Solving Integer Programming Models for the Multi-Level Bin Packing Problem with Conflict Constraints

Jakub Tokarz
Supervisor: Dr. Matthias Horn
Responsible Professor: Assoc. Prof. Dr. Neil Yorke-Smith
EEMCS, Delft University of Technology, The Netherlands
20-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

The Multi-Level Bin Packing problem and its variants are some of the most popular combinatorial optimization problems. They have a wide range of real-life applications yet, they are some of the harder problems we know of. In this paper we solve the Multi-Level Bin Packing Problem and a variant of it, Multi-Level Bin Packing Problem with Conflict Constraints, using Integer Programming. We propose two Integer Programming models, a standard one and one with additional flow optimizations. We hypothesized that the second model will have a smaller solution space and consequently an improved performance for both of the problems. However, we find that the second model's performance is worse and that although it may have a smaller solution space, the comparison of the BnB nodes points towards inconclusiveness.

1 Introduction

The Bin Packing (BP) problem "is one of the best-known and most widely studied problems of combinatorial optimization" [1]. Finding an optimal solution of a BP problem instance is an NP-hard problem, with the corresponding problem of determining whether an instance of the BP problem has a solution, which is NP-Complete [2]. The BP problem has countless real life applications which include: supply chain management, routing and resource allocation, financial budgeting or scheduling, to name a few [3, 4]. A generalisation of the BP problem, known as the Multi-Level Bin Packing (MLBP) problem, has numerous real life applications such as: coating of tools with several required intermediate steps, or network load balancing [5]. Additionally, it can be used for logistic planning for products that need to be packed into boxes before they are loaded into different containers for shipping. Many variants of MLBP, like the Multi-Level Bin Packing Problem with Conflict Constraints (MLBPCC), also have several different real world applications. An example of an MLBPCC application is a scenario in which items may be conflicting with each other, like food products and heavy chemicals, but still have to be packed into a minimum amount boxes before they are loaded into different containers.

Although the magnitude of usefulness for both BP and MLBP is similar, MLBP is not nearly as much considered in the literature as BP is. Table 1 shows the search results for both keywords from some of the top scientific publication search engines and highlights this vast difference. This is also the case for MLBPCC which is omitted from the table since none of the mentioned search engines yielded any results for the keyword of "Multi-Level Bin Packing Problem with Conflict Constraints". Due to this research gap, the focus of this research paper is to explore the MLBP and its variant, MLBPCC.

As mentioned before, BP, but also MLBP(CC), are NP-Hard combinatorial optimization problems, meaning there is no algorithm that we know of that solves any of these problems in polynomial time. A paper by Chen et al. was one of the few papers that researched a way of solving MLBP. It did so by using dynamic programming to find optimal solutions for instances of what they call "normal" sized problems, and a multi-level fuzzy matching algorithm to solve problems for "large" instance sizes [6]. A promising alternative approach for solving combinatorial optimization problems is Mixed Integer Programming (MIP) [7], which is a generalisation of the Integer Programming (IP). This is where a problem is reduced to an instance of an (M)IP problem and then specialized (M)IP solver software is used to solve it. One of the most popular professional grade optimization software that is used for solving (M)IP problems, and that will be used in this project, is the CPLEX Optimizer. It is designed to provide flexible, high-performance, mathematical programming solvers for linear programming and (mixed) integer programming, naming just a few [8].

Scientific publication search engine	"Multi-Level Bin Packing"	"Bin Packing"
Google Scholar	7	39600
Semantic Scholar	4	9130
Scopus	3	3108
Web of Science	12	5120
IEEE Xplore	2	1010
TU Delft Library	2	1503

Table 1: Amount of search results given by each scientific publication search engine, for keywords "Multi-Level Bin Packing" and "Bin Packing" on the 6th of May 2022.

The aim of this project is to find, implement, optimize and evaluate IP models for the Multi-Level Bin Packing problem as well as for Multi-Level Bin Packing problem with Conflict Constraints. More specifically, we will answer the question: Do flow based IP models outperform other kind of IP models to solve MLBP(CC) instances in terms of solution time and the number of branch-and-bound nodes required?

The following section, Section 2, formally describes the BP problem, the MLBP problem and the variant of it, MLBPCC. Next, Section 3 illustrates the IP approach and then explains the mathematical reasoning with regard to IP representations of MLBP and MLBPCC. Then, Section 4 discusses the experimental setup used for evaluating different models, and brief instructions on how to run the set up are described in Section 5. The results based on the empirical measurements of the performances of the two models are described in Section 6. The discussion of the results and potential explanation for why our initial hypothesis was wrong are reasoned about in the Section 7. Next, the ethical aspects of this research and the reproducibility of it are reflected upon in Section 8. Finally, Section 9 contains the summary of this paper, the concluding remarks and the future recommendations for research in this topic.

2 Formal Problem Description

2.1 Bin Packing

The standard version of the BP problem consists of a set of n items and an infinite set of bins where each item has an as-

