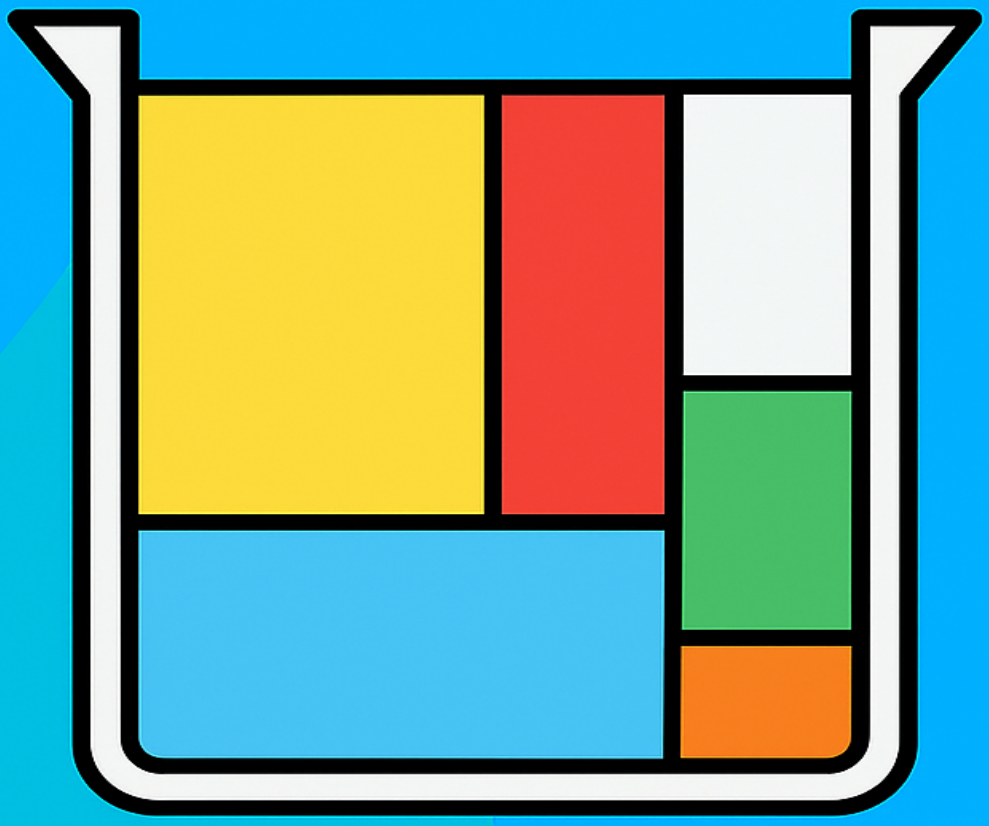
TU Delft



The Bin Packing Problem with Negative Items

Delft University of Technology

Bin Packing Problem with Negative Items

Dieter Bos
Student Number: 5618258

Bachelor Project Thesis
BSc Applied Mathematics

Supervisor: Teun Janssen
Graduation Committee: Wim van Horssen
Defense Date: 23 June 2025

TU Delft

Summary

The standard bin packing problem is about distributing a set of items with positive sizes over bins of certain capacity in such a way that the number of bins used is minimized. In this thesis a relatively undocumented version of bin packing with negative items is discussed. Negative items added to the standard problem can reduce the load in a bin. In context, this means energy is regenerated or recharged. This forces us to take a look at the problem in a different way. The goal of this thesis is to find methods that give solutions with a guaranteed desired quality.

For the standard problem, we already have well-working heuristic algorithms such as the First Fit (FF), Best Fit (BF), and their decreasing-order variants (FFD and BFD). These algorithms have nice bounded approximation ratios. However, introducing negative items sabotages these nice approximation ratios. We know negative items can reduce the total number of bins, so naively applying classical algorithms does not perform well, as these methods do not take the potential of negative items to offset large positives and reduce bin usage into account.

