuous space the models create an elevate number of variable to compute and every variable have a Non linear constraint, a sign is the fact that the Kantorovich model is the easier one to compute. the Experiment done on the solver gives good result even if didn't improve the actual optimal solution, they reach a level that solving instances with more than 100.000 non linear constraint took less than 100 seconds.

Conclusion

The work done for the thesis could be seen as a starting point to the exploration of the method applied,
(i) for the standard that it can be in the transformation of the model to non linear formulation and (ii) the new way to explore the bin-packing problem.

for the (i) argument it could be interesting to use with the development of non-linear solve that can easily solve the non linear constraint used in the models obtained, with that assumption the solver could be used to solve the hard instances of the literature to examine how the non linear solver work without integer or binary variable.

(ii) The exploration of the bin-packing problem used could have a lot of enhancement, in a first analysis there are new and interesting study on the binarization method that use binary variables ($\sum 2^i * y^i$) for the representation of an integer number obtaining a reduction of the variable and constraints explored by the non linear solver, other work could be done in the reduction of the large number of zeros that the solution have even for the literature models, for the reduction of variable explored could be implemented in the instances the use of the code obtained in the section (3.3) the reduction of the bin and the parameter k could halve the number of variable obtaining an improvement on the computational operations, some experiment show how the number of variable didn't affect the time to solve the instances but generating a large number of constraint in the non linear solvers explode the time of setting the problem and sometime exceed the maximum capacity of the memory storage, for that reason the Kantorovich model have better results in the non linear experiment while in the others is the worst one.

Acknowledgements

For this work, first of all, i want to thank my professor Fabrizio Marinelli, for his passion on the field and the capacity to involve his student, through him it was possible the experience in Portugal.

I want to thank the academic crew of FEUP in particular the professor José Fernando Oliveira for the assistence and the ideas shared.

I wish to thank the person that shared the experience in Porto with me in particular: Kübra, Yegor, Petry, Judit, Marco, Sevda.

I wish to thank my oldest friend Francesco and Riccardo for their constant and unfailing support.

I wish to thank Abdul: we start together and we will finish together.

I wish to thank Marco T. for his presence in this period.

I wish to thank everyone that helped me during the university experience and make it lighter.

For my career and my life I thank my family for their support even when the difficult moment strikes.

References

1. J.M. Valério de Carvalho. LP models for bin packing and cutting stock problems. European Journal of Operational Research, 141(2):253-273, 2002.

2. F. Marinelli, Models and algorithms for one-dimensional cutting problems, UNIVPM Università Politecnica delle Marche, Università Tor Vergata,Roma, 2013.

3. F. Marinelli, Ottimizzazione discreta: modelli e algoritmi, UNIVPM Università Politecnica delle Marche.

4. C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance, Branch-and-price: Column generation for solving huge integer programs, Operations Research 46 (1998) 316-329.

5. J.M. Valério de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, Annals of Operation Research 86 (1999) 629-659.

6. J.M. Valério de Carvalho, Using dual cuts to accelerate column generation processes, Working paper, Departamento de Producéao e Sistemas, Universidade do Minho, 2000.

7. J.M. Valério de Carvalho, A note on branch-and-price algorithms for the one-dimensional cutting stock problem, Computational Optimization and Applications 21 (3) (2002) 339-340.

8. H. Dyckhoff, A new linear programming approach to the cutting stock problem, Operations Research 29 (1981) 1092

9. H. Dyckhoff, Production theoretic foundations of cutting and related processes, in: G. Fandel, H. Dyckhoff, J. Reese (Eds.), Essays in Production Theory and Planning, Springer, Berlin, 1988, pp. 151-180.

10. H. Dyckhoff, A typology of cutting and packing problems, European Journal of Operational Research 44 (1990) 145-159.

11. P. Gilmore, Cutting stock, linear programming, knapsacking, dynamic programming and integer programming, some interconnections, in: P.L. Hammer, E.L. Johnson, B.H. Korte (Eds.), Annals of Discrete Mathematics, vol. 4, North-Holland, Amsterdam, 1979, pp. 217-236.

12. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem, Operations Research 9 (1961) 849- 859

13. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem - part II, Operations Research 11 (1963) 863-888.

14. P. Vance, Branch-and-price algorithms for the one-dimensional cutting stock problem, Computational Optimization and Applications 9 (1998) 211-228.

15. P. Vance, C. Barnhart, E.L. Johnson, G.L. Nemhauser, Solving binary cutting stock problems by column generation and branchand- bound, Computational Optimization and Applications 3 (1994) 111-130.

16. F. Vanderbeck, On Dantzig?Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm, Operations Research 48 (2000) 111-128.

17. G. Waescher, T. Gau, Heuristics for the integer one-dimensional cutting stock problem: A computational study, OR Spektrum 18 (3) (1996) 131?144.

18. S. MARTELLO, P. TOTH, LOWER BOUNDS AND REDUCTION PROCEDURES FOR THE BIN PACKING PROBLEM, Discrete Applied Mathematics 28 (1990) 59-70 North-Holland

19. M. Delorme, M. Iori, S. Martello, Bin Packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms, Research Report OR-15-1,DEI Guglielmo Marconi, University of Bologna, DISMI, University of Modena and Reggio Emilia.

20. L. V. Kantorovich, MATHEMATICAL METHODS OF ORGANIZING AND PLANNING PRODUCTION, Leningrad State University 1939.

21. Johnson, D.S.: Near-optimal bin-packing algorithms. Doctoral Thesis. MIT, Cambridge (1973).

22. Baker, B.S.: A new proof for the first-fit decreasing bin-packing algorithm. J. Algorithms, 49-70 (1985).

23. P. Belotti, J. Lee, L. Liberti, F. Margot, A. Waechter, Branching and Bounds Tightening Techniques for Non-Convex MINLP, IBM Research Report, RC24620 August 13,2008.

24. D. S. Johnson, M. R. Garey, A 71/60 Theorem for Bin Packing, JOURNAL OF COMPLEXITY 1,65-106 (1985).

25. G. Scheithauer, J. Terno, A. Muller, and G. Belov. Solving one-dimensional cutting stock problems exactly with a cutting plan.

26. C. Nitsche, G. Scheithauer, and J. Terno. Tighter relaxations for the cutting stock problem. European Journal of Operational Research, 112(3):654-663, 1999.

27. S. Martello and P. Toth. Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Chichester, 1990.

28. L.V. Kantorovich. Mathematical methods of organizing and planning production. Management Science, English translation of a 1939 paper written in Russian, 6(4):366-422, 1960.

29. G. Belov and G. Scheithauer. A branch-and-cut-and-price algorithm for onedimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research, 171(1):85-106, 2006.

30. G. Belov and G. Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. European Journal of Operational Research, 141(2):274-294, 2002.