Solving the Multi-Level Bin Packing Problem with Time Windows using Integer Programming

Max Le Blansch
Supervisor(s): Dr. Matthias Horn, Assoc. Prof. Dr. Neil Yorke-Smith
EEMCS, Delft University of Technology, The Netherlands

June 19, 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

This paper introduces and studies the Multi-Level Bin Packing Problem with Time Windows. This is a NP-hard problem with real-world applications in logistics. This problem states that items with certain sizes need to be packed inside bins without exceeding their capacity. These bins then again need to be assigned to other bins while minimizing the cost of the used bins. At the same time, each item has a time interval and there must be a common time point between all the items that are packed in the same bin that should be as early as possible. Not much research has been done regarding this problem. In this paper, we suggest multiple models that can be solved using Integer Programming and compare their performance. The results show slight differences in performance between the models.

1 Introduction

This project tackles the Multi-Level Bin Packing Problem with Time Windows (MLBPTW). Which is an extension of the Multi-Level Bin Packing Problem (MLBP) which in its turn is extension of the Bin Packing Problem (BP). The BP problem aims to pack a set of item with certain sizes into a set bins with a given capacity while minimizing the cost of the used bins.

The MLBP problem adds multiple levels of bins, where each bin must be packed into a bin of one level higher until the top level is reached while still minimizing the cost of the used bins. A real-life application of this problem could be the assignment of certain products that must be packed with other products in boxes which in their turn must be packed with other boxes in shipping containers while using the least amount of boxes and containers.

The MLBPTW problem adds a time window constraint to each item such that there needs to be an intersection of the time windows of all the items that are packed together in a bin, while minimizing the cost of the used bins and packing the items the earliest possible in their given time windows.

Being able to find an optimal solution in a reasonable time can make a big difference when applied where reducing packaging cost are a big part of the the total cost of a product. For example, large distribution centers that transport trucks filled with different size of boxes filled with products that need to be packed during certain time slots, when packing a product requires availability of a certain machine for example, could lower their packaging costs when the resulting model can find solutions to sufficiently large problem instances. This could mean a lower usage of packing materials and trucks which is economically beneficial for the business and more sustainable.

The work of Ongarj et al. [11] are solving the Bin Packing Problem with Time Windows (BPTW) as a sub-problem in their case study to reduce transportation cost and scheduling time of a company. Their study manages to reduce the transportation cost and scheduling time by 23% and 50% respectively. However, it is not clear what part of this reduction came from solving the BPTW problem as it was a smaller part of their study.

The MLBP problem has been investigated far less then the Bin Packing problem (BP), which has also been considered with several additional constraints. Chen et al. [2] have solved MLBP in a real-life scenario. Their solution uses a dynamic programming approach and a fuzzy-matching algorithm where the latter is able to solve larger problem sizes. However, they only consider dynamic programming and fuzzy-matching when finding solutions

to instances of MLBP.

Integer Programming (IP) is a promising alternative approach of finding optimal solutions to the problem. In short, IP is a mathematical optimization algorithm with given linear inequality constraints where all the variables are restricted to be integers and the goals is to maximize or minimize an objective function. The fact that several papers won the Edelman prize while using IP proves its success, as concluded by Johnson et al. [8].

The goal of this research is to come up with IP models to solve the MLBP and MLBPTW problems and analyse and compare the performance for different problem instances. We formulate it in the following research questions: Which IP models perform best when finding an optimal solution for MLBP and MLBPTW for different problem instances?

The remainder of the paper is organized as follows. In Section 2, the problem is mathematically defined, our method of coming up with IP models for the different problems is described and IP is explained in more detail. Section 3 summarizes related work in order to provide context on the BP problem and its variations and give an overview of the relevant takeaways from the research that is useful to our research. Next, we define, in Section 4, the IP models that are implemented in CPLEX after which the results are reported in Section 5. Section 6 reflects upon the ethical aspects and the reproducability of the research. The results are discussed in Section 7 after which the paper is summarized in Section 8.

