An Integer Programming Approach to the Multi-Level Bin Packing Problem with Partial Orders

Can Sağtürk
Supervisors: Matthias Horn, 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

The multi-level bin packing problem (MLBP) and its variant problem with partial orders (MLBPPO) are NP-Hard problems that can be applied in a logistical setting to improve efficiency in packing. However, despite their possible use cases, there is little to no literature on how to solve these problems in a reasonable amount of time. In this work, we propose two Integer Programming (IP) models for each problem, one of which is an adaptation from an already existing bin packing implementation, and another with network flow constraints for stronger LP relaxations. The comprehensive experimentation conducted on varying sizes of problem instances suggests that the presented models for the MLBP are highly effective at solving instances with up to 50 items and 5 levels, while neither of the models outperforms the other decisively. For the MLBPPO, the model based on the BP implementation is potent at solving up to 20 items and 5 levels depending on the number of partial orders, while the network flow model cannot compete.

1 Introduction

1.1 Background Information

Every product brought to a consumer's doorstep or the local supermarket has to be packaged, put into larger boxes at the back of trucks, or possibly containers inside cargo ships. With the continued growth of online shopping [1], the logistical complexity of packing items into larger forms of storage grows, as well as the value of efficiency when it comes to packaging. All types of storage media, that we will refer to as bins, have a cost and capacity associated with them. Unnecessarily used bins and unused space inside bins incur additional unwanted costs and avoiding them requires thorough planning. These conditions are what the bin packing problem (BP) and its variations help us with.

The bin packing problem models such logistical problems, where its variants impose different attributes of the problem at hand. The standard bin packing problem deals with packing a number of items of varying sizes to boxes with the same capacity. The aim is to reduce the number of boxes used while not leaving out any items. This work concerns two variants of this problem, namely the multi-level bin packing problem (MLBP) and MLBP with partial orders (MLBPPO).

In MLBP, bins are partitioned into multiple levels. Items are packed into lower-level bins, which are packed into bins one level higher than them. This goes on until the top level, at which point all items and all used bins need to be packed into a higher-level bin. Unlike the standard bin packing problem, all bins have a cost associated with them, and the aim is to minimize the sum of the used bin's costs. Due to this reason, MLBP is an optimization problem, where we are trying to optimize our choice of bins while satisfying the constraints of the problem. With one extra constraint, MLBPPO is derived from this problem.

The MLBPPO extends MLBP by adding precedence constraints between items. If item A has precedence over item B, then the index of A's top-level bin needs to be higher than or equal to B's bin. Hence, we assume bins are ordered according to some index. As an example use case, Amazon provides priority delivery to its Prime[1] customers. MLBPPO could be used during product packaging by enforcing precedence constraints between items bought by Prime customers and other items bought by non-Prime customers.

[1] https://www.amazon.com/amazonprime

The MLBP can be multidimensional, meaning the size of an item might not be the only variable that is considered while trying to have an item fit into a bin. For example, A two-dimensional MLBP might want to consider the width and height of an item and a bin, adapting its capacity constraints accordingly. There exist one-dimensional [2], two-dimensional [3], and three-dimensional [4] variants of the bin packing problem. The MLBP that will be considered in this work is one-dimensional.

In real-life applications, optimizing packing solutions will usually have a direct effect on efficiency, for example, on the savings of a postal service or logistics company. Huawei's Noah's Ark Lab has applied the two-dimensional version of MLBP to its own packing pipeline [5]. The paper reported a 30% decrease in the workload of packing workers, which provides motivation for solving the MLBP problem that we are concerned with.

1.2 The Integer Programming Approach

Unfortunately, MLBP and MLBPPO are NP-Hard problems. Hence, using a brute-force algorithm on a reasonably-sized problem instance could take more seconds to solve the problem than the number of atoms in the universe. That means we have to solve these problems in a smarter way, by excluding large parts of the search space in a systematic way that still ensures that a proven optimal solution can be found. This brings the Integer Programming (IP) method into consideration.

Integer Programming is a way to mathematically model economics and computer science problems, commonly used on optimization problems [6]. They describe optimization problems that consist of a set of decision variables, restricted to integers, a set of linear (in)equalities, and a linear objective function. The linear (in)equalities, commonly referred to as "constraints", are what differentiates feasible and infeasible solutions [7]. Although the properties of IP can be quadratic as well, ours are always linear as they are much easier to solve. The aim is to find a solution that satisfies the constraints and, depending on the problem, maximizes or minimizes the linear objective function. Though the linear objective function always tries to minimize the sum of the used bin's cost in MLBP's case, how the decision variables and constraints represent MLBP depends on the model itself. These models are presented in Section 4 for MLBP and Section 5 for MLBPPO.

Even though we are referring to IP as a way to model MLBP throughout this paper, IP is just another kind of problem. So, to speak terminologically correct, we are actually mathematically reducing MLBP into IP. Technically, our "models" are different approaches for this reduction. It is important to note that even the fact that MLBP is reducible to IP proves that IP is NP-Hard, so we are not reducing the complexity of our problem. However, because of effective branch-and-bound methods, which are usually based on relaxing the integrality constraint, IP solvers may perform in many cases better than a naive complete enumeration of the search space. So the "Integer Programming Approach" means that we will be utilizing these solvers for MLBP by reducing MLBP to IP.

1.3 Research Questions and Contributions

This paper poses the following question: *Which IP models can solve the MLBP and MLBPPO problems with an optimal result in reasonable time on small to medium problem instances?* Emphasis on "optimal", we are looking specifically for a proven optimal solution. By reasonable time, we consider a computation time of 15 minutes. For normal queries or problems, the expectation would be a few seconds, but that is not the case for an NP-Hard problem.

It is frequent for a realistic NP-Hard problem instance to take a couple of hours to solve when approached in an inefficient manner. Small to medium problem instances, on which the IP models will be tested, contain 5 to 50 items. However, we have also exercised our models on larger instances to observe how well they scale with input.

In this paper, we are proposing two IP models each for MLBP and MLBPPO. For both of the problems, the first of these models is based on a BP model that is adapted for MLBP. The second model takes a network flow structure as its basis, though it is still formulated as an IP problem.

To answer our questions, we first explore the relatively limited literature related to our problem in Section 2. We explain our methodology in Section 3, and present our models in Sections 4 and 5. We then share our experimental setup in Section 6, the results of which are shown in Section 7. These results and their interpretations are discussed in Section 8, while their reproducibility and validity are deliberated in Section 9. We conclude our report and suggest a further research topic in Section 10.

2 Related Work

There is very little research concerning the MLBP, and none we could find for the MLBPPO. To the best of our knowledge, we are the first ones to propose IP models to the MLBP and the MLBPPO. Chen et. al. [5] developed a dynamic programming solution for the two-dimensional version of MLBP, and that is the paper closest to our topic that we could find. Their formulations cannot provide insight into our approach since they are using another method, though their results confirm the usefulness of solving the MLBP as a representation of an actual logistics problem.

IP, on the other hand, has been used on other variants of the BP problem and problems akin to BP while performing reasonably well. Some examples would be the two-dimensional cutting stock problem [8], BP with time windows [9], and three-dimensional BP [10], and there are many more works readily available that utilize IP. These problems are optimization problems with their main objective being cutting or packing. Hence, they are similar enough to the MLBP to warrant experimentation with IP models on this problem.

3 Method

We first formulated two different IP models for the MLBP which were extended in a second step to account for partial orders for the MLBPPO. The two models for the MLBP in Section 4 and for MLBPPO in Section 5 are compared in terms of computation time and number of branch-and-bound nodes required. To solve the considered IP models we used the optimization tool called IBM ILOG CPLEX, often referred to as just CPLEX.

CPLEX[2] is an optimization suite that implements an Linear Programming (LP) solver for IP problems. CPLEX supports a C++ API and its own set of interfaces that are collectively called Concert Technology, both of which allow us to implement our IP models.

To solve IP instances, CPLEX implements a Linear Program (LP)-based branch-and-bound approach. The core idea is to use a strong dual bound to prune large parts of the search space by relaxing the integrality constraints of the IP model [11]. The resulting continuous model is called LP relaxation. Strong LP relaxations can significantly speed the procedure up since CPLEX can get rid of candidate solutions without having to evaluate

[2]https://www.ibm.com/docs/en/icos/20.1.0?topic=mc-what-is-cplex

each of them one by one. CPLEX, among other algorithms, uses the Simplex algorithm to achieve these relaxations. [6].

The usage of LP relaxations entails that strong and valid inequalities should be added to obtain stronger formulations. However, more constraints in a model slow down the LP relaxation since all constraints need to be checked while solving any relaxation. This slow-down effect makes sense on paper, but we have also empirically observed this case while implementing our models. Thus the addition of constraints other than those needed for designating feasible solutions is a trade-off between larger cuts to the problem space and the time it takes to solve an LP relaxation. All of these necessitate a smart approach while modeling problem constraints. Also, extra constraints should be experimentally checked to see whether they increase or decrease performance.

Once we were done putting theoretical models into code, we were able to run the models on our problem instances in CPLEX. By measuring the time spent solving these instances that have a definite amount of items, bins, and levels, we were able to determine how performant the models were. These results are used directly to answer the research question. The experimental setup concerning the instances and CPLEX will be described in detail in its respective section.

4 Solving MLBP with IP

In this section, we will introduce our contributions in the form of mathematical models. The results achieved when the models were run on problem instances will be presented in Section 7. The discussion of the results and comparison of the models will be made in Section 8.

We will present the mathematical formulations of the two models we have developed for MLBP in this section. We will introduce the models by first showing the mathematical notation of a property, and then verbally describing its meaning. From a high-level perspective, the first model is a classic LP problem, while the second model builds on top of the first by adding network flow properties.

To be able to model MLBP, we introduce the sets that represent the properties of our problem instances. Variable $m \in \mathbb{N}_{>0}$ refers to the amount of levels. The number of bins is denoted by n_i at each level i, $1 \leq i \leq m$, and n_0 is for the amount of items. Variable B_i is the set of indexes of bins at each level i, $1 \leq i \leq m$, and B_0 is the set of indexes for items. If $j \in B_i$, then $j = \{1 \dots n_i\}$. Variable s_j^i refers to the size of the item\bin at level i, $0 \leq i \leq m$, with index j, $1 \leq j \leq n_i$. Variable w_j^i is the capacity of the bin at level i with index j, and variable c_j^i is the cost of the bin at level i with index j, where $1 \leq i \leq m$ and $1 \leq j \leq n_i$.

4.1 MLBP Model 1

The first mathematical model is an adaptation of the BP implementation that was provided to us at the start of the project. The objective function was changed, bins were introduced as variables with size, capacity, and cost, and the constraint regarding packing bins into higher-level bins was established. This model will be considered to be the baseline model, as all other models add new properties to this model rather than changing it.

To start, we have two fundamental solution variables, both of which are binary decision variables. First is variable $x_{j,k}^i \in \{0, 1\}$, where $x_{j,k}^i = 1$ if item\bin at level i with index j is packed inside a bin at level $i + 1$ with index k, $x_{j,k}^i = 0$ otherwise. Variable x decides

which item\bin is packed into which bin. Second, variable $y_j^i \in \{0, 1\}$, where $y_j^i = 1$ if the item\bin at level i with index j is used, $y_j^i = 0$ otherwise. Hence, variable y decides which bins are being used. With these established, below are the objective function constraints of this model.

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n_i} y_j^i \; c_j^i \tag{1}$$

$$\sum_{k=1}^{n_1} x_{j,k}^0 = 1 \qquad j \in B_0 \tag{2}$$

$$\sum_{k=1}^{n_{i+1}} x_{j,k}^i \iff y_j^i \qquad i \in \{1 \dots m-1\}, j \in B_i \tag{3}$$

$$\sum_{j=1}^{n_{i-1}} s_j^{i-1} \; x_{j,k}^{i-1} \leq y_k^i \; w_k^i \qquad i \in \{1 \dots m\}, k \in B_i \tag{4}$$

For the objective function (1), we want to minimize the sum of the used bins' cost. Constraint (2) ensures every item is packed into exactly one bin. Constraint (3) expresses that a bin must be packed into another bin if and only if it is being used. Note that CPLEX transforms constraint (3) into a linear constraint. Constraint (4) enforces the capacity constraint by making sure the sum of the item\bin sizes inside a bin doesn't exceed its capacity.

4.2 MLBP Model 2

The idea of the second IP model is to consider flow-based constraints to properly obtain stronger LP relaxations. To develop this different approach, we investigated a very famous and well research problem that has been solved with network flow, the Traveling Salesman Problem (TSP).

There are many different ways to solve TSP [12], including different integer programming solutions [13]. However, we will be focusing on the TSP versions of Single Commodity Flow Formulation (SCFF) [14], and Multi Commodity Flow Formulation (MCFF) [15]. Both of these formulations take the network flow approach to solving TSP with enough success to be utilized until today. We have attempted to adapt their methodology to MLBP.