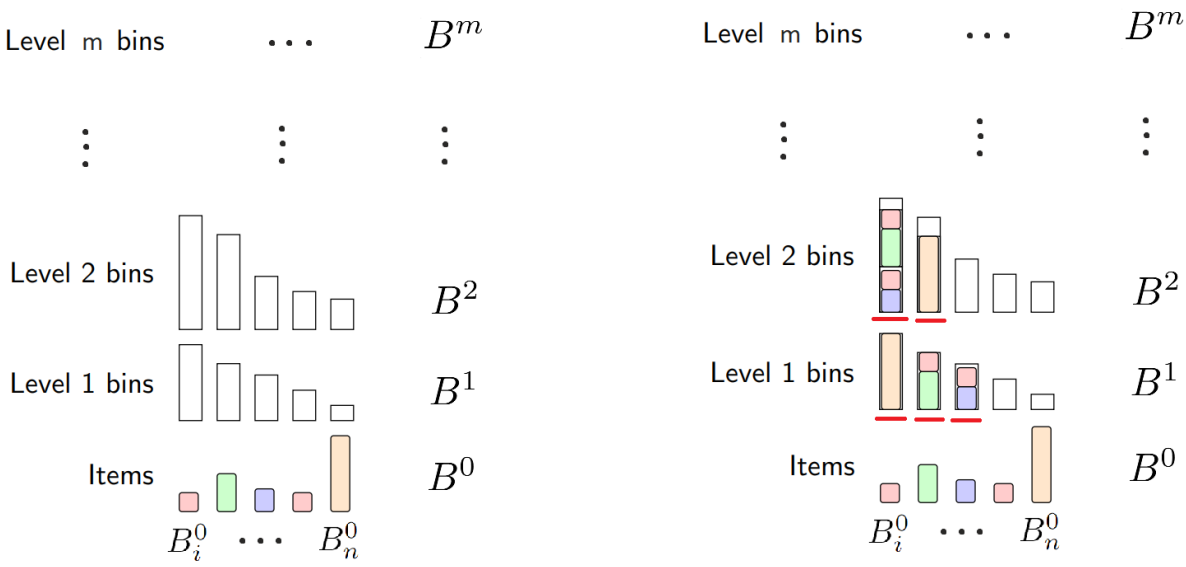

Figure 2: Left: Given MLBP Instance with $n = 5$ and $m = 2$. Right: Solved MLBP instance, with the total cost equal to the sum of the costs of the 5 used bins, highlighted in red.

sociated size a_i and bins have equal associated capacity W. The task is to insert all items into the bins such that the total number of bins used is minimized while adhering to the constraint that no bin's capacity is exceeded by the combined size of all the contained items. A visualization of this can be seen in Figure 1.

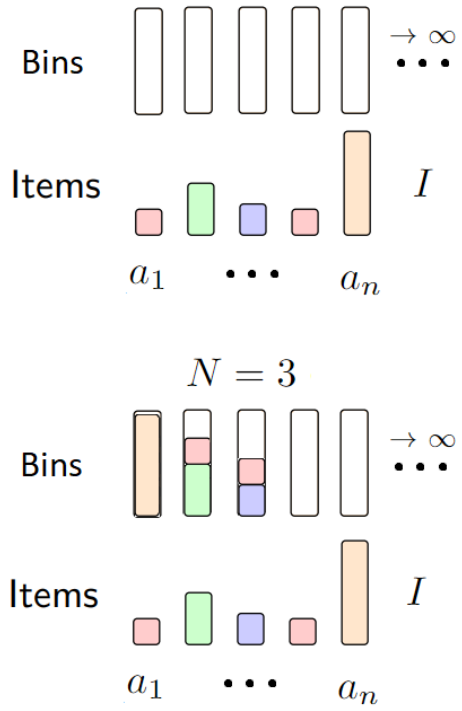

Figure 1: Top: Given instance of BP with $n = 5$. Bottom: Solved instance of BP with $N = 3$.

More formally: given a list $I = (a_1, ..., a_n)$ of sizes of n items, where $a_i \in \mathbb{N}_{>0}$ denotes the size of the item i and the capacity W of each bin. Find an assignment $f : \{1, ..., n\} \to \{1, ..., N\}$, with $\sum_{i:f(i)=j}^{I} a_i \leq W_j$ for all $j \in \{1, ..., N\}$ such that $N \in \mathbb{N}$ is minimum. [2]

2.2 Multi-Level Bin Packing

The MLBP problem is a generalisation of the BP problem with some differences. First, in MLBP there is a finite amount of bins, all of varying capacities and sizes. Second, bins are partitioned into m levels. Before an item can be put in one of the final top level bins at level m, it must be first put into a bin of the first level, which in turn must be put into a bin of the second level. This is continued until the item is finally packed into a bin of the top level. The goal of MLBP is to put each item in one of the top level bins while minimising the total cost of the bins used at all levels. All this has to be done while adhering to the capacity constraint of each bin used. A visualization of this can be seen in Figure 2. More formally: given $m \in \mathbb{N}_{>0}$ levels, where each level k is represented by a list of bins B^k for $k \in \{1, ..., m\}$. Each bin B_j^k at level k, has a designated size $s(B_j^k)$, capacity $w(B_j^k)$ and cost $c(B_j^k)$. Additionally a list of n items, B^0, is given, which like bins have a size, however items do not have capacity nor cost, therefore $w(B_i^0) = \emptyset \wedge c(B_i^0) = \emptyset \wedge s(B_i^0) \in \mathbb{N}_{>0}, \forall i \in B^0$. Find an assignment of item indices $\{1, ..., n\}$ to bin indices at each of the levels $(\{1^1, ..., j^1\}, ..., \{1^m, ..., j^m\})$, such that the sum of the sizes of all the items/bins $(B_{j_i}^k)$ in, bin j at level k, B_j^k is less than or equal to its capacity, $\sum_{i \in B_j^k} s(B_{j_i}^k) \leq w(B_j^k)$ for all bins $j \in B^k$ at level k, for every $k \in \{1, ..., m\}$. The assignment f should minimize the cost of all the bins used, $\min \sum_{B_j^k \in T} c(B_j^k)$ where T is the set of all the bins used.