2 Problem Definition and Method

2.1 Problem Definition

MLBP can be defined as follows. We are given a set of items $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_{n^0}\}$ that each have an integer size $s(\mathcal{I}_j) \in \mathbf{N}_{>0}$ where $1 \leq j \leq n^0$ and a set of bins $\mathcal{B}^k = \{\mathcal{B}^k_1, \mathcal{B}^k_2, ..., \mathcal{B}^k_{n^k}\}$ that each have their own integer size, capacity and cost such that $s(\mathcal{B}^k_i), w(\mathcal{B}^k_i), c(\mathcal{B}^k_i) \in \mathbf{N}_{>0}$ respectively. The bins are divided into m levels such that $1 \leq k \leq m$ and where each level can have a different amount of bins n^k such that $1 \leq i \leq n^k$.

Each item needs to be put into a bin of level one and each used bin needs bin of level k needs to be put into a bin of level $k+1$. For all the bins of level k, the sum of the sizes of the bins/items of level $k-1$ that have been put in that bin cannot exceed its capacity. The goal is to minimize the sum of the cost of all the used bins.

The extended problem with time windows introduces a given time window $t_j = [e_j, l_j]$ for each item $\mathcal{I}_j$ where e_j and l_j represent the earliest and latest packing time of item $\mathcal{I}_j$ respectively and a penalty factor p. Each item $\mathcal{I}_j$ needs to be pack as early as possible within its time window and two items can only be put into the same bin if their time windows overlap. The new goal adds a penalty value $(u(\mathcal{I}_j) - e_j)p$ to the sum of the cost of all the used bins for each item $\mathcal{I}_j$ where $u(\mathcal{I}_j)$ represents the earliest time item I_j is put into a bin.

The MLBPTW problem is strongly NP-hard when the amount of levels $m = 1$ and for all item pairs $\mathcal{I}_x, \mathcal{I}_y \in \mathcal{I}$ both $e_x = e_y$ and $l_x = l_y$ hold, since it then boils down to the BP problem, which is a strongly NP-hard problem as shown by Martello et al. [9].

2.2 Method

As stated in the Introduction, we want to answer the following question: Which IP models perform best when finding an optimal solution for MLBP and MLBPTW for different problem instances? In order to do so, we first need to know how IP works in finding solutions to NP-hard problems.

2.2.1 Integer Programming

As stated in the Introduction, IP is a mathematical optimization algorithm with given linear inequality constraints where all the variables are restricted to be integers and the goals is to minimize an objective function.

The main idea to solve an Integer Program is called Branch-&-Bound. This idea is based on dividing the problem into subproblems, which is called branching, and then finding a lower bound for those subproblems using Linear Program-Relaxations (LP-Relaxations).

An LP-Relaxation is the IP problem without the integer constraint of the variables. This LP-Relaxation is now a continuous problem which can easily be solved, for example by using the Simplex algorithm, as shown by Nash et al. [10] which gives a lower bound for the subproblem.

The algorithm then iterates over the subproblems and tries to prune which is possible in the following three cases. If the solution to its LP-Relaxtion is integral, its an optimal solution for that subproblem which is called a primal solution. In that case, the subproblem does not need to be divided into further subproblems. When the LP-Relaxation is infeasable, the subproblem itself is also infeasable and can therefore be pruned. When the lower-bound given by the solution of the LP-Relaxation is larger than the best primal solution found in any subproblem so far, it can also be pruned. If none of these situations occur, the subproblem is divided into further subproblems.

For a more detailed introduction to IP we refer to the textbook of Wolsey et al. [14].

Problems like the ones considered in this paper, need to be modeled by linear (in)equalities in order to be solved using IP. However, the same problem can be modeled in a lot of different ways. We define model A to be tighter than model B if the solution space of the (in)equalities of model A is a subset of the one of model B.

2.2.2 CPLEX

"CPLEX is a tool for solving, first of all, linear optimization problems" according to the CPLEX User's Manual [13] which can also solve extensions to those problems where any or all of the variables are constrained to integer values. CPLEX will be used in this paper to solve the IP models that will be explained in Section 4. These equations stated in those models can be directly programmed into CPLEX after which it will use IP to try to find the optimal solution for the model while using several methods to improve performance, their implementation can be found in the `*formulation.cpp` files in the GitHub repository [1].

One of these methods is called preprocessing. This performance optimization stage is executed before the IP algorithm. During this stage, CPLEX will simplify constraints to reduce problem size and eliminate redundancy. We refer to the CPLEX User's Manual [13] for more details about this optimization stage. This manual states that, "For most models, preprocessing is beneficial to the total solution speed, and CPLEX reports the

model's solution in terms of the user's original formulation, making the exact nature of any reductions immaterial" (p. 141).