In SCFF [14], Gavish and Graves extend the integer programming formulation for TSP by Miller et al. [16] by adding additional network flow constraints to eliminate sub-tours. They propose having the starting city supply n-1 flow to the graph, where n is the number of places the merchant has to visit. All other nodes (cities) can only consume one unit of flow, which represents the merchant visiting them. All nodes are connected to each other with edges that incur a cost to put any amount of flow through. The objective function is minimizing the sum of the costs of the edges.

MCFF [15] is similar to SCFF with a twist on how the flow is provided. Instead of supplying n-1 flow, the first node delivers 1 flow of n-1 different types of commodities, hence the name Multi Commodity. A commodity of type t can only be consumed by the node with that type, where all nodes consume only one unit. This method requires different edges per type of commodity per node to accommodate multiple commodities, thus the amount of variables grows with a scale of n^3. It can be seen that the number of variables would

skyrocket even in a medium-sized problem, though the idea is to compensate for that fact with stronger LP relaxations with the extra constraints added by having different types of flow.

We have used SCFF instead of MCFF due to the scaling factor of MCFF's variables. In the MLBP, all items are on the same level of the problem and one bin can have multiple items in it. This situation limits the usefulness of having multiple commodities. Thus the increase in the amount of solutions LP relaxations can eliminate by implementing MCFF does not warrant the increased time cost caused by having additional variables.

Model 2 builds directly on top of model 1 by extending it with one extra type of decision variable and three additional types of constraints. The empirical results achieved via experimentation of models 1 and 2 will determine whether these new constraints have contributed to the overall solving performance.

This model considers flow variables in addition to the decision variables from model (1)-(4). Flow variables are there to strengthen the LP relaxations. The new flow variables are $f^i_{j,k} \in \{0 \dots n_0\}$, for the flow from item\bin j at level i to a bin k at level $i+1$. The lowest possible value is $f = 0$, which represents no flow. The maximum value is the number of items, $f = n_0$, since it is equal to the flow present in the entire graph. Below are the added constraints:

$$\sum_{k=1}^{n_1} f^0_{j,k} = 1 \qquad j \in B_0 \tag{5}$$

$$(\sum_{k=1}^{n_{i-1}} f^{i-1}_{k,j}) - (\sum_{k=1}^{n_{i+1}} f^i_{j,k}) = 0 \qquad i \in \{1 \dots m-1\}, j \in B_i \tag{6}$$

$$\sum_{k=1}^{n_m} \sum_{j=1}^{n_{m-1}} f^{m-1}_{j,k} = n_0 \tag{7}$$

Constraint (5) has each item provide exactly one unit of flow to one level above and constraint (7) ensures that the flow consumed by top-level bins is equal to the number of items. Constraint (6) prevents non-top-level bins from consuming any flow.

Note that, unlike in Gavish and Graves' SCFF [14], in our model the items provide the flow and top-level bins consume it. This change is made for ease of implementation but has no impact other than a few sign switches in notation.

5 Solving MLBPPO with IP

This section presents two models for the MLBPPO and their properties by showing the mathematical notation and then explaining the meaning of these properties verbally. Both models extend the first model (1)-(4) for MLBP from Section 4, also using the decision variables x and y. As per our methodology, the first model is an IP problem representing MLBPPO, while the second model adds flow variables to be able to leverage constraints that would come from a network flow model. However, unlike in MLBP where we have used network flow only to facilitate LP relaxations, this second model enforces feasibility constraints with flow variables.

To properly formulate these models, we need to add the precedence constraint property of the problem instances to our list of variables. Hereby, O refers to the set of partial orders

among items. Given that $o \in O$, o_{first} refers to the item that has to come first at the top level, and o_{second} refers to that which comes after.

Due to the problem definition of MLBPPO, we need to enforce precedence constraints among items at the top-level. However, with the current implementation of the MLBP, we lose track of the items once we pack them to the bins at the first level. Both models for MLBPPO introduce a new type of decision variable to trace the items back to the top-level bins. Once we know which top-level bins the items are packed into, it is a matter of a single constraint to ensure the precedence constraints are met.

5.1 MLBPPO Model 1

The first MLBPPO model builds directly on top of model 1 for MLBP by adding a new decision variable. We introduce $p^h_{i,j} \in \{0, 1\}$, for each item h and each bin j at level $i + 1$, $0 \leq i < m$. These decision variables track which item is in which bin at each level. For example, $p^h_{i,j} = 1$ if item h or the bin that contains it is put into bin j at level $i + 1$.

The four new constraints are divided into two. The first three of the four constraints are concerned with mirroring the decision information in variable x to variable p from the perspective of where each item is going. The MLBP part of the MLBPPO is still fully decided by x and y variables, so p repurposes their information to trace the items. The last constraint uses the knowledge of which top-level bin each item is in to enforce the precedence relationships.

Do note that MLBPPO model 1 is an extension of MLBP model 1 with the new properties which are described in this section. Below are the new constraints.

$$\sum_{k=1}^{n_{i+1}} p^h_{i,j} = 1 \qquad i \in \{0 \dots m-1\}, h \in B_0 \tag{8}$$

$$p^j_{0,k} = x^0_{j,k} \qquad j \in B_0, k \in B_1 \tag{9}$$

$$(p^h_{i-1,j} \ \& \ x^i_{j,k}) \implies p^h_{i,k} \qquad h \in B_0, i = \{1 \dots m-1\}, j \in B_{i-1}, k \in B_i \tag{10}$$

$$\sum_{k=1}^{n_d} p^{o_{first}}_{m-1,k} \geq \sum_{k=1}^{n_d} p^{o_{second}}_{m-1,k} \qquad d \in \{0 \dots m\}, o \in O \tag{11}$$

Constraint (8) ensures that item h is traced to a single bin. Constraint (9) tracks the first bins the items are put into. Constraint (10) tracks where the bin item h is in goes to. Note that constraint (10) is transformed into a linear constraint by CPLEX. Constraint (11) enforces the precedence constraints among pairs of items.

5.2 MLBPPO Model 2

For the network flow variation of the MLBPPO, we are using the Multi Commodity Flow Formulation (MCFF). While we were choosing SCFF for the second model of the MLBP, we had found no use for having multiple different commodities representing each item. Now, however, we need to be able to identify all items at the top level. MCFF allows us to do that by providing a different edge for each item.

Variable f is also making a return from MLBP model 2 to replace variable p from the previous model. $f^i_{j,k,h} \in \{0, 1\}$, where $f = 1$ if item\bin j at level i sends flow of commodity h to bin k at level $i + 1$, where $h \in B_0$, $0 \leq i \leq m$. This time, f is also a binary decision

variable instead of being an integer denoting the amount of flow. We can do this now because there is a single flow commodity coming from each item. The flow variables are now four-dimensional to account for each edge that is added between bins for each type of commodity.

This time we have also added a sink node connected to the top-level bins. $f^m_{j,1,h}$ denotes flow of commodity type h from top-level bin j to the sink note, represented with index 1. This node is introduced to simplify summation at the top-level bins, which is used for enforcing precedence constraints. It is formulated at constraint 17 below.

Note that MLBPPO model 1 extends MLBP model 1 (1)-(4) with the new properties that are described in this section. Below are the new constraints.

$$\sum_{k=1}^{n_1} f^0_{j,k,j} = 1 \qquad j \in B_0 \tag{12}$$

$$(\sum_{k=1}^{n_{i-1}} f^{i-1}_{k,j,h}) - (\sum_{k=0}^{n_{i+1}} f^i_{j,k,h}) = 0 \qquad i = \{1 \dots m-1\}, j \in B_i, h \in B_0 \tag{13}$$

$$(\sum_{k=1}^{n_{m-1}} f^{m-1}_{k,j,h}) - f^m_{j,1,h} = 0 \qquad j \in B_m, h \in B_0 \tag{14}$$

$$\sum_{j=1}^{n_m} f^m_{j,1,h} = n_0 \qquad h \in B_0 \tag{15}$$

$$x^i_{j,k} \geq f^i_{j,k,h} \qquad i = \{0 \dots m-1\}, j \in B_i, k \in B_{i+1}, h \in B_0 \tag{16}$$

$$\sum_{j=1}^{n_d} f^m_{j,1,o_{first}} \geq \sum_{j=1}^{n_d} f^m_{j,1,o_{second}} \qquad d \in \{0 \dots m\}, o \in O \tag{17}$$

Constraint (12) has each item provide exactly one unit of the flow of its own type to one level above. Constraints (13) and (14) prevent bins from consuming any flow. Constraint (15) ensures that the sum of flow consumed by the sink node is equal to the number of items. Constraint (16) prevents unused edges from sending flow. Constraint (17) enforces the precedence constraints among pairs of items.

6 Experimental Setup

This section presents the experimental setup used while solving both MLBP and MLBPPO. Here we will mention our solver, CPLEX, and our problem instances. Finally, specifications of our running environment in TU Delft's High Performance Computing Center, DelftBlue [17], will be shared. All the models running on problem instances will be working on a single thread and they will timeout if they can't find the optimal solution in 15 minutes. To strengthen the soundness of the statistics we have collected, for each problem instance group we have run 100 different instances and averaged the results. All the code for models and the problem instances can be found in our repository[3].

[3]https://github.com/CanSagturk/mlbp-with-ip

6.1 CPLEX

We have used CPLEX's C++ API, which uses Concert Technology for object orientation, to program our models into CPLEX. The code of the models that can be found in the shared repository is thus in C++. We have used G++ version 10.2.0 to compile our code.

It should be noted that all problems are declared and treated as IP models, both by us and CPLEX. This goes even for the network flow aspects of the second model, which are modeled as linear constraints. CPLEX does offer a network flow optimizer which could improve performance in the case that the second model is implemented as an Integer Programming-Network Flow hybrid, but that is outside of the scope of this paper.

6.2 Problem Instances

All problem instances were provided to us at the start of the project. There are no infeasible instances in the problem set. Since it is possible to achieve the optimal objective value with different solutions, it is not a problem for two models to find two distinct solutions if they share the same objective value.

For the MLBP, we classify a problem instance with the number of items and levels it has. All instances will have an item amount between 10 and 100 with a step size of 10, so 10, 20...100. The levels are from 1 to 5, inclusive. For each item and level combination, there are 100 different problem instances.

For the MLBPPO, we classify a problem instance with the number of items, levels, and the ratio of partial order pairs to items as a percentage. For example, the pair ratio of an instance with 5 items and 10 pairs would be 200%. The level interval for the MLBPPO instances is the same as the setting for the MLBP. The item amount of the instances is between 10 to 50, inclusive, with a step size of 10. The pair percentages are 50%, 100%, 150%, and 200%. For each item, level, and pair combination, there are 100 different problem instances.

6.3 DelftBlue

We used the supercomputer DelftBlue to execute our experiments. We employed DelftBlue's Intel Xeon 2648R x86 compute nodes with a base CPU frequency of 3.0GHz. Even though the nodes have 48 cores each and we have run instances in parallel, this will not affect performance metrics since problem instances are atomic and are forced to run on a single thread.

7 Results

In this section, we present the results of the experiment detailed in the previous section, first for the MLBP at Table 1, and second for the MLBPPO at Table 2. The problem instances are grouped level first, item number second, and, for the MLBPPO, pair percentage third. For values of each grouping, 100 distinct problem instances were solved and their values were averaged. For all groupings, we present the performance of both models side-by-side.

As for values on the table, we have used symbols and abbreviations. Columns m and n represent the number of bin levels and the number of items, respectively. For the MLBPPO only, they are followed by column o, which is the number of precedence pairs present as a percentage of items in the problem. The rest of the values occur once for each model.

Column t represents the number of seconds it took to either solve the problem to optimality or to timeout. Since we had set the timeout as 15 minutes, the maximum value for column t is 900 seconds. Column B&B is the total number of Branch-and-Bound nodes used, including the instances that timed out. Column t\o is the percentage of instances that timed out. Column opt is the percentage optimality gap for timed-out instances. The optimality gap represents the difference between the cost of the best solution that the CPLEX could find and the optimal solution, as a percentage of the optimal value.

For the MLBPPO only, there is also the fail column. This is the percentage of problem instances for which CPLEX was not able to find a single feasible solution before timing out They don't contribute to the values of t and B&B columns. Even though they have technically timed out, they are not factored in for the optimality gap either since there was no solution found. Fails have never occurred for the MLBP.

Note that the results present in the MLBPPO table are slightly truncated. 40 and 50 item instances, which had too many failures to be reliable results, as well as the instances with 150 pair percentages were removed. The full table can be found in Appendix A at Table 3.

8 Discussion

In this section, we will interpret and reflect on the results we have achieved with the setup detailed in Section 6. Some important values in the table are colored for comparison with other values in a single row, which corresponds to an instance group. Green and blue represent the best time and B%B count, respectively, among the two models for that group. If t and B%B values for both models are red, that means both models have a timeout percentage of above 95%. If the value(s) of a single model is colored red, then the model that actually has the better value has more than 95% timed out instances. Purple is used when both models achieve the same or extremely close values, namely within an interval of 0.05 seconds for column t and 0.1 for B%B.

For the MLBP, the results are quite satisfactory. For small to medium instances, which are 10 to 50 items large, the timeouts are only noticeable for 3 to 5 level instances with 50 items, with a maximum of 24%. As the table shows, for most of the instance groups, the time required does not even come close to the time out of 900 seconds. These results demonstrate that both models that we propose in this work can successfully solve small to medium instances of the MLBP problem in a reasonable time.

For large problem instances, the required time and the number of timeouts increase significantly. However, even though it is not the focus of this paper, it is important to note that the optimality gap is very low for even the largest instances, with an average of 2.32% for 3 to 5 level instances with 60 to 100 items. This shows that our MLBP models yield a solution that is very close cost-wise to the optimum even if they time out.

The question of which of the two models is better cannot be answered from these results. The superior values for columns t and B%B are scattered between the two models and they are inconsistent among the instance groups. Their scaling with the size of the input compared to each other also does not show a pattern. The added SCFF constraints do not seem to have had a definitive improvement.

As expected of a more complex problem, the results for the MLBP are less promising. Comparing the results of similar size instances from the MLBP table and judging by the number of failures, the MLBPPO was even harder to solve than we had initially anticipated. The models significantly struggle with the time with 30 item instances, and failures surpass

Table 1: Results of the Two MLBP Models Averaged Over 100 Instances

		Model 1: IP				Model 2: IP + SCFF			
m	n	t[s]	B&B	t\o[%]	opt[%]	t[s]	B&B	t\o[%]	opt[%]
1	10	0.01	32.34	0	0.00	0.01	32.34	0	0.00
1	20	0.24	875.73	0	0.00	0.25	875.73	0	0.00
1	30	0.68	1060.25	0	0.00	0.67	1051.48	0	0.00
1	40	1.41	1329.62	0	0.00	1.42	1350.13	0	0.00
1	50	2.91	2159.61	0	0.00	2.88	2125.47	0	0.00
1	60	7.54	5699.61	0	0.00	7.49	5707.66	0	0.00
1	70	10.05	5676.65	0	0.00	10.12	5676.65	0	0.00
1	80	13.13	6396.12	0	0.00	12.29	5906.19	0	0.00
1	90	52.16	24829.10	3	0.48	51.69	24579.70	1	0.09
1	100	54.58	22063.65	0	0.00	41.15	16446.22	1	0.02
2	10	0.03	106.44	0	0.00	0.04	106.44	0	0.00
2	20	1.24	1823.16	0	0.00	1.25	1839.87	0	0.00
2	30	22.89	12172.97	2	0.91	14.34	8905.62	1	1.00
2	40	37.52	17210.97	0	0.00	39.5	18863.20	0	0.00
2	50	124.25	37630.08	1	0.10	126.5	52509.07	5	0.18
2	60	310.26	82609.12	14	0.46	253.62	68015.88	8	0.29
2	70	571.65	138667.04	44	0.57	552.77	134906.08	42	0.57
2	80	736.08	157549.56	64	0.76	695.42	170005.10	60	0.73
2	90	828.80	181810.22	83	0.99	829.32	190847.08	82	0.96
2	100	859.32	196607.27	92	1.15	871.79	203122.09	93	1.13
3	10	0.05	159.80	0	0.00	0.05	159.80	0	0.00
3	20	1.66	2159.72	0	0.00	1.70	2179.50	0	0.00
3	30	27.97	14441.65	0	0.00	28.47	16457.72	1	0.50
3	40	101.15	27393.73	1	1.24	105.33	28040.39	1	0.99
3	50	331.42	64004.61	10	0.58	288.05	60710.48	5	0.57
3	60	604.88	94503.15	47	0.93	599.18	97414.20	43	0.84
3	70	836.72	114157.31	85	1.33	811.29	105560.52	81	1.24
3	80	885.68	120771.49	96	1.68	889.38	121012.30	98	1.72
3	90	892.28	132314.36	99	2.07	897.25	128347.89	100	1.96
3	100	897.19	138546.80	100	2.38	897.20	143240.30	100	2.31
4	10	0.06	165.49	0	0.00	0.06	165.49	0	0.00
4	20	2.01	2173.49	0	0.00	2.02	2218.84	0	0.00
4	30	21.61	8578.83	0	0.00	23.11	10131.92	0	0.00
4	40	176.53	39505.21	4	0.76	145.38	30380.91	2	1.48
4	50	490.84	70144.43	24	0.97	446.81	63762.74	21	0.83
4	60	762.62	83995.49	67	1.21	704.83	78526.84	59	1.28
4	70	868.66	77030.34	92	1.91	881.35	81521.75	93	1.90
4	80	896.00	67603.30	99	2.54	897.31	69261.85	100	2.35
4	90	897.41	63755.79	100	2.80	897.37	59380.67	100	2.77
4	100	897.28	60255.04	100	3.26	897.30	62896.80	100	3.27
5	10	0.07	145.92	0	0.00	0.07	145.92	0	0.00
5	20	2.39	2454.38	0	0.00	2.34	2409.05	0	0.00
5	30	21.23	8237.09	0	0.00	19.95	8042.67	0	0.00
5	40	168.56	29524.94	5	1.41	161.64	27884.49	6	0.54
5	50	402.76	53136.89	15	1.83	424.27	55636.01	22	1.36
5	60	745.95	68187.82	66	1.52	743.65	70285.36	64	1.65
5	70	875.69	62712.13	94	2.45	868.83	62987.49	93	2.44
5	80	892.44	46862.45	97	3.21	893.23	51818.57	97	2.99
5	90	897.15	38884.77	100	3.61	897.26	41031.51	100	3.49
5	100	897.18	35430.03	100	4.16	897.28	35899.81	100	4.21

Table 2: Results of the Two MLBPPO Models Averaged Over 100 Instances

			Model 1: IP					Model 2: IP + MCFF				
m	n	o[%]	t[s]	B&B	t\o[%]	opt[%]	fail[%]	t[s]	B&B	t\o[%]	opt[%]	fail[%]
1	10	50	0.02	17.78	0	0.00	0	0.03	24.26	0	0.00	0
1	10	100	0.02	6.94	0	0.00	0	0.03	5.64	0	0.00	0
1	10	200	0.01	0.39	0	0.00	0	0.03	0.51	0	0.00	0
1	20	50	0.68	930.18	0	0.00	0	0.93	1026.61	0	0.00	0
1	20	100	0.76	1041.06	0	0.00	0	1.00	924.59	0	0.00	0
1	20	200	0.19	172.97	0	0.00	0	0.51	174.73	0	0.00	0
1	30	50	2.74	1698.51	0	0.00	0	3.43	1587.53	0	0.00	0
1	30	100	14.11	4308.66	0	0.00	0	15.11	4338.50	0	0.00	0
1	30	200	3.43	1749.69	0	0.00	0	5.04	1555.67	0	0.00	0
2	10	50	0.24	264.46	0	0.00	0	0.62	219.76	0	0.00	0
2	10	100	0.49	418.03	0	0.00	0	1.09	245.08	0	0.00	0
2	10	200	1.95	1032.33	0	0.00	0	3.31	476.85	0	0.00	0
2	20	50	52.77	3109.07	0	0.00	0	242.86	2537.33	4	0.06	0
2	20	100	341.73	12917.64	18	0.43	0	592.76	4087.99	38	1.20	0
2	20	200	817.59	22005.69	86	4.83	0	851.92	4727.64	92	7.17	0
2	30	50	667.35	8469.83	56	1.28	0	876.43	4675.61	92	4.09	0
2	30	100	830.77	8881.60	87	4.91	0	890.21	2253.99	98	7.87	0
2	30	200	897.70	7740.81	100	12.97	1	896.69	543.67	100	16.87	6
3	10	50	0.51	348.19	0	0.00	0	1.28	280.32	0	0.00	0
3	10	100	1.53	747.98	0	0.00	0	6.05	600.54	0	0.00	0
3	10	200	8.70	3436.14	0	0.00	0	28.29	1846.62	1	0.02	0
3	20	50	130.03	4669.24	0	0.00	0	568.53	2917.19	35	0.8	0
3	20	100	490.72	12885.95	33	1.17	0	779.83	2452.06	76	4.59	0
3	20	200	816.25	15514.85	81	8.05	0	891.48	1182.92	97	14.87	0
3	30	50	870.96	8164.87	93	7.53	0	896.82	1307.35	99	9.57	0
3	30	100	897.67	6621.25	100	14.40	1	897.13	379.13	100	15.61	7
3	30	200	898.02	4675.56	100	25.27	27	896.89	183.29	100	28.23	76
4	10	50	1.09	484.45	0	0.00	0	4.50	462.34	0	0.0	0
4	10	100	3.08	1134.55	0	0.00	0	17.11	814.74	0	0.00	0
4	10	200	3.16	881.48	0	0.00	0	18.46	609.57	0	0.00	0
4	20	50	171.27	6010.31	4	0.08	0	654.35	2469.88	47	1.85	0
4	20	100	460.63	10469.44	30	1.40	0	839.80	1708.54	81	7.6	0
4	20	200	728.34	13852.23	71	8.15	3	886.89	694.04	98	18.04	1
4	30	50	884.18	6185.51	96	12.20	2	890.66	528.35	99	14.3	6
4	30	100	897.95	4639.17	100	21.75	17	897.17	282.47	100	25.12	53
4	30	200	875.92	3464.78	98	30.81	59	897.06	338.29	100	21.36	93
5	10	50	1.21	442.15	0	0.0	0	6.60	459.17	0	0.0	0
5	10	100	2.78	652.64	0	0.0	0	29.69	617.03	0	0.0	0
5	10	200	13.47	3423.48	1	0.01	0	69.81	991.37	1	0.01	0
5	20	50	231.31	5996.59	6	0.23	0	758.11	1649.30	66	4.19	0
5	20	100	338.47	8510.74	12	0.59	0	842.47	1197.18	85	9.02	0
5	20	200	547.40	8691.26	35	3.66	0	890.79	452.34	98	11.84	10
5	30	50	895.20	5592.21	99	16.92	5	897.20	415.29	100	21.09	27
5	30	100	896.48	4145.59	99	22.8	18	896.91	297.61	100	22.23	72
5	30	200	893.37	3561.28	99	26.87	40	897.36	1.00	100	23.39	99

50% for most 40 and 50 item 3 to 5 level instances. Pair percentage is also a significant contributor to time values, which can increase solve times by a factor of 1.5 for most medium-size instances.

Unlike the MLBP models, there is a compelling difference between the time and B%B values of the two MLBPPO models. Model 1 is always faster, while Model 2 made fewer cuts for all cases except a few small instances. Hence, we have been successful at developing stronger constraints that lead to a significant reduction in the number of Branch-and-Bound nodes. This also proves our earlier hypothesis that a model which has stronger constraints could take longer to execute but would still make larger cuts to the search space.

Regardless of B%B values, these results establish that, in their current state, Model 1 is better than Model 2 for all possible cases. The table also shows that Model 2 scales much worse when the item amount or the number of bin levels is increased. This means that at no point will Model 2 outperform Model 1 if we continue to increase the sizes of the problem instances.

9 Responsible Research

The reproducibility of experiments is essential to guarantee the authenticity of the results. To this end, we have shared our entire framework and models, all 15000 of our problem instances, and the output files of each individual instance that we have aggregated in this work[4]. We have also explained our experimental setup in detail in Section 6, as well as clarifying what each metric is in our results in Section 7. Though it could be fiscally difficult to procure a hardware setup that can performance-wise match DelftBlue, this is made somewhat easier by limiting the execution of each problem instance to a single thread.

For every single problem group, we have solved 100 different instances. The results are the averages of the outputs of each instance, which reduces the bias a model could have towards arbitrary elements of an instance. This leads us to believe in the validity of our results.

10 Conclusions and Future Work

In this work, we have developed four Integer Programming models, two for the MLBP and two for the MLBPPO, to solve small to medium instances in under 15 minutes. The results suggest both models for MLBP were successful, without one model being decisively better than the other. Our MLBPPO models struggled with medium size problems with a higher number of levels, though they were able to solve up to 20 item instances with acceptable results. Among the MLBPPO models, the first one was convincingly better time-wise than the second one. However, the second MLBPPO model was able to make much larger cuts in the search space and reduce the amount of Branch-and-Bound nodes it had to use.

Both the second model of the MLBP and the second model of the MLBPPO have network flow properties but are implemented and solved as an IP problem. This situation warrants further research into a hybrid implementation, where network flow elements of the formulations are actually implemented as a network flow model and ideally solved with an appropriate optimizer. This looks promising especially due to the Branch-and-Bound results of the strong constraints introduced by our second model for the MLBPPO.

[4]https://github.com/CanSagturk/mlbp-with-ip

References

[1] M. Young, J. Soza-Parra, and G. Circella, "The increase in online shopping during covid-19: Who is responsible, will it last, and what does it mean for cities?", *Regional Science Policy & Practice*, 2022, Online version of the article before inclusion in a journal issue. DOI: https://doi.org/10.1111/rsp3.12514. eprint: https://rsaiconnect.onlinelibrary.wiley.com/doi/pdf/10.1111/rsp3.12514. [Online]. Available: https://rsaiconnect.onlinelibrary.wiley.com/doi/abs/10.1111/rsp3.12514 (visited on 06/18/2022).

[2] J. A. George and D. F. Robinson, "A heuristic for packing boxes into a container", *Computers & Operations Research*, vol. 7, pp. 147–156, 3 1980.

[3] N. Ma and Z. Zhou, "Mixed-integer programming model for two-dimensional non-guillotine bin packing problem with free rotation", in *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 456–460.

[4] C. Paquay, M. Schyns, and S. Limbourg, "A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application", *International Transactions in Operational Research*, vol. 23, pp. 187–213, 1-2 2016.

[5] L. Chen, X. Tong, M. Yuan, and J. Zeng, "A data-driven approach for multi-level packing problems in manufacturing industry", in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, 2019, pp. 1762–1770.

[6] R. E. Bixby, "A brief history of linear and mixed-integer programming computation", *Documenta Math. Extra Volume: Optimization Stories*, vol. ISMP, pp. 107–121, 2012.

[7] G. Nemhauser and L. Wolsey, "Linear programming", in *Integer and Combinatorial Optimization*. John Wiley Sons, Ltd, 1988, ch. I.2, pp. 27–49.

[8] E. Silva, F. Alvelos, and J. M. Valerio de Carvalho, "An integer programming model for two- and three-stage two-dimensional cutting stock problems", *European Journal of Operational Research*, vol. 205, no. 3, pp. 699–708, 2010.

[9] L. Ongarj and P. 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 (ICSSSM)*, X. Cai, J. Chen, J. Jia, Y. Xiao, and S. Zhou, Eds., 2013, pp. 826–830.

[10] S. Elhedhli, F. Gzara, and Y. F. Yan, "A mip-based slicing heuristic for three-dimensional bin packing", *Optimization Letters*, vol. 11, pp. 1547–1563, 8 2017.

[11] S. Agmon, "The relaxation method for linear inequalities", *Canadian Journal of Mathematics*, vol. 6, pp. 382–392, 1954.

[12] S. Sangwan, "Literature review on travelling salesman problem", *International Journal of Research*, vol. 5, p. 1152, 2018.

[13] A. J. Orman and H. Williams, "A survey of different integer programming formulations of the travelling salesman problem", in *Optimisation, Econometric and Financial Analysis*, C. K. E. John and Gatu, Eds., ser. AICM, Springer Berlin Heidelberg, 2007, pp. 91–104.

[14] B. Gavish and S. C. Graves, "The travelling salesman problem and related problems", *Working paper GR-078-78*, 1978.

[15] R. T. Wong, "Integer programming formulations of the traveling salesman problem", in *Proceedings of the IEEE international conference of circuits and computers*, IEEE Press Piscataway NJ, 1980, pp. 149–152.

[16] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems", *J. ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[17] D. H. P. C. C. (DHPC), *DelftBlue Supercomputer (Phase 1)*, `https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1`, 2022.

A The Non-Truncated MLBPPO Table

Table 3: Results of the Two MLBPPO Models Averaged Over 100 Instances

			Model 1: IP					Model 2: IP + MCFF				
m	n	o[%]	t[s]	B&B	t\o[%]	opt[%]	fail[%]	t[s]	B&B	t\o[%]	opt[%]	fail[%]
1	10	050	0.02	17.78	0	0.0	0	0.03	24.26	0	0.0	0
1	10	100	0.02	6.94	0	0.0	0	0.03	5.64	0	0.0	0
1	10	150	0.01	2.05	0	0.0	0	0.03	3.08	0	0.0	0
1	10	200	0.01	0.39	0	0.0	0	0.03	0.51	0	0.0	0
1	20	050	0.68	930.18	0	0.0	0	0.93	1026.61	0	0.0	0
1	20	100	0.76	1041.06	0	0.0	0	1.0	924.59	0	0.0	0
1	20	150	0.31	417.69	0	0.0	0	0.63	449.31	0	0.0	0
1	20	200	0.19	172.97	0	0.0	0	0.51	174.73	0	0.0	0
1	30	050	2.74	1698.51	0	0.0	0	3.43	1587.53	0	0.0	0
1	30	100	14.11	4308.66	0	0.0	0	15.11	4338.5	0	0.0	0
1	30	150	10.42	4176.24	0	0.0	0	11.23	3917.77	0	0.0	0
1	30	200	3.43	1749.69	0	0.0	0	5.04	1555.67	0	0.0	0
1	40	050	6.79	2898.55	0	0.0	0	7.31	2427.62	0	0.0	0
1	40	100	70.08	10287.8	1	0.0	0	66.98	9275.52	1	0.0	0
1	40	150	164.83	24293.73	4	0.02	0	161.02	23764.43	2	0.02	0
1	40	200	73.01	14736.93	0	0.0	0	85.77	15503.07	0	0.0	0
1	50	050	14.5	5181.19	0	0.0	0	19.86	4857.29	0	0.0	0
1	50	100	140.13	13461.51	5	0.01	0	162.34	13596.94	5	0.03	0
1	50	150	508.93	35035.59	34	0.22	0	489.18	32136.49	34	0.23	0
1	50	200	513.78	42005.85	30	0.29	0	532.69	40501.07	34	0.35	0
2	10	050	0.24	264.46	0	0.0	0	0.62	219.76	0	0.0	0
2	10	100	0.49	418.03	0	0.0	0	1.09	245.08	0	0.0	0
2	10	150	2.3	1248.03	0	0.0	0	2.98	497.55	0	0.0	0
2	10	200	1.95	1032.33	0	0.0	0	3.31	476.85	0	0.0	0
2	20	050	52.77	3109.07	0	0.0	0	242.86	2537.33	4	0.06	0
2	20	100	341.73	12917.64	18	0.43	0	592.76	4087.99	38	1.2	0
2	20	150	740.9	21624.21	73	2.74	0	830.25	4674.48	85	4.93	0
2	20	200	817.59	22005.69	86	4.83	0	851.92	4727.64	92	7.17	0
2	30	050	667.35	8469.83	56	1.28	0	876.43	4675.61	92	4.09	0
2	30	100	830.77	8881.6	87	4.91	0	890.21	2253.99	98	7.87	0
2	30	150	891.0	7726.0	98	9.92	0	896.88	1002.1	100	12.63	1
2	30	200	897.7	7740.81	100	12.97	1	896.69	543.67	100	16.87	6
2	40	050	891.53	8813.14	98	5.03	0	897.02	1290.16	100	6.76	1
2	40	100	897.79	6374.2	100	8.99	0	896.99	482.44	100	12.2	13
2	40	150	897.95	5475.14	100	13.31	7	897.02	408.63	100	17.97	30
2	40	200	898.13	5977.49	100	16.87	20	897.32	290.22	100	20.18	59
2	50	050	897.67	4931.93	100	7.94	0	896.71	493.25	100	8.0	9
2	50	100	897.86	2044.6	100	13.71	4	897.32	257.98	100	16.55	58
2	50	150	898.2	2024.05	100	17.31	35	897.68	166.8	100	23.32	95
2	50	200	898.54	1683.67	100	19.78	61	NaN	NaN	100	NaN	100
3	10	050	0.51	348.19	0	0.0	0	1.28	280.32	0	0.0	0
3	10	100	1.53	747.98	0	0.0	0	6.05	600.54	0	0.0	0
3	10	150	4.32	1885.49	0	0.0	0	20.22	1254.31	0	0.0	0
3	10	200	8.7	3436.14	0	0.0	0	28.29	1846.62	1	0.02	0
3	20	050	130.03	4669.24	0	0.0	0	568.53	2917.19	35	0.8	0
3	20	100	490.72	12885.95	33	1.17	0	779.83	2452.06	76	4.59	0
3	20	150	789.32	16089.57	77	5.0	0	878.5	1545.06	95	11.18	0
3	20	200	816.25	15514.85	81	8.05	0	891.48	1182.92	97	14.87	0

			Model 1: IP					Model 2: IP + MCFF				
m	n	o[%]	t[s]	B&B	t\o[%]	opt[%]	fail[%]	t[s]	B&B	t\o[%]	opt[%]	fail[%]
3	30	050	870.96	8164.87	93	7.53	0	896.82	1307.35	99	9.57	0
3	30	100	897.67	6621.25	100	14.4	1	897.13	379.13	100	15.61	7
3	30	150	897.85	5748.04	100	21.98	18	897.0	254.11	100	25.94	53
3	30	200	898.02	4675.56	100	25.27	27	896.89	183.29	100	28.23	76
3	40	050	894.35	3512.76	99	15.27	4	897.16	236.74	100	20.48	65
3	40	100	898.04	2432.63	100	21.06	16	897.03	255.5	100	12.43	98
3	40	150	898.19	1683.51	100	23.65	59	NaN	NaN	100	NaN	100
3	40	200	898.84	1173.0	100	28.51	80	NaN	NaN	100	NaN	100
3	50	050	898.15	1075.13	100	18.11	22	NaN	NaN	100	NaN	100
3	50	100	898.54	743.66	100	20.57	56	NaN	NaN	100	NaN	100
3	50	150	899.21	718.48	100	23.76	79	NaN	NaN	100	NaN	100
3	50	200	900.58	882.33	100	26.76	94	NaN	NaN	100	NaN	100
4	10	050	1.09	484.45	0	0.0	0	4.5	462.34	0	0.0	0
4	10	100	3.08	1134.55	0	0.0	0	17.11	814.74	0	0.0	0
4	10	150	2.7	778.92	0	0.0	0	18.69	640.13	0	0.0	0
4	10	200	3.16	881.48	0	0.0	0	18.46	609.57	0	0.0	0
4	20	050	171.27	6010.31	4	0.08	0	654.35	2469.88	47	1.85	0
4	20	100	460.63	10469.44	30	1.4	0	839.8	1708.54	81	7.6	0
4	20	150	642.44	12511.33	56	4.72	1	834.74	1106.36	89	12.74	0
4	20	200	728.34	13852.23	71	8.15	3	886.89	694.04	98	18.04	1
4	30	050	884.18	6185.51	96	12.2	2	890.66	528.35	99	14.3	6
4	30	100	897.95	4639.17	100	21.75	17	897.17	282.47	100	25.12	53
4	30	150	898.06	4624.91	100	27.17	36	896.71	267.88	100	21.41	83
4	30	200	875.92	3464.78	98	30.81	59	897.06	338.29	100	21.36	93
4	40	050	897.84	2700.86	100	19.33	26	897.04	153.67	100	28.86	94
4	40	100	898.21	1251.27	100	26.13	56	NaN	NaN	100	NaN	100
4	40	150	899.1	1118.9	100	27.68	71	NaN	NaN	100	NaN	100
4	40	200	899.84	776.29	100	32.28	86	NaN	NaN	100	NaN	100
4	50	050	898.15	782.25	100	20.06	49	NaN	NaN	100	NaN	100
4	50	100	899.25	732.48	100	23.56	69	NaN	NaN	100	NaN	100
4	50	150	901.19	764.0	100	28.55	87	NaN	NaN	100	NaN	100
4	50	200	902.34	652.0	100	25.88	99	NaN	NaN	100	NaN	100
5	10	050	1.21	442.15	0	0.0	0	6.6	459.17	0	0.0	0
5	10	100	2.78	652.64	0	0.0	0	29.69	617.03	0	0.0	0
5	10	150	4.01	800.64	0	0.0	0	49.33	756.65	0	0.0	0
5	10	200	13.47	3423.48	1	0.01	0	69.81	991.37	1	0.01	0
5	20	050	231.31	5996.59	6	0.23	0	758.11	1649.3	66	4.19	0
5	20	100	338.47	8510.74	12	0.59	0	842.47	1197.18	85	9.02	0
5	20	150	528.01	9153.59	29	2.6	0	892.04	570.78	98	13.32	2
5	20	200	547.4	8691.26	35	3.66	0	890.79	452.34	98	11.84	10
5	30	050	895.2	5592.21	99	16.92	5	897.2	415.29	100	21.09	27
5	30	100	896.48	4145.59	99	22.8	18	896.91	297.61	100	22.23	72
5	30	150	898.38	2789.93	100	26.89	29	897.54	129.0	100	27.31	98
5	30	200	893.37	3561.28	99	26.87	40	897.36	1.0	100	23.39	99
5	40	050	897.94	1515.33	100	26.78	37	NaN	NaN	100	NaN	100
5	40	100	898.73	1024.86	100	28.84	49	NaN	NaN	100	NaN	100
5	40	150	899.88	1086.89	100	31.03	73	NaN	NaN	100	NaN	100
5	40	200	900.94	960.88	100	29.89	75	NaN	NaN	100	NaN	100
5	50	050	898.35	744.19	100	26.74	68	NaN	NaN	100	NaN	100
5	50	100	900.09	886.38	100	30.04	79	NaN	NaN	100	NaN	100
5	50	150	902.37	877.6	100	33.22	90	NaN	NaN	100	NaN	100
5	50	200	902.95	974.0	100	29.44	98	NaN	NaN	100	NaN	100