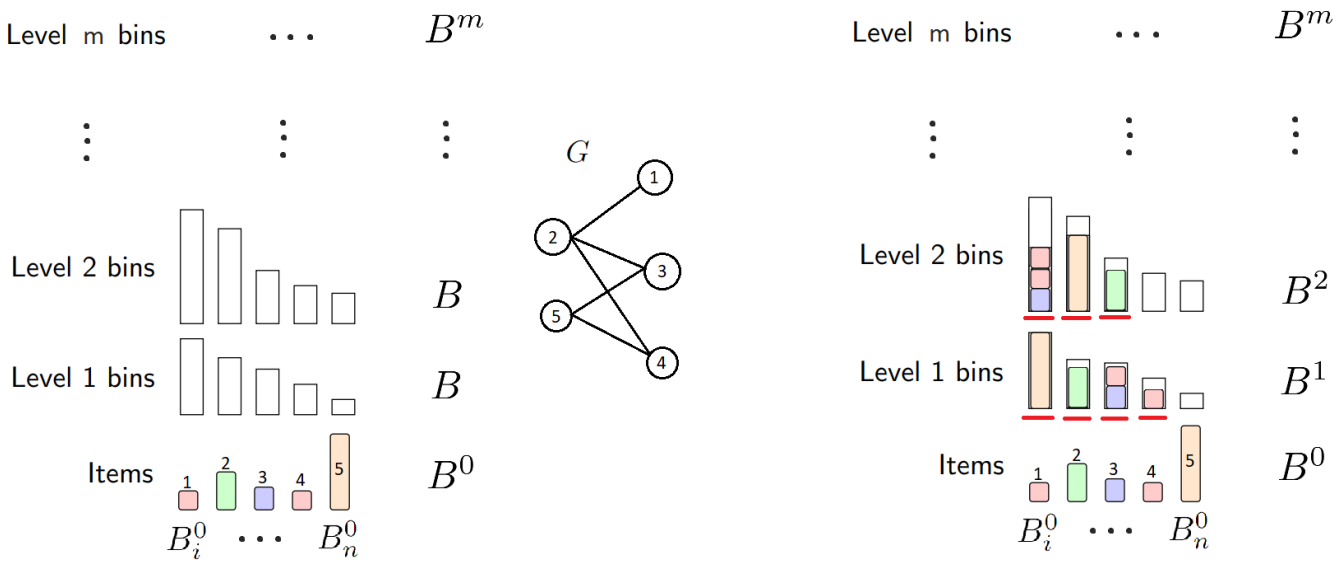

Figure 3: Left: Given MLBPCC Instance with $n = 5$ and $m = 2$ and the conflict graph G. Right: Solved MLBPCC instance, with the total cost equal to the sum of the costs of the 7 used bins, highlighted in red.

2.3 Multi-Level Bin Packing With Conflict Constraints

A variant of the MLBP problem this paper is also focusing on is the MLBPCC problem. On top of the standard MLBP instance, a conflict graph between the items is given. Items that share an edge in this graph have a conflict relationship and cannot be put in the same bin at any level. A visualisation of this can be seen in Figure 3. The additional constraint can be described more formally by a conflict graph $G = (B^0, E)$, where each node corresponds to an item and an edge $(B_q^0, B_r^0) \in E$ which indicates a conflict between items B_q^0 and B_r^0. Bin j of level k cannot hold both B_q^0 and $B_r^0, \forall k \in m, \forall j \in B^k$.
Other constraints and the objective function are the same as in the case of the MLBP.

3 Integer Programming models

An integer programming (IP) problem is a mathematical model in which decision variables are restricted to integer values and those discrete decision variables are restricted by linear (in)equality constraints. The objective is to minimize/maximize a linear objective function. A mixed integer programming (MIP) model, besides using the discrete decision variables also uses continuous decision variables [9].

Considering (M)IPs is interesting because (1) many combinatorial optimization problems can be expressed as an IP or MIP model and (2) there exist powerful general purpose (M)IP solvers that can solve (M)IP models. It is important to note that this project's focus will be only on IP models.

Often times additional constraints are included in the IP models whose sole purpose is to improve the performance of a model by eliminating symmetries and consequently reducing the search space. These constraints are called the symmetry-breaking constraints.

IP solvers take the IP models that represent the required problem instance and use various techniques and algorithms to efficiently solve them. Techniques such as the Cutting-plane method or Branch-and-Bound algorithms are used with a combination of various heuristics for node selection [10]. The main idea behind the Branch-and-Bound algorithm is to branch on specific values of decision variables reducing the search space [11]. This is particularly important since the amount of Branch-and-Bound nodes that it takes to come up with an optimal solution tells us something about how strong a model is. In general less Branch-and-Bound nodes indicates that the model is stronger and therefore should theoretically be faster.

For a more detailed introduction to the IP, we refer the reader to the Integer Programming and Combinatorial Optimization [12] textbook.

3.1 BP IP model

An example of an IP model for the BP problem is as follows:

Decision variable $y_j \in \{0, 1\}, \forall j \in B$, where B is the set of all the bins, is used to indicate if bin j has been used, that is $y_j = 1$ if so and $y_j = 0$ otherwise. Similarly decision variable $x_{ij} \in \{0, 1\}$ equals to 1 if item $i \in I$, where I is a set of items, is inserted into bin j. The objective function is

$$\min \sum_{j \in B} y_j \tag{1}$$

where the amount of bins used is minimised.

The first constraint that needs to be enforced is the capacity constraint,

$$\sum_{i \in I} x_{ij} \cdot s(i) \leq y_j \cdot w(B_j)$$
$$\forall j \in B \tag{2}$$

that is, the sum of the sizes of all the items in bin j, needs to be less than or equal to the capacity of bin j for all $j \in B$. Second, Constraint 3 ensures that each item is used exactly once.

$$\sum_{j \in B} x_{ij} = 1, \forall i \in I \tag{3}$$

Additionally, for the BP a symmetry-breaking constraint which enforces a partial ordering of the bins,

$$y_j \geq y_{(j+1)}, \forall j \in B \tag{4}$$

can be added. Since all the bins have the same capacity, this constraint prunes symmetric solutions, where solutions are different but have the same objective value score. For example consider $y_a = \{0, 1, 1\}$ and $y_b = \{1, 1, 0\}$. The two solutions are different, $y_a \neq y_b$, however they have the same score, of 2. Adding the symmetry-breaking constraint prunes y_a, forcing our model to only consider y_b.