To try and solve these problems, we introduce the adjusted heuristic algorithms Negative First Fit (NFF), Negative Best Fit (NBF), and their decreasing-order variants, which work by placing all negative items into the first bin before applying the standard algorithm. By placing all negative items in the first bin, we can get a packing that uses 1 bin or more. If the packing uses only 1 bin, the problem is trivial. If it uses more than 1 bin, we know that all negative items in the first bin have been canceled out, so we can retain the logic of the standard problem and thus the approximation ratio.

Although the negative adjusted algorithm has a nice approximation ratio, putting all negative items in the first bin does not make sense in a real-life context. That is why we also explore the heuristic method of dividing negative items over large positive ones before applying standard packing algorithms. Here we focus on the notion that fewer big items, where we define big items as items larger than half the capacity, lead to a smaller number of bins for the solution because all big items need to be in a separate bin. First we have Decreasing Division (DD), which pairs the largest big item with the smallest negative items. Unfortunately, this does not guarantees a reduction of big items. To try and improve on this, we also introduce Increasing Division (ID). ID pairs the smallest big item to the smallest negative item. This way, ID can give a bigger guarantee to reduce the number of big items. DD and ID both have the same approximation ratio and both give more logical packing results. However, we proved that for DD there is no better approximation ratio. For ID however, we saw that a better approximation ratio may still be found.

This thesis is mostly based on literature research, mathematical analysis and explanatory examples. We show the potential performance and limitations of different heuristic algorithms for bin packing with negative items. The results achieved give a different perspective on the bin packing problem and leave open future research in optimizing bin packing with negative items. But most importantly, the goal of finding methods that work on the bin packing problem with negative items was achieved.

Layman's summary

In this thesis, the bin packing problem with negative items was discussed. Bin packing consists of distributing a number of items with different sizes over bins with a certain capacity. The problem is to minimize the number of bins used. Possible application for solving this problem are time or cost reduction problems. The goal was to find methods that give solutions with a guaranteed desired quality. Before adding negative items, we first looked at the standard case where all items are positive. Here we saw that putting the items in one by one is an easy and good way to do it. Putting the items in decreasing order and then putting them in one by one worked best. We showed that there was no way to guarantee a better solution. Unfortunately, if negative items are added, these approaches do not work anymore. We saw the negative items should be used to try and decrease the number of bins and that the current methods did not take this into account. The first thing we adjusted was putting all negative items into the first bin and then applying some of the positive bin packing methods. This again gave a guarantee in performance. However, resulting packings did not make sense in a real-life context. Another idea was to divide the negative items over big positive items so that we get the positive bin packing problem again. After this, we could then apply an earlier method. This also worked nicely and resulted in more logical packings. Unfortunately, there was uncertainty in how to divide the negative items optimally, so there is still improvement that can be done here. In conclusion, our goal was to find methods that guaranteed good solutions, which succeeded.

Contents

List of variables

n:	Number of bins
w_i:	The size of item i, which is positive
v_i:	The size of item i, which is negative
L:	A set containing a number of positive or negative items
Big item:	A positive item whose size is more than half the capacity
L^*:	A set L on which a certain transformation is applied
W:	$W = \sum_{i=1}^{n} w_i$, i.e. the total sum of all the sizes of all the items
Load:	The sum of the sizes of the items in a bin
b:	The number of bins a certain bin packing algorithm gives for a set L
b^*:	The optimal number of bins needed for a certain set L
α:	The approximation ratio
NF:	Next Fit Algorithm
FF:	First Fit Algorithm
BF:	Best Fit Algorithm
FFD:	First Fit Decreasing Algorithm
BFD:	Best Fit Decreasing Algorithm
NFF:	Negative First Fit Algorithm
NBF:	Negative Best Fit Algorithm
NFFD:	Negative First Fit Decreasing Algorithm
NBFD:	Negative Best Fit Decreasing Algorithm
DD:	Decreasing Division
ID:	Increasing Division

Chapter 1

Introduction

Mathematical optimization problems exist everywhere in the world. Problems like the famous traveling salesman problem or minimizing the number of people getting infected by the Covid-19 virus are important problems that spark the interest of many mathematicians. In general, optimization problems are about a certain function or value that needs to be minimized or maximized. When talking about optimization problems in the context of capacity, the most important one is the bin packing problem.