As a user of the program, however, it's unclear how the model is simplified internally by CPLEX. This makes it harder to reason about the difference in performance between different models. Fortunately, we can use the number of Branch-&-Bound nodes as a metric to compare the strength of different models. When CPLEX algorithm finishes running, it displays how many nodes were created during the execution of this algorithm, where each node represents the subproblems while executing the IP algorithm as explained in the Integer Programming Section. The lower the amount of nodes needed to find a solution for a model, the stronger that model is. This means the model needs less iterations of the IP algorithm, even though it might have a longer execution time.

3 Related Research

The MLBP problem has barely been considered in research so far, apart from a practical application in the paper by Chen et al. [2] as mentioned in the Introduction.

However, the BP problem and its variations have been researched far more extensively. In the research of Coffman et al. [3] and Friesen et al. [7], different approximation algorithms for the BP problem and its variations are analyzed based on their worst-case perfomance. However, Coffman et al. [3] also classify the algorithms based on the type of problem, algorithm class, result complexity and parameter limitations. The classes they are assigned to provide an overview of results of other research on the BP problem and their corresponding algorithms such that they can be compared based on their complexity and worst-case performance.

Both papers devote their attention to variants of the one dimensional BP problem. However, no variation similar to time windows is considered. This research mainly serves as context to what approximation algorithms for problems similar the BP problem already have been proposed and their performance.

Our paper will focus on the performance of CPLEX using IP to find a exact optimal solution.

The work of Dell'Amico et al. [6] and proposes a algorithm that uses IP to solve a variation of the BP problem where every item consumes the capacity of a bin during a give time window after which the capacity of the bin that was consumed by the item can be reused. They introduce a IP model with polynomial-size and one with exponential-size. The exponential-sized cannot be explicitly enumerated for all instances of the problem. Therefore, CPLEX cannot be used to find an optimal solution. In the paper by Dell'Amico et al. [6] it is solved by introducing a branch-and-price framework.

However, the polynomial-sized model can be explicitly written and can therefore be solved using CPLEX without any additional framework. In the context of our research, the model needs to be adjusted since the MLBPTW problem stated in the Problem Definition does not allow the capacity of a bin that was consumed by an item to be used after the time window of that item has ended.

The Variable-Sized Bin Packing Problem with Time Windows (VSBPTW) is the variation of the BP problem that is the most similar to the MLBPTW problem. This problem

states that, given a set of items with different sizes and time windows and a set of bins with different costs and capacities, the sum of the cost of the used bins to pack the problem must be minimized while all the items within every used bin must have a common point within their time window.

The work of Qiang Liu et al. [12] proposes a Integer Programming model for a problem very similar to the VSBPTW problem almost the same problem and attempts to solve it using CPLEX in order to test the difficulties of the problem. The only difference is that the problem they are solving does not use a specifically defined set of bins but considers a few different types of bins with a given capacity and cost where a unlimited amount of bins of each type can be used to find a solution to the problem. They compare the results of the CPLEX solver and other heuristics for several problem instances of varying sizes where the cost of the optimal solution is known. The Integer Programming model they propose can be adjusted to represent the VSBPTW problem as described in the previous paragraph which in its turn can be extended to become the MLBPTW problem.

In their work they also mention that their VSBPTW problem can be transformed in the similar Variable-Sized Bin Packing Problem with Conflicts (VSBPC) where certain pairs of items are not allowed to be assigned to the same bin. In the case of the MLBPTW or VSBPTW problem this would conflict would be defined as pairs of items of which the time intervals do not intersect. However, this transformation cannot be solved in the same way as one of the Bin Packing problems with conflicts since the rules regarding time windows in the objective function and when packing more than two items in the same bin.

4 IP Models for the MLBP(TW) problem

Before defining a IP model to solve the MLBPTW problem, first models for the MLBP problem will be defined and tested using CPLEX. When the models correct such that CPLEX is able to find a optimal solution, we will define the model to solve the MLPBTW problem by extending the models.