3.2 MLBP IP model 1

The first approach of representing MLBP as an IP model is based on the IP representation of the BP. The decision variables are all kept as binary values with the addition of a whole another dimension to keep track of the levels. Decision variable $y_j^k \in \{0, 1\}$ keeps track of which bins have been used, $y_j^k = 1$ if bin j at level k has been used, $y_j^k = 0$ otherwise. Decision variable $x_{ij}^k \in \{0, 1\}$ keeps track of connections between different levels, $x_{ij}^k = 1$ if item/bin i in level $(k-1)$ is assigned to bin j at level k and $x_{ij}^k = 0$ otherwise. The objective function,

$$\min \sum_{k=1}^{m} \sum_{j \in B^k} y_j^k \cdot c(B_j^k) \tag{5}$$

is calculated by going through all the bins and adding their cost if they have been used. This is done by taking the advantage of the binary variable y where if a bin is not used, $y_j^k = 0$, then the cost of that bin, is not added to the total sum.

The main constraint that was explicitly described in the mathematical description of this problem (Section 2.2), is the capacity constraint:

$$\sum_{i \in B^{(k-1)}} x_{ij}^k \cdot s(B_i^{(k-1)}) \leq y_j^k \cdot w(B_j^k) \quad \forall k \in m, \forall j \in B^k \tag{6}$$

It ensures that the contents of any of the bins do not exceed its capacity. The next constraint is that each item must be used in all the levels. This can be split into two smaller constraints. First, each item must be inserted into the bins of level 1,

$$\sum_{i \in B^0} x_{ij}^1 = 1, \forall j \in B^1 \tag{7}$$

Second, once a bin i is used at level $(k-1)$, it also must be used at level k:

$$\sum_{i \in B^{(k-1)}} x_{ij}^k = y_i^{(k-1)}, \forall k \in m, \forall j \in B^k \tag{8}$$

These two constraints combined together inductively ensure the above constraint.

A big advantage of this model is that since the decision variable x is used to represent the connections between the bins, it is not possible for a solution to contain "item splitting". That is, if an item/bin a and an item/bin b have been put into the same bin at level k, they also have to be together in whatever bins they go to in levels $> k$.

On top of this model additional constraints can be added to improve the performance, however for the sake of research they are only added in the second MLBP IP model and are described in the following section.

3.3 MLBP IP model 2

The second IP model for MLBP is based on flow constraints and is an extension of the first IP model from Section 3.2. A solution of the MLBP can be represented as a forest, where each tree represents the packing of a top level bin. Hence, the top level bin represents the root node and child nodes represent intermediate nodes. Finally, leaf nodes represent the items. To enforce such a tree structure we introduce an additional flow variable f_{ij}^k for each variable x_{ij}^k that represent the amount of flow on the corresponding edge. More formally, f_{ij}^k represents the amount of items flowing from bin/item i of level $(k-1)$ to bin j of level k. Variable f_{ij}^k has the exact dimensions of x_{ij}^k, however, f is not strictly binary as x is. The first level of f is binary, $f_{ij}^1 \in \{0, 1\}$, forcing each item to be used. However, in all the other levels of f, all items can go to the same bin which means f can take a value of any natural number up to n (amount of items), $f_{ij}^{>1} \in \mathbb{N}_{[0,n]}$. The idea of this model is that each of the leaf nodes sends out exactly one unit of flow and the root nodes (all top level bins) consume exactly n units of flow. The incoming flow of intermediate nodes is always equal to the outgoing flow. This leads to the following additional constraints:

$$\sum_{j \in B^1} f_{ij}^1 = 1, \forall i \in B^0 \tag{9}$$

$$\sum_{j \in B^m} \sum_{i \in B^{(m-1)}} f_{ij}^m = n \tag{10}$$

$$\sum_{a \in B^{(k-1)}} f_{aj}^k = \sum_{b \in B^{(k+1)}} f_{jb}^k \quad \forall k \in [2, ..., (m-1)], \forall j \in B^k \tag{11}$$

Constraint 9 forces all items to be used by stating that the amount of flow between each item and the first level bins has to be exactly one. Next, we need to make sure the total amount of flow leaving our network is the same as the amount put in. Constraint 10 enforces this by stating that the amount of flow between the last two levels must be equal to

the amount of items in the problem instance. Additionally, we also need to make sure that the amount of flow leaving each node (excluding the start and end nodes) is exactly the same as the amount of flow that enters that node, this is imposed by the Constraint 11.

Thus far, the model contains and correctly constraints the the f variable, however, f is not "connected" to the rest of the model, meaning it does not affect the solution space represented by the model. In order to connect f and x variables so that they mimic each other, we add the following two connecting constraints:

$$f_{ij}^k \leq x_{ij}^k \cdot n$$
$$\forall k \in m, \forall i \in B^{(k-1)}, \forall j \in B^k \tag{12}$$

$$x_{ij}^k \leq f_{ij}^k$$
$$\forall k \in m, \forall i \in B^{(k-1)}, \forall j \in B^k \tag{13}$$

This allows the two decision variables to work together, making the model stronger in a sense that the solution space is tighter. On top of this, another connecting constraint can be added to theoretically further tighten the model and consequently improve the performance. In a similar way that f and x have been connected, we can also connect x and y. The constraint of:

$$x_{ij}^k \leq y_i^{(k-1)}$$
$$\forall k \in [2, m], \forall i \in B^{(k-1)}, \forall j \in B^k \tag{14}$$

ensures that as soon as we know bin i of level $(k-1)$ is not used, connections between that bin and any of the bins of the level k will not be considered, consequently reducing the search space.