The bin packing problem is a really basic optimization problem. Suppose we have a number of elements with a certain size and we have bins with a certain capacity that cannot be exceeded. The question is, how can we distribute the elements over the bins such that the number of bins used is minimized?

The nice thing about the bin packing problem is that it is applicable for all sorts of capacity optimization problems. Take for example the packaging of products over freight trucks or the storage of online files over servers to minimize the number of trucks or servers to reduce costs. In these situations, the trucks or servers are the bins and the products and files are the elements. There are also a lot of variations of the bin packing problem. We can look at it from two dimensions or three dimensions, change the sizes of the capacities of the bins or add conflicts so that certain items cannot go in the same bin. In this thesis we are going to discuss a much lesser researched variation of the bin packing problem. What if we add negative items to the problem? Does this have such a big impact on the known research of the standard problem? Does it become easier or harder?

Let us first discuss what kind of meaning negative items could have in a real-life scenario. Suppose we have a machine in a factory with a certain energy capacity that needs to execute a number of tasks. Every time the machine performs a task, it uses an amount of energy, but the machine can also recover energy by charging up. The factory wants to be as profitable as possible and thus wants to reduce costs by using as few machines as possible. In this situation, the bins are the machines, the positive elements are the tasks and the negative items are the charge-up moments. This example shows that there are real-life applications for bin packing with negative items, so it is important to find out more about it.

The negative items seem like a small addition, but they have great consequences for the standard bin packing problem. It seems that the minimum number of bins possible should be lower, but is that always the case? And if so, how would we determine a way to distribute the negative items such that it is optimal? That brings me to the research question of this research thesis. Can we find methods that give solutions with a guaranteed desired quality?

To answer this research question, we will be looking at some examples of bin packing problems with negative items. Then the problem will be mathematical analyzed and proofs will be provided for characteristics of this problems.
Firstly, in Chapter 2 the standard bin packing problem will be analyzed and some heuristic algorithms will be discussed to get familiar with the fundamentals of the problem. Then in Chapter 3 we will extend the problem by adding the negative items. We will try and link the observations of Chapter 2 and the bin packing problem with negative items. In this section bin packing with negative items will be analyzed on why certain theorems or conclusions of standard bin packing do or do not apply to negative bin packing. If certain statements do not work, they will be analyzed to see if they can be adjusted in such a way that we can still make some conclusions about them. In Chapter 4 the conclusions of the results of this thesis will be drawn. Also some shortcomings of this thesis will be discussed and some possible future research in on this subject will be mentioned.

Chapter 2

Bin packing with positive items

To get a good idea of the bin packing problem with negative items, we will analyze the classical problem of bin packing. First the problem will be mathematically defined and then some heuristic algorithms will be discussed and analyzed.

2.1 Problem formulation

The bin packing problem consists of the number of items that need to be distributed, the size of these items and the capacity of the bins. It will be mathematically stated as the following. Consider a list of $n \in \mathbb{N}$ numbers and define this as the set $L = (w_1, w_2, \ldots, w_n)$, where w_i is the size of the item i. The problem is to assign each item to a bin such that the sum of the item sizes in a bin does not exceed the capacity while minimizing the number of bins used. Therefore, a certain capacity and a range of sizes for the items need to be decided. If the capacity is way higher than the range of sizes, the problem gets more trivial as it becomes easier to put items in a bin. If the capacity is way lower than the range of sizes, the problem becomes impossible as some items may become larger than the capacity and thus cannot fit into any bin. So the capacity and the range of the sizes need to be in proportion. For this thesis, we consider a capacity of 1 and a range of items between 0 and 1. So $w_i \in (0, 1]$. These sizes are useful, as all bin packing problems can be transformed into this problem by scaling.

Now that we have defined the problem, let us take a look at an example and try to solve it manually. Consider the set $L = (0.4, 0.8, 0.3, 0.1, 0.7, 0.6, 0.2)$. The most trivial solution is to put every single item into a unique bin, which can be seen in Figure 2.1.