4.1 Compact MLBP model

First, we define the a model to solve the MLBP problem that is as compact as possible. To do so, we define two sets of binary decision variables $x_{k,i,j}$ and $y_{k,i}$. When a bin with index i at level k, or item with index i when $k = 0$, is assigned to bin with index j at level $k + 1$, $x_{k,i,j}$ is set to 1 and 0 otherwise. When a bin with index i at level k is used, $y_{k,i}$ is set to 1 and 0 otherwise. Using these decision variables, the compact model used to solve the MLBP problem can be expressed as follows:

$$\min \sum_{k=1}^{m} \sum_{i \in \mathcal{B}^k} y_{k,i} * c(\mathcal{B}_i^k) \tag{1}$$

$$\text{s.t.} \sum_{j \in \mathcal{B}^1} x_{0,i,j} = 1 \qquad i \in \mathcal{I} \tag{2}$$

$$x_{k,i,j} \leq y_{k,i} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{B}^k,\ j \in \mathcal{B}^{k+1} \tag{3}$$

$$\sum_{j \in B^{k+1}} x_{k,i,j} = y_{k,i} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{B}^k \tag{4}$$

$$\sum_{i \in B^{k-1}} x_{k-1,i,j} * s(\mathcal{B}_i^{k-1}) \leq y_{k,j} * w(\mathcal{B}_j^k) \qquad k = 1, \ldots, m,\ j \in \mathcal{B}^k \tag{5}$$

The objective function 1 minimizes the sum of the cost of all the used bins. Constraint 2 enforces that every item needs to be packed into a bin of level one. Constraint 3 ensures each bin can only be assigned to a bin of the next level when it is used and 4 states that every used bin must be assigned to exactly one bin of the next level when it is used. Constraint 5 represents the limit of the capacity of each used bin (note that $\mathcal{B}^0 = \mathcal{I}$).

4.2 Network FLow MLBP model

As an attempt to make this model stronger, we propose a second model to solve the MLBP problem that makes use of network flow. Usually, these models are used in traffic engineering of communication networks. In our case, we can represent MLBP problems as networks where each node represents a item or bin and the flow between nodes represents the amount of items that are assigned via that edge. For example, when a bin has 3 items assigned to it, the flow between that bin and the bin of the next level it's assigned to should be 3. This formulation will add more constraints between nodes which will help make the model stronger. For more details about the working of network flow models and examples that can be implemented and solved in CPLEX we refer to chapter 4 of the book by Deep et al. [4].

To implement this model we extend the Compact MLBP model by adding a set of integer decision variables $f_{k,i,j}$ such that for each variable $0 \leq f \leq n^0$ where each variable represents the amount of items that flow from item/bin with index i at level k to bin with index j at level $k+1$. We add on the following constraints to the objective function and constraints stated in Equations 1 to 5:

$$x_{k,i,j} \leq f_{k,i,j} \qquad k = 0, \ldots, m-1,\ i \in \mathcal{B}^k,\ j \in \mathcal{B}^{k+1} \tag{6}$$

$$f_{k,i,j} \leq x_{k,i,j} * n^0 \qquad k = 0, \ldots, m-1,\ i \in \mathcal{B}^k,\ j \in \mathcal{B}^{k+1} \tag{7}$$

$$\sum_{j \in \mathcal{B}^1} f_{0,i,j} = 1 \qquad i \in \mathcal{I} \tag{8}$$

$$\sum_{i \in \mathcal{B}^{m-1}} \sum_{j \in \mathcal{B}^m} f_{m-1,i,j} = n^0 \tag{9}$$

$$\sum_{l \in \mathcal{B}^{k-1}} f_{k-1,l,i} = \sum_{j \in \mathcal{B}^{k+1}} f_{k,i,j} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{B}^k \tag{10}$$

Constraint 6 enforces that there is flow between two item/bins if the item/bin is assigned to the bin of the next level and 7 makes sure there can only be flow between those item/bins. Constraint 8 ensures every item provides one unit of flow to the network and 9 makes the amount of flow the top level bins together consume equal to the amount of items. Finally, Constraint 10 states that every bin apart from the top level bins needs to have same in- as outflow.

4.3 Compact MLBPTW model

In order to come up with the model used to solve the MLBPTW problem, Qiang et al. [12] propose a decision variable for each bin that represents its latest delivery time. Using a similar approach would mean we would have to multiply decision variables to calculate the value of the objective function. This would make our model a quadratic problem instead of a linear program which is out of scope of this paper. Instead, we introduce a set of integer decision variables u_i that represents the earliest packing time of $i \in \mathcal{I}$ such that $e_i \leq u_i \leq l_i$ and a set of binary decision variables $z_{k,i,j}$. When an item with index i or a bin to which that item is assigned is assigned to a bin with index j at level k, $z_{k,i,j}$ is set to 1 and 0 otherwise. Again, we extend the Compact MLBP model with an adjusted objective function and the following additional constraints:

$$\min \sum_{k=1}^{m} \sum_{i \in \mathcal{B}^k} y_{k,i} * c(\mathcal{B}_i^k) + \sum_{i \in \mathcal{I}} p * (u_i - e_i) \tag{11}$$

$$\text{s.t.} \sum_{j \in \mathcal{B}^k} z_{k,i,j} = 1 \qquad k = 1, \ldots, m,\ i \in \mathcal{I} \tag{12}$$

$$z_{1,i,j} \geq x_{0,i,j} \qquad i \in \mathcal{I},\ j \in \mathcal{B}^1 \tag{13}$$

$$z_{k,i,j} + x_{k,i,j} \leq 1 + z_{k+1,i,l} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{I},\ j \in \mathcal{B}^k,\ l \in \mathcal{B}^{k+1} \tag{14}$$

$$\text{iloIfThen}(z_{m,a,t} >= 0.5, z_{m,b,t} < 0.5) \qquad t \in \mathcal{B}^m,\ a, b \in \mathcal{I},\ a \neq b,\ e_a > l_b \vee l_a < e_b \tag{15}$$

$$\text{iloIfThen}(z_{m,a,t} >= 0.5 \wedge z_{m,b,t} >= 0.5, u_a \geq u_b) \qquad t \in \mathcal{B}^m,\ a, b \in \mathcal{I},\ a \neq b, (e_a > l_b \vee l_a < e_b),\ e_a < e_b \tag{16}$$

The objective function 1 minimizes the sum of the cost of all the used bins and the lateness of each item multiplied with the penalty value p. Constraint 12 makes sure that all items are assigned to a bin at each level and 13 enforces $z_{1,i,j} = x_{0,i,j}\ \forall i \in \mathcal{I},\ j \in \mathcal{B}^1$. Constraint 29 states that an item with index i is assigned to bin with index l at level $k+1$, if the item is packed in a bin with j at level k that is assigned to that bin at level $k+1$. Constraint 15 forbids a pair of items that do not have an overlapping interval to be assigned to the same top level bin and 16 ensures that the earliest packing time is not before the latest starting time of between the pair of items. Here, the iloIfThen(`condition`, `constraint`) is a constraint that is available in CPLEX which acts as a big-M constraint but is optimized

internally by CPLEX. Using this constraint means that the `constraint` is only active when the `condition` holds.

4.4 Network FLow MLBPTW model

Finally we can combine all the constraints and decision variables of all the models mentioned above with the objective function of the Compact MLBPTW model to define a stronger model for the MLBPTW problem. You can find the overview of this model in Appendix A.

5 Experiments

The models explained above were all implemented in the C++ framework for ILOG CPLEX 20.1.0 using the ISO C++17 Standard and compiled using GCC 10.2.0. The experiments were all executed on the DelftBlue Supercomputer [5] where each computing node used for solving one problem instance has two Intel XEON E5-6248R 24C 3.0GHz processors, with 192 GB of memory. Every experiment has a maximum time of 900 seconds after which the algorithm returns the best solution it found so far if it managed to find any. Each problem instance used can be found in the GitHub repository [1].

When comparing computation time of experiments, we use the amount of ticks instead of the CPU time. The reason for this is that the number of ticks remains the same regardless of the specifications of the CPU performing the experiment, since it is independent of other tasks the CPU might be performing during execution.

5.1 Observations

5.1.1 Models for the MLBP problem

When we compare the results of the compact model and network flow model of the MLBP problem we can see that they are both equally able to find optimal solutions. We do see a difference when comparing the optimality gap, the network flow model has a larger optimality gap for larger instance sizes for instances considering two to five levels. Regarding the number of Branch-&-Bound nodes, we can see that the network flow model has less nodes for the larger instances that have more than one level while the compact model has less nodes for the smaller instances that have more than one level. Regarding the computation time there is no clear model performing better than the other.