Additionally, we can add a minor symmetry-breaking constraint that is based on the symmetry-breaking Constraint 4 from BP. The Constraint 15 ensures that when two bins, j and q, are in level k, and j is "better" or equal to q then j should be used first.

$$y_j^k \geq y_q^k$$
$$\forall k \in m, \forall j \in B^k, \forall q \in B^k :$$
$$s(B_j^k) \geq s(B_q^k) \wedge w(B_j^k) \geq w(B_q^k) \wedge c(B_j^k) \leq c(B_q^k) \tag{15}$$

In this case, "better" means: greater size, greater capacity and lower cost. It is worth mentioning that this constraint has a much smaller effect than the one in the BP. This is because in BP all bins are the same and in MLBP instances it is rare to find bins that are strictly better than or as good as each other.

The objective function is exactly the same as in the first model, Equation 5.

3.4 MLBPCC IP models

Since MLBPCC is a variant of MLBP and most constraints are the same, both of the MLBPCC IP models are based on the two previous MLBP IP models. As mentioned before the main difference between MLBP and MLBPCC is the conflicting items constraint (see Section 2.3). In order to satisfy this constraint another decision variable, z is needed, where $z_{aj}^k \in \{0, 1\}$ is a binary variable overseeing whether an item a is in a bin j of level k. This variable needs to be populated for the whole model, which is done inductively with two constraints. First, z_{aj}^1 tracks every connection between items and bins in the first level, x_{aj}^1, by adhering to the Constraint 16.

$$x_{aj}^1 = z_{aj}^1, \forall a \in B^0, \forall j \in B^1 \tag{16}$$

This makes sure that if an item a goes to bin j ($x_{aj}^1 = 1$), then item a is in bin j ($z_{aj}^1 = 1$). Second, if an item a is in a bin i of level $(k-1)$ and there is a connection between that bin and some bin j of level k, then item a also needs to be in the bin j of level k. This is done in the Constraint 17, which under the hood is transformed to a linear constraint by CPLEX.

$$(z_{ai}^{(k-1)} \wedge x_{ij}^k) \rightarrow z_{aj}^k$$
$$\forall k \in [2, m], \forall i \in B^{(k-1)}, \forall j \in B^k, \forall a \in B^0 \tag{17}$$

Once all z is populated, the actual conflict constraint, which ensures that no conflicting items end up in any of the same bins, can be added. This constraint,

$$z_{aj}^k + z_{bj}^k \leq y_j^k$$
$$\forall k \in m, \forall j \in B^k, \forall a \in B^0, \forall b \in B^0 : \text{conf}_{ab} \tag{18}$$

means that if there is a conflict between items a and b, then both of these items cannot be in the same bin j (y_j^k, $\forall k \in m$, $\forall j \in B^k$). Also, an optimizing shortcut is applied here, if bin j is not used ($y_j^k = 0$) then none of the items can be in that bin.

The above constraints, with the constraints of the standard MLBP IP model (Section 3.2), constitute the first MLBPCC IP model. Similarly, the above constraints, with the constraints of the flow based MLBP IP model (Section 3.3), constitute the second MLBPCC IP model. It is worth mentioning that all four models have the exact same objective function, Equation 5.

4 Experimental Setup

The IP models described in the previous section have been implemented in ISO C++17 Standard using the CPLEX Optimization Studio 22.1.0 environment and the GCC 10.2.0 compiler for Windows. The experiments have been run on CPLEX's standard settings while always using only one thread on a machine with an Intel Core i7-9750H CPU with 2.60 GHz. The instances have been randomly generated using scripts that can be found in the */inst/scripts* folder in the repository, a link to which can be found in the Appendix A. Each instance has a specified amount of items and levels, however, since difficulty within those two parameters can vary widely, each problem instance with $n \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 100\}$ items and $m \in$

$\{1, 2, 3, 4, 5\}$ levels has five different randomly generated instances. This results in 275 instances for MLBP and 1100 instances for MLBPCC. The latter has four times the amount of instances because it has an additional parameter $p \in \{25, 50, 75, 90\}$ which represents the probability that two items are in a conflict with each other. All instances are located in the */inst/mlbp* and */inst/mlbpcc* folders in the repository, a link to which is in the Appendix A.

5 Reproducibility of Results

In order to reproduce the results or to verify the set up, we provide brief instructions on how to run the source code and explanations for some implementation choices. The solver can be run through the terminal by changing the directory to the root project directory and by running the following command: *"./x64/Debug/MLBP.exe ifile inst/mlbp/n0030_m03__000.inst prob MLBP ttime 0 threads 1"*. The first string is the location of the project executable, the string after "ifile" is the specific problem instance that is to be run, the string after "prob" is the problem that is to be solved (BP, MLBP or MLBPCC) and finally that last two digits represent the time limit in seconds and amount of threads to be used. After all the computations the results are shown with the best objective value found and the solution, as shown in Figure 4. The solution is displayed in two formats, the first one showing a 2-dimensional array sol where sol_j^k returns the index of the bin at level k into which item j has been inserted. This allows for a quick verification of the correctness of the solution by seeing which item goes to which bin at which point. The second format is easier to visualize when composing a graph, as shown in Figure 5. It is also a 2-dimensional array sol, however here sol_j^k returns an index of a bin of level k+1 where an item/bin j of level k has been inserted into.

```
MIP: CPLEX finished.
MIP: CPLEX status: Optimal
MIP: Objective value: 4247
MIP: Lower Bound: 4247
MIP: Branch-and-Bound nodes: 0

# best solution:
best objective value:   4247
best dual bound value:  4247
optimality gap: 0%
solution item to bin:
       [[2, 0, 3, 3, 3], [1, 4, 4, 4, 4], [3, 3, 3, 3, 3]]
solution k to k+1:
       [[2, 0, 3, 3, 3], [4, -1, 1, 4], [-1, 3, -1, -1, 3]]
CPU time:       0.015
```