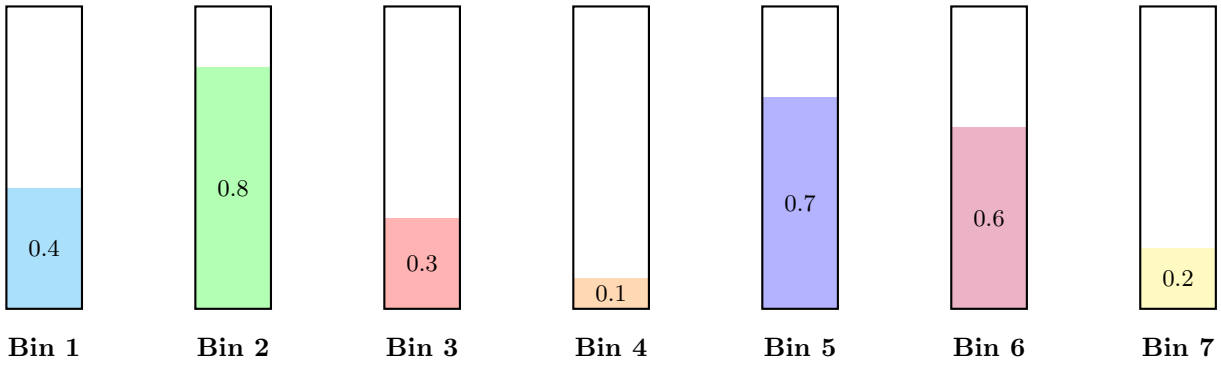


Figure 2.1: Seven bins each with a single item.

This solution gives 7 bins. Note that this is possible for every situation, so the optimal number of bins is always less than or equal to the number of items in the set. We can also conclude something about the lower bound for the optimum number of bins. Define a big item as an item whose size is larger than half the capacity. When looking at the number of big items in a set, we can observe that all these items all need to be in a different bin. So for each big item we need a unique bin, which means that a lower bound for the optimum is the total number of big items.
Nevertheless, Figure 2.1 is clearly not optimal. Some bins need to be combined. Observe that bin 1 and 6, bin 2 and 7 and bin 3 and 5 sum perfectly to the capacity, so combining them gives Figure 2.2.

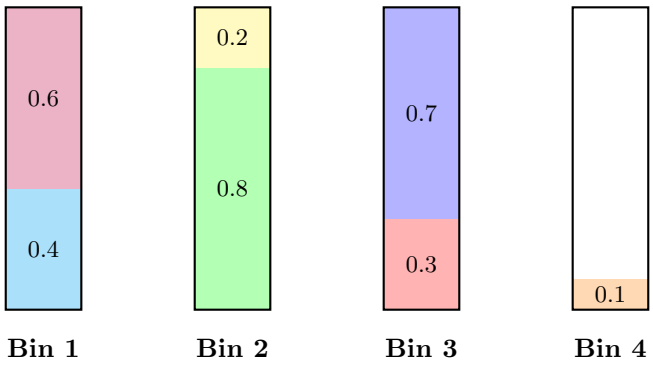


Figure 2.2: Four bins with items.

This gives 4 bins as a solution, which is optimal as no bins can be removed anymore. This example shows that the bin packing problem is not too hard for small sets, but observe that it becomes much more complicated when the number of items in the set gets bigger. For big sets, algorithms are needed which will be discussed in the next section.

2.2 Heuristic algorithms and their bounds

The first and most basic algorithm is the Next Fit (NF) algorithm. For NF, take an item and check if it fits. If it does, put it in; else close the bin and put it in a new one. Applying NF to the set L from the previous section, we get the solution shown in Figure 2.3.