We conclude that the compact model performs better than the network flow model in terms of optimality gap meaning it finds a better feasible result when its unable to find a optimal result. The network flow model is a stronger model in its turn as it has fewer Branch-&-Bound nodes in general meaning it makes better cuts when dividing the problem in subproblems as explained in Section 2.2.1. For more insights, the graphs of all the instances for different levels are available in the GitHub repository [1].

5.1.2 Models for the MLBPTW problem

When comparing the results of the experiments for the compact model and network flow model of the MLBPTW problem we see that both method alternate in being able to find better solutions for instances with more than fifteen items. Both models are unable to find optimal solutions for instances with more than ten items and more than one levels and have

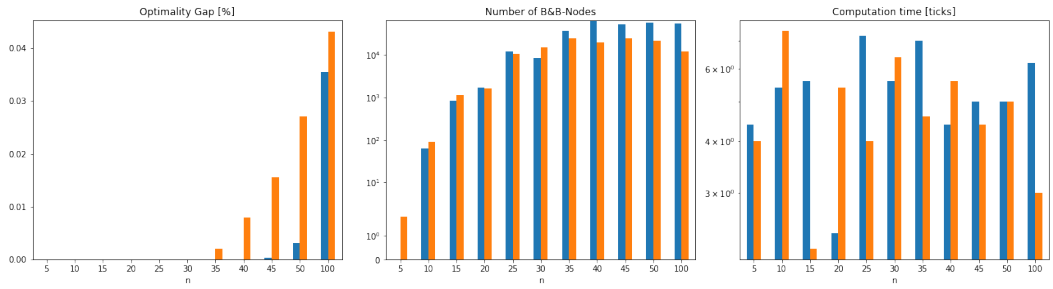

Figure 1: Optimality gap, Number of Branch-&-Bound nodes and Computation time over the number of items n of the compact model (blue) and network flow model (orange) averaged over five instances for the MLBP problem with four levels.

a hard time finding feasible solutions. The instances for which they are able to find solutions they perform similarly. However, we do see that the network flow model again has a lower number of Branch-&-Bound nodes, also for the instances in which both models are unable to find a feasible solution.

Since the compact model does not consistently perform better than the network flow model in terms of optimality gap, we can only conclude the network flow model again is stronger than the compact model. For more insights, the graphs of all the instances with different levels and different maximum time window sizes can be found in the GitHub repository [1].

6 Responsible Research

As mentioned in the Introduction, being able to find (optimal) solutions to this problem can lower costs in real-life situations when considering distribution centers for example. The instance sizes considered in this research that consider a maximum of 100 items which is insignificant when compared to cargo ships that can hold more than 20.000 containers. Therefore, this research would not make any significant impact regarding real-life situation.

All the experiments and results mentioned in this paper are reproducible by anyone as CPLEX is a deterministic algorithm and all information regarding versions of CPLEX, C++ and GCC is mentioned in Section 5. Having a different CPU would not influence the results, it will only take more time to gather results when anyone who would want to reproduce the results does not have access to a supercomputer like Delft Blue, which was used during our research. All the code, instances and results can be found on the GitHub repository [1] such that there will not be different result due to slight differences in implementation if the code of the implemented models was not publicly available.

7 Discussion

The goal of this research is to come up with IP models to solve the MLBP and MLBPTW problems and analyse and compare the performance of the models for different problem instances. Ultimately, we want to know why a model might be performing better than another

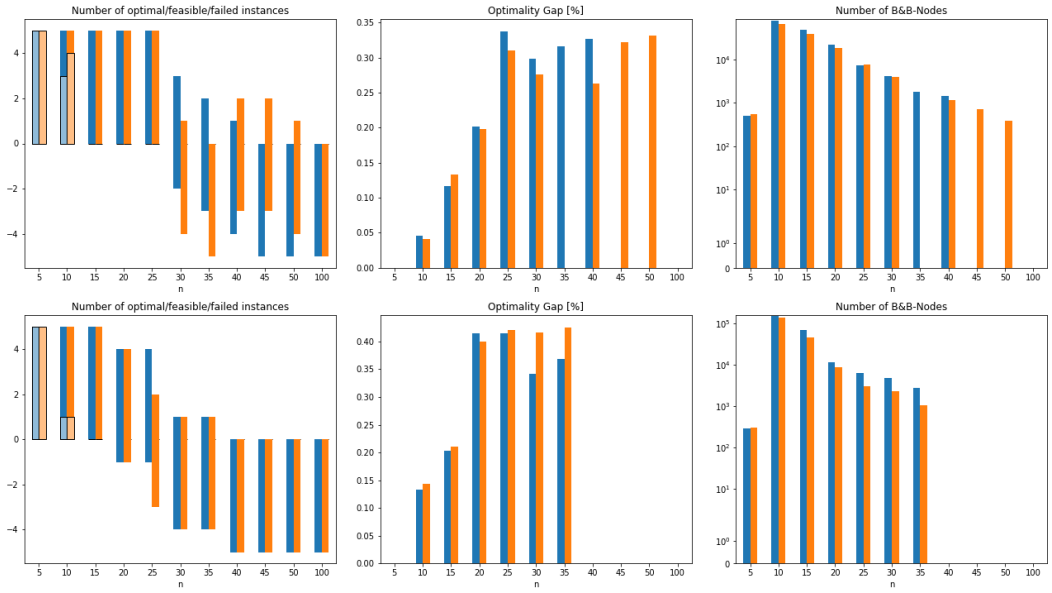

Figure 2: Number of Optimal/Feasible/Failed instances, average Optimality Gap and average Number of Branch-&-Bound Nodes over the number of items n of the compact model (blue) and network flow model (orange) averaged over five instances for the MLBPTW problem. The top row represents instances with three levels and the bottom row with four levels, all instances have a maximum time window size of ten. The grey bars represent the amount of optimal solutions found over five instances.

model. Unfortunately, understanding why different models perform better than others is highly dependent on the internal working of CPLEX. Small differences in implementation, such as the order in which the constraints are implemented, can already impact the results such that a model performs better for certain instances and worse for other instances. No major difference in the performance of the models considered in this research have been shown. A reason for this could be that the models are not inherently different but extend upon each other and therefore have no major impact on the performance.

8 Conclusions

This research aims to find models to for Integer Programming in order to answer the question to which models perform best when finding an optimal solution for the MLBP and MLBPTW problems. This paper suggests two models for both problems and compares their performance after implementing and solving them in CPLEX. The models did not show major differences in performance. However, one of them manages to find feasible solution in a given time and the other performs more efficient steps considering the Integer Programming algorithm.

Fur future work, it could be interesting let the algorithm run indefinitely for the problem instances it did not manage to find a (feasible) solution within the given time as this might highlight some bigger difference in performance between the two models.

Additionally, models that are inherently different than the ones suggested in this paper can be considered. This could produce different performance results as the models suggested in this paper extend upon each other.

The network flow models could potentially be strengthened by calculating the greatest possible flow level by level. This would basically become a recursive knapsack problem.

Regarding the MLBPTW problem, the size of an instance can potentially be reduced by eliminating possible assignments of a pair of items of which the time intervals do not intersect. This could improve performance as the models suggested in this paper did not manage to find optimal solutions for instances with more than fifteen items.

A Complete Network Flow MLBPTW model

$$\min \sum_{k=1}^{m} \sum_{i \in \mathcal{B}^k} y_{k,i} * c(\mathcal{B}_i^k) + \sum_{i \in \mathcal{I}} p * (u_i - e_i) \tag{17}$$

$$\text{s.t.} \sum_{j \in \mathcal{B}^1} x_{0,i,j} = 1 \qquad i \in \mathcal{I} \tag{18}$$

$$x_{k,i,j} \leq y_{k,i} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{B}^k,\ j \in \mathcal{B}^{k+1} \tag{19}$$

$$\sum_{j \in B^{k+1}} x_{k,i,j} = y_{k,i} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{B}^k \tag{20}$$

$$\sum_{i \in B^{k-1}} x_{k-1,i,j} * s(\mathcal{B}_i^{k-1}) \leq y_{k,j} * w(\mathcal{B}_j^k) \qquad k = 1, \ldots, m,\ j \in \mathcal{B}^k \tag{21}$$

$$x_{k,i,j} \leq f_{k,i,j} \qquad k = 0, \ldots, m-1,\ i \in \mathcal{B}^k,\ j \in \mathcal{B}^{k+1} \tag{22}$$

$$f_{k,i,j} \leq x_{k,i,j} * n^0 \qquad k = 0, \ldots, m-1,\ i \in \mathcal{B}^k,\ j \in \mathcal{B}^{k+1} \tag{23}$$

$$\sum_{j \in \mathcal{B}^1} f_{0,i,j} = 1 \qquad i \in \mathcal{I} \tag{24}$$

$$\sum_{i \in \mathcal{B}^{m-1}} \sum_{j \in \mathcal{B}^m} f_{m-1,i,j} = n^0 \tag{25}$$

$$\sum_{l \in \mathcal{B}^{k-1}} f_{k-1,l,i} = \sum_{j \in \mathcal{B}^{k+1}} f_{k,i,j} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{B}^k \tag{26}$$

$$\sum_{j \in \mathcal{B}^k} z_{k,i,j} = 1 \qquad k = 1, \ldots, m,\ i \in \mathcal{I} \tag{27}$$

$$z_{1,i,j} \geq x_{0,i,j} \qquad i \in \mathcal{I},\ j \in \mathcal{B}^1 \tag{28}$$

$$z_{k,i,j} + x_{k,i,j} \leq 1 + z_{k+1,i,l} \qquad k = 1, \ldots, m-1,\ i \in \mathcal{I},\ j \in \mathcal{B}^k,\ l \in \mathcal{B}^{k+1} \tag{29}$$

$$\text{iloIfThen}(z_{m,a,t} >= 0.5, z_{m,b,t} < 0.5) \qquad t \in \mathcal{B}^m,\ a, b \in \mathcal{I},\ a \neq b,\ e_a > l_b \vee l_a < e_b \tag{30}$$

$$\text{iloIfThen}(z_{m,a,t} >= 0.5 \wedge z_{m,b,t} >= 0.5, u_a \geq u_b) \qquad t \in \mathcal{B}^m,\ a, b \in \mathcal{I},\ a \neq b, (e_a > l_b \vee l_a < e_b),\ e_a < e_b \tag{31}$$

References

[1] Max Le Blansch. Solving the Multi-Level Bin Packing Problem with Time Windows using CPLEX, June 2022. `https://github.com/MLeBlansch/MLBPTW`.

[2] Lei Chen, Xialiang Tong, Mingxuan Yuan, Jia Zeng, and Lei Chen. A data-driven approach for multi-level packing problems in manufacturing industry. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD 2019, pages 1762–1770, Anchorage, AK, USA, August 4-8, 2019. ACM.

[3] Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. *Bin Packing Approximation Algorithms: Survey and Classification*, pages 455–531. Springer New York, 2013.

[4] Medhi Deep and Ramasamy Karthik. *Network Flow Models*, book section 4, pages 114–157. Morgan Kaufmann, Boston, second edition edition, 2018. Left at 4.6.2.

[5] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 1). `https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1`, 2022.

[6] M. Dell'Amico, F. Furini, and M. Iori. A branch-and-price algorithm for the temporal bin packing problem. *Computers and Operations Research*, 114, 2020.

[7] D. K. Friesen and M. A. Langston. Variable sized bin packing. *SIAM Journal on Computing*, 15(1):222–230, 1986.

[8] Ellis L. Johnson, George L. Nemhauser, and Martin W. P. Savelsbergh. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing*, 12(1):2–23, 2000.

[9] Silvano Martello and Paolo Toth. *Knapsack problems : algorithms and computer implementations*. Wiley-Interscience series in discrete mathematics and optimization. J. Wiley Sons, Chichester ;, 1990.

[10] J. C. Nash. The (dantzig) simplex method for linear programming. *Computing in Science & amp; Engineering*, 2(1):29–31, 2000.

[11] Lattadet Ongarj and Pornthipa Ongkunaruk. An integer programming for a bin packing problem with time windows: A case study of a thai seasoning company. In *2013 10th International Conference on Service Systems and Service Management*, pages 826–830, 2013.

[12] Liu Qiang, Cheng Huibing, Tian Tian, Wang Yongsheng, Leng Jiewu, Zhao Rongli, Zhang Hao, and Wei Lijun. Algorithms for the variable-sized bin packing problem with time windows. *Computers & Industrial Engineering*, 155:107175, 2021.

[13] IBM ILOG CPLEX Optimization Studio. CPLEX User's Manual, 2017.

[14] Laurence A. Wolsey and George L. Nemhauser. *Integer and Combinatorial Optimization*. John Wiley & Sons, Incorporated, Somerset, UNITED STATES, 1988.