Figure 4: Result output after solving n0005_m03__000.inst instance of MLBP

6 Results

In order to compare the two models, two different approaches have been taken. The first approach looks into how well a model is able to solve each instance given a specified time limit. For MLBP a time limit of $t = 600$ seconds (10 minutes) has been chosen while for MLBPCC $t = 60$ seconds (1 minute). Time limit t for MLBPCC is lower than for MLBP because MLBPCC instances are harder to solve and also because the amount of variables is greater, causing the amount of total instances to be much greater (275 for MLBP and 1100 for MLBPCC). Within t the model can either optimally solve the problem instance, give a feasible solution which is not necessarily optimal, or not come up with any solution in time. The results are added up and compared for the two models in Tables 2 and 3 for MLBP and MLBPCC respectively. The comparison in the tables focuses on three different cases; "Same Count", "Better Status" and "Both Feasible, Better Score". "Same Count" counts the amount of times both models achieved the same score. "Better Status" counts the amount of times one model had a better status in the hierarchy of Optimal $>$ Feasible $>$ Unknown. The last row represents the amount of times both models found a feasible solution, but the score of one model was better than the other one.

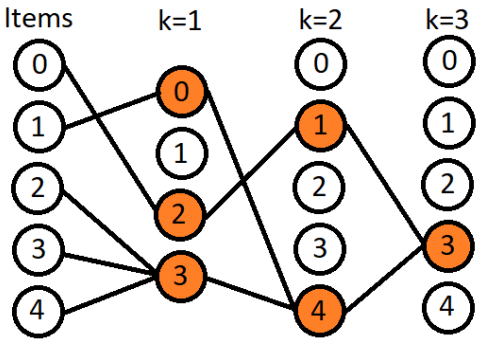

Figure 5: Visualization of the solution from Figure 4, where the used bins are colored

	MLBP model 1	MLBP model 2
Same Count	233	
Better Status	17	1
Both Feasible, Better Score	19	5

Table 2: Comparison of scores between the two MLBP models, with timeout $t = 600$ and 275 instances.

	MLBPCC model 1	MLBPCC model 2
Same Count	676	
Better Status	71	15
Both Feasible, Better Score	184	154

Table 3: Comparison of scores between the two MLBPCC models, with timeout $t = 60$ and 1100 instances.

The second approach focuses on the time taken to optimally solve the problem instances. Since MLBP(CC) are NP-Hard problems, the time needed to solve the instances of bigger sizes grows exponentially. This is why in this approach not all instances have been tested against, instead, for MLBP, the first 250 instances have been run. It is important to note that since there are five different instances for each n and m combination, we average the results from those instances to compute a single point on the graph, this is why the x-axis goes from 0-50 instead of 0-250. In or-

der to make the results consistent, instead of measuring time in seconds, we measure the performance in the amount of ticks provided by CPLEX. Where a tick is a unit used to measure work done deterministically. Since the difference in the amount of ticks it takes to solve simpler versus harder instances varies enormously and steadily increases with difficulty, the y-axis presents the amount of ticks on a logarithmic scale. The results of this approach for MLBP are shown in Figure 6. The results for MLBPCC can be seen in Figure 8. Since the complexity of MLBPCC is greater, the results had to be run on a smaller time limit t. This caused a lot of instances to provide feasible but not optimal solutions, this is why in Figure 8 we can see the amount of ticks to converge near 10^5, which is the amount of ticks the models compute in t. The occurrence of the spikes in that Figure is due to the complexity of instances not increasingly linearly but rather in a "spiky" manner. For example, the instances n5m1p75, n5m1p90, n5m2p25, n5m2p50, n5m2p75 are executed in the given order, however the order of difficulty more accurately resembles the order of: n5m2p25 $<$ n5m2p50 $\leq$ n5m1p75 $<$ n5m1p90 $\leq$ n5m2p75. It is important to note that some of the instances came with no solutions in the given t, consequently having no influence on the comparison between the models and therefore were omitted from the graphs. A brief comparison of the results can be seen in the first two rows of Tables 4 for MLBP and 5 for MLBPCC, where the amount of times one model is faster than the other is counted.

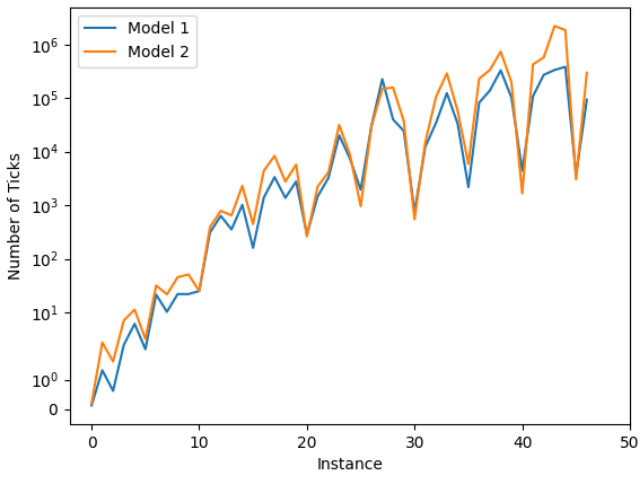

Figure 6: Visualisation of the tick comparison from Table 4 for the two MLBP models.

Additionally, we also look into the amount of the Branch-and-Bound (BnB) nodes in order to learn more about the models. As mentioned in Section 3, less BnB nodes indicates a stronger model and usually a better performance. The last two rows of Tables 4 and 5 briefly compare the amount of BnB nodes, while Figures 7 and 9 show graphs for the amount of BnB nodes for MLBP and MLBPCC respectively. Both the x and y axes are presented in the same manner as in the previous tick figures. The reason for the amount of BnB

	Model 1	Model 2
Same Ticks count	0	
Less Ticks count	208	42
Same BnB nodes count	36	
Less BnB nodes count	110	104

Table 4: Comparison of ticks and BnB nodes between the two MLBP models, with timeout $t = 600$ and the first 250 instances.

	Model 1	Model 2
Same Ticks count	359	
Less Ticks count	385	356
Same BnB nodes count	434	
Less BnB nodes count	335	331

Table 5: Comparison of ticks and BnB nodes between the two MLBPCC models, with timeout $t = 60$ and the first 1100 instances.

nodes not increasing in the MLBPCC graph is the same as mentioned before with regard to Figure 8, namely, given the complexity of the instances and the given t, the models often timeout only reaching a limited amount of progress which is shown as the amount of BnB nodes. However, because this happens for both models 1 and 2, the comparison of the two models is still valid and is explained in the next section.

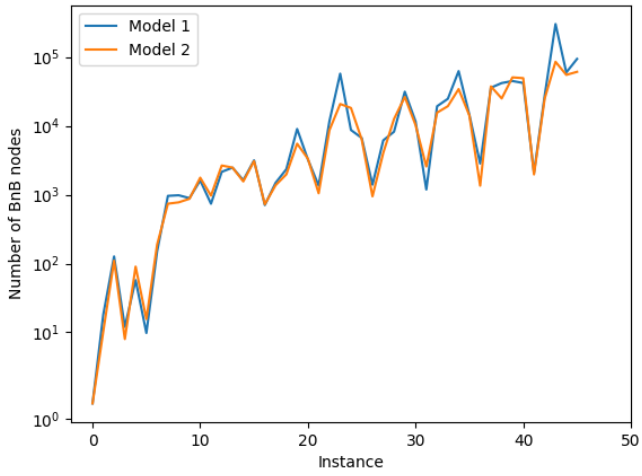

Figure 7: Visualisation of the BnB node comparison from Table 4 for the two MLBP models.

The full result text files for the graphs and tick/BnB node comparison tables are in the format "f_ticks_MLBP(CC)..." and for score comparison are in the format "f_scores_MLBP(CC)...", all of which can be found in the */evaluation* folder in the repository, a link to which can be found in the Appendix A.

7 Discussion

Our initial answer for the question of "Do flow based IP models outperform other kind of IP models to solve MLBP(CC)

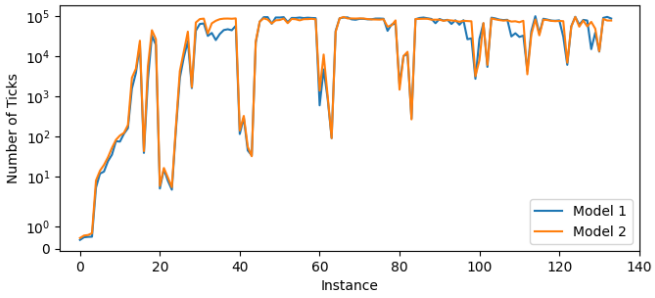

Figure 8: Visualisation of the tick comparison from Table 5 for the two MLBPCC models.

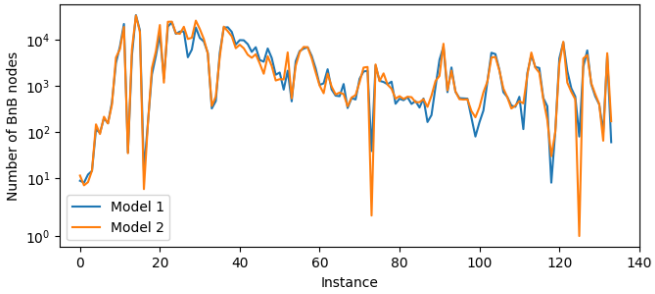

Figure 9: Visualisation of the BnB node comparison from Table 5 for the two MLBPCC models.

instances in terms of solution time and the number of branch-and-bound nodes required?", was yes. This is because, flow based IP models use additional decision variables and constraints to make the models stronger, we expect to see this in lower amount of BnB nodes and lower total computation time. However, the results from Table 2 show that model 1 was able to better solve the given instances in the given time limit than model 2. This point is further reinforced in Figure 6, where model 1 has consistently less ticks for each instance, which is further summarized in the top two rows of Table 4. Looking at the amount of BnB nodes, in Figure 7, we can see a slight advantage of model 2, however the bottom two rows of Table 4 point to inconclusiveness of this. This might be due to a fact that the effects of the stronger model start to show on more complex instances. This would mean that some of the BnB node count in the last row of Table 4 could increase rapidly on easier instances for model 1 where there is only a minor difference whereas the count for model 2 is increased on harder instances where the difference is more significant.

As mentioned previously, data from MLBPCC is not as clear as the one for MLBP due to a higher complexity and a consequent lower time limit. Despite this, we can still see in Table 3, that model 1 outperforms model 2. This is further highlighted in Figure 8, where although both models perform similarly, the amount of ticks for model 1 consistently dips in comparison to model 2; this also holds when looking at the top two rows of Table 5. When looking at the amount of BnB nodes for both models for MLBPCC, the results seem to also point towards inconclusiveness. The results in Figure 9 seem to align with the results from Figure 7, except for the x-axis range 80-120, which gives the opposite result. The inconclusiveness of the comparison of the amount of BnB nodes is further shown by the bottom two rows of Table 5, which may be due to the same point made previously with regard to the bottom two rows of Table 4 for MLBP.

Having seen the results, it is clear that our initial hypothesis was wrong. Additional flow based constraints do not necessarily outperform other kind of IP models for MLBP(CC). A possible explanation for this is that in order to make model 2 stronger, more decision variables and constraints were added. Although more variables and constraints potentially leads to less BnB nodes in the more complex instances of MLBP, the results were inconclusive for all other instances, as shown in Tables 4, 5 and Figure 9. The additional variables and constraints do not guarantee a stronger model, however they can lead to a higher computational time for a computation of each solution. This means that the extra flow based variables and constraints cause the computation of each solution to take more time, which even in a case where the solution space is smaller, the total time taken would still be longer.

8 Responsible Research

Conducting ethical and sustainable research as well as addressing those topics is filled with difficulties, yet, we believe it is extremely important to do so with regard to any research, regardless of the field. We shall discuss different measures we take on to ensure our research is both reproducible and ethical.

Reproducibility of experimental results as well as transparency of the experimental set up are essential to a healthy academia and society as a whole. Both of these relate to the Mertonian norm of 'Organized Skepticism' [13] where in order to be critical towards one's own work and to ease others' critique of one's work it is necessary for transparent experimental results and straightforward reproducibility of them. This is why we want to make sure this paper is a good foundation for future work. We therefore provide all source materials that were used during this research, that is, the source code implementation, source code for evaluation scripts, and the results achieved. The link to the repository containing all materials can be found in the Appendix A. Additionally, in Section 5 we provide instructions on how to run our project after it has been built.

This paper explores the different IP models for solving MLBP and MLBPCC. It's main focus is the mathematical reasoning, code implementation and the results of the two. We believe this research is conducted ethically with regards to the methods and conclusions reached. As mentioned in the Introduction, MLBP and MLBPCC have many real-world applications, specifically in the field of logistics. It has the potential for a lot of benefits for the human-kind such as, lower-cost and more efficient transport which consequently often times results in lower emissions, a characteristic increasingly demanded nowadays. However, we do acknowledge a possibility of using efficient MLBP techniques for packing goods for malicious purposes such as weaponry. Thus, we notice both the potential advantages and drawbacks of the use of

this research. However, it is beyond our qualifications to discuss what are the 'right' ways of using MLBP in real-world or to discuss the different regulations that could be enforced to ensure only the 'right' applications of MLBP are used. Therefore we leave such task to the competent authorities.

9 Conclusions and Future Work

In this paper, we consider two different IP models for both MLBP and MLBPCC problems to answer the question of: "Do flow based IP models outperform other kind of IP models to solve MLBP(CC) instances in terms of solution time and the number of branch-and-bound nodes required?". All four models have been carefully thought of, implemented, optimized and evaluated. The second approach models (flow based) have been built on top of the first ones with additional flow based constraints. It was predicted that the second approach models would outperform the first ones, however, we found that they are slower and that although the flow based models might be stronger, the results of the BnB node comparison between the models are inconclusive.

For future research, two main improvements can be added. First, the experiments can be run on a more powerful machine like a supercomputer. This would enable the gathering of results of bigger/more complex problem instances. This especially helps the MLBPCC problem, where there were issues in gathering the results due to exponential difficulty of the instances. Second, the flow based model can be further strengthened, for example by finding stronger bounds on the maximum flow over edges for Constraint 12. This can be done by determining such bounds level by level. For level 0, $f^1_{ij} = 1$ for all i and j, since each item must be in exactly one bin. For level 1, we have to find how many items can be packed into bin B^1_j for all $j \in B^k$. This can be done by sorting items by size and solving the Knapsack problem where we want to fit as many items as we can into bin j. For levels >1, we also solve the Knapsack problem for each bin $B^{>1}_j$, but this time it is weighted since a bin of the first level can contain multiple items. The weight of each bin of the previous level is a combination of its size and the number of maximum possible items it can hold, which is determined in the previous step. Such constraint could make the flow based model even stronger, resulting in a lower number of BnB nodes.

A Link to the repository

https://github.com/jakubtokarz/MLBP

References

[1] N. G. Kinnerseley and M. A. Langston, "Online variable-sized bin packing," *Discrete Applied Mathematics*, vol. 22, no. 2, pp. 143–148, 1988.

[2] B. H. Korte and J. Vygen, "Bin-packing," in *Combinatorial optimization: Theory and algorithms*, pp. 449, 450, Springer, 2008.

[3] U. Eliiyi and D. Eliiyi, "Applications of bin packing models through the supply chain," *International Journal of Business and Management Studies*, vol. 1, pp. 11–19, 01 2009.

[4] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for multidimensional bin packing: A survey," *Computer Science Review*, vol. 24, p. 63–79, 2017.

[5] T. cker Chiueh and A. Raniwala, "Load-balancing routing problem," in *Architecture and protocols for a high-performance, secure ieee 802.11-based wireless mesh network*, p. 20, State University of New York at Stony Brook, 2009.

[6] L. Chen, X. Tong, M. Yuan, J. Zeng, and L. Chen, "A data-driven approach for multi-level packing problems in manufacturing industry," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[7] J. Broek, van den, *MIP-based approaches for complex planning problems*. PhD thesis, Mathematics and Computer Science, 2009. Proefschrift.

[8] "What can ibm cplex optimizer do for business." https://www.ibm.com/nl-en/analytics/cplex-optimizer.

[9] D.-S. Chen, R. G. Batson, and Y. Dang, "Introduction, modeling and models," in *Applied integer programming: Modeling and solution*, Wiley, 2010.

[10] "Mixed-integer programming (mip) - a primer on the basics." https://www.gurobi.com/resource/mip-basics/, Jan 2022.

[11] B. W. Taylor, "Integer programming: The branch and bound method," in *Introduction to management science*, Pearson Education Limited, 2019.

[12] G. L. Nemhauser and L. A. Wolsey, "Integer and combinatorial optimization," John Wiley & Sons, 1988.

[13] R. K. Merton and N. W. Storer, "The normative structure of science," in *The Sociology of Science: Theoretical and Empirical Investigations*, University of Chicago Press, 1998.