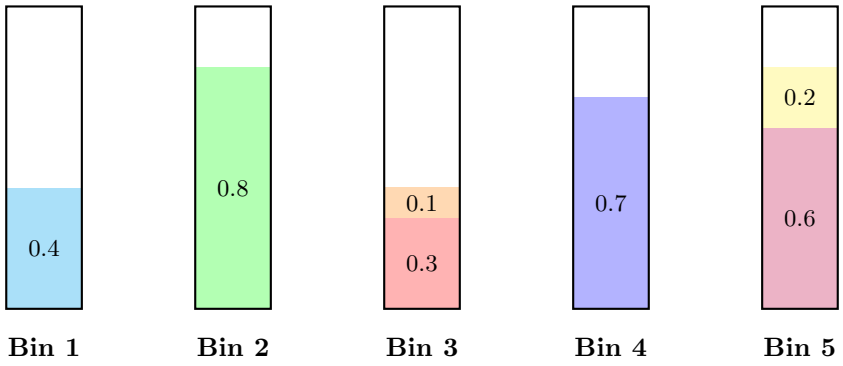


Figure 2.3: Next Fit packing.

This results in a packing using 5 bins. Although it is better than 7 bins, it is still not the optimal solution, because NF blindly closes bins even if there is still space left in them. In the example it is clear that the items in bin 3 easily fit in bin 1. But Next Fit can be improved by checking if items fit in previous bins.

This gives us the First Fit (FF) algorithm and the Best Fit (BF) algorithm. These work in a similar way with a small difference. Start by taking the first item of the set and putting it in the first bin. For FF, put each next item in the first bin it fits in. For BF, put each next item in the first bin it fits with the biggest load, where "load" means the sum of the sizes of all items of a certain bin. Continue this process until it is finished. These algorithms both improve on NF and they work in a similar fashion. If we apply FF or BF to the set $L = (0.5, 0.7, 0.5, 0.2, 0.4, 0.2, 0.5, 0.1, 0.6)$, we get the packing shown in Figure 2.4.

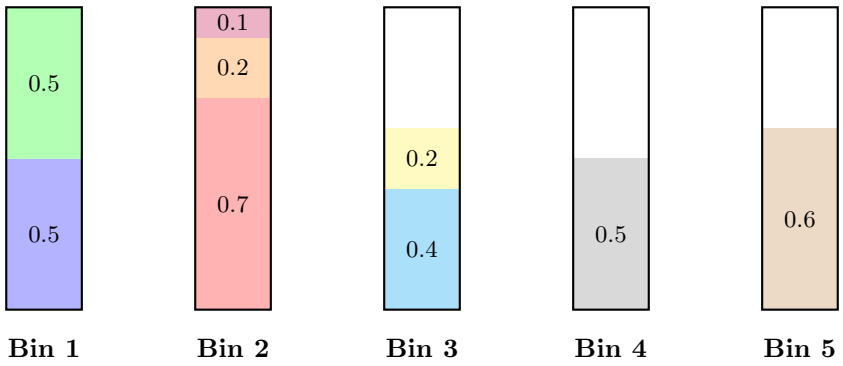


Figure 2.4: First Fit and Best Fit packing.

Applying FF or BF on L gives a solution that uses 5 bins. For larger sets, FF and BF give more different packing results. Note that this packing is not optimal. Still, FF and BF perform quite well. To get an idea of how good the performance is of these algorithms, we define b as the number of bins a certain algorithm gives for a set L and b^* as the optimal number of bins for that same set L. Then we can define the following bound: $b \leq \alpha b^*$. This means that a certain algorithm gives a number of bins that is at most α times the optimal number of bins. The factor α in this inequality is called the approximation ratio. This gives us the following Theorem.

Theorem 2.2.1. *The FF and BF algorithms both have an approximation ratio of 2, so for both algorithms it holds that* $b \leq 2b^*$.

Proof. We will prove the statement for FF. Note that the case for BF can be proven in a similar way.

Let $W = \sum_{i=1}^{n} w_i$ be the total sum of all item sizes. We aim to show that the following two properties hold:

1. $W \leq b^*$
2. $b < 2W + 1$

Then it follows from these properties that $b \leq 2W \leq 2b^*$.

Property 3.1: $W \leq b^*$

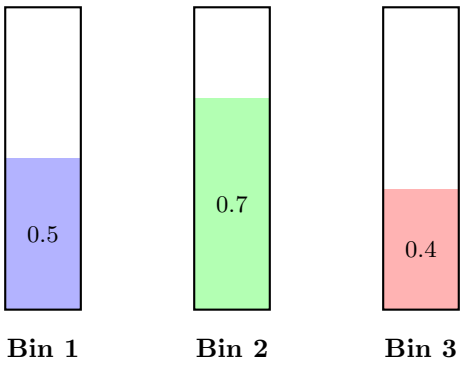
Assume for contradiction that $W > b^*$. This means that the total sum of sizes of the items is bigger than the minimum number of bins required to pack the items. So if we were to fill up b^* amount of bins, there would still be some size left for which an extra bin is needed. But then b^* is not an optimal number of bins because it requires one bin more, so we get a contradiction. Therefore, $W \leq b^*$ must hold.

Property 3.2: $b < 2W + 1$

To show this, we prove the following claim:

Claim: At most one bin is at most half full in the result of the First Fit (FF) algorithm.

Suppose that the FF algorithm gives a packing with 2 or more bins that are at most half full. Take a look at the bin that is at most half full and that is the farthest away. But because the total size of the items in this bin is at most half the capacity, the items in that bin should also have fit in the previous bin, which was at most half full. So this is not a packing that the FF algorithm could have given. Thus we get a contradiction, so the claim must be true. For clarity, take a look at the following example.



In this situation, there are 2 bins that are at most half full. But if we follow the FF algorithm, it puts an item in a bin if it fits. The item of size 0.4 fits in bin 1 and therefore should be put into bin 1, so this packing result is not possible.

From the claim, we conclude that $b-1$ bins are more than half full. Thus,

$$\frac{1}{2}(b-1) < W \quad \Rightarrow \quad b-1 < 2W \quad \Rightarrow \quad b < 2W+1$$

□

So the First Fit and Best Fit algorithms give at most 2 times the optimal number of bins. David Simchi-Levi even proved that the worst-case bound can be lowered to 1.75 in his paper [1]. However, these algorithms still do not give the optimal solution. But they can be improved by ordering the list of numbers of the set in order of decreasing sizes. These are the First Fit Decreasing (FFD) and the Best Fit Decreasing (BFD) algorithms. Take again the set $L = (0.5, 0.7, 0.5, 0.2, 0.4, 0.2, 0.5, 0.1, 0.6)$. When we order it, we get a new set $L^* = (0.7, 0.6, 0.5, 0.5, 0.5, 0.4, 0.2, 0.2, 0.1)$. If we then apply FF or BF to the new set L^*, we get the packing of Figure 2.5.

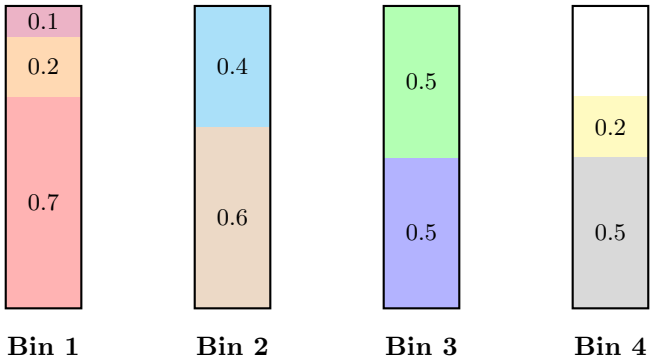


Figure 2.5: First Fit Decreasing and Best Fit Decreasing packing.

Applying FFD or BFD on L gives a solution that uses 4 bins. This is one bin less than the FF and BF algorithms gave and so works better on this set. Also observe that this is an optimal solution. That is because the first 3 bins are full and the fourth bin has only 0.3 capacity left. If we want to delete one bin, there would need to be a capacity of at least one left over which is not the case.

Because the FFD and BFD algorithms are just a way of using the FF and BF algorithms on a different set, we know that FFD and BFD must give results that are at least equal to or better than the FF and BF algorithms. This also means that both FFD and BFD must have an approximation ratio of at most 2. David Simchi-Levi also proved that FFD and BFD both have an approximation ratio of 1.5. This still means that it does not give the optimal solutions in every situation. But is it possible to find an algorithm that gives the optimal solution every time? Is it even possible to get a better approximation ratio than 1.5 with an algorithm with a polynomial running time? The answer unfortunately is no.

Theorem 2.2.2. *There is no $\alpha < 1.5$ such that $b \leq \alpha b^*$ assuming $P \neq NP$.*

Proof. We prove this by looking at the NP-hard partition problem. Suppose we have a set of n non-negative integers $a_1, a_2, ..., a_n$ where $S = \sum_0^n a_i$. The problem is to partition the set into 2 sets such that each set is equal to $S/2$. Now suppose there exists an algorithm $\mathcal{A}$ for bin packing with approximation ratio $\alpha < 1.5$. Now take an instance of the bin packing problem with set $L = (a_1, a_2, ...a_n)$, where the total sum of items is S, the bin capacity is equal to $S/2$ and where the optimum number of bins is 2. Then if we apply algorithm $\mathcal{A}$, it must find a packing with 2 filled bins as a solution. But this problem is the same as partitioning the items in set L over 2 sets that sum to $S/2$. So $\mathcal{A}$ should be a working polynomial-time algorithm that can also be used for the partition problem. But it is proven that there is no such algorithm to solve the partition problem unless $P \neq NP$, thus there cannot be an algorithm with approximation ratio $\alpha < 1.5$.

□

For more information about the theory behind the P vs NP problem, take a look at the book Computers and Intractability: A Guide to the Theory of NP-Completeness by Garey and Johnson [2]. This book offers more detailed explanations and classic results about the field of computational complexity.
Now that we have a taken a look at the standard bin packing problem, we will continue in the next section by adding negative items.

Chapter 3

Bin packing with negative items

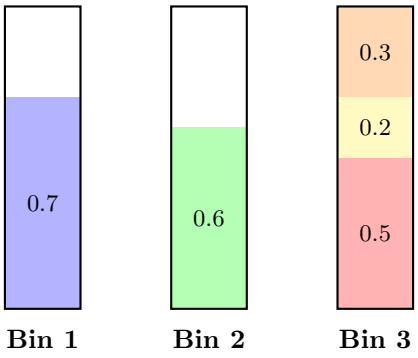
In the previous chapter, the bin packing problem with positive items was analyzed. In this chapter, the problem will be extended by adding negative items. We will take a look at how this affects the problem. Then some heuristic algorithms from the previous chapter will be analyzed on the extended problem.

3.1 Problem formulation

As seen in the previous chapter, we saw that the bin packing problem with positive items consists of a list of $n \in \mathbb{N}$ numbers and a set $L = (w_1, w_2, \ldots, w_n)$, where w_i is the size of the item i. For convenience, a bin capacity of 1 was chosen. Also, a range of sizes for the items was chosen to be $(0, 1]$. But now we add the negative items, so a set could look like $L = (w_1, v_1, \ldots, v_n, w_n)$, where w_i is the size of the positive item i and v_i is the size of the negative item i. The range $(0, 1]$ does not suffice for v_i anymore and needs to be extended with negative items. But this cannot be done blindly, as it raises some questions.

What if, by putting negative items in a bin, the capacity becomes negative? Should this be allowed? In the context of machines with a certain energy capacity, it initially does not make sense. At 0 the energy capacity is full, so how can it be negative? But it can work if it is seen like an extension of the capacity of the batteries. So there are possible situations where the capacity could be negative.
These negative items also need to be proportional to the bin capacity. If the item is really small, it does not have an effect on the problem. If the item is really big (thus a large negative value) it makes the problem more trivial as more positive items can fit with that item. So therefore we will assume the item sizes are in the range of $[-1, 1]$.
Lastly, the total sum of the sizes of the negative items needs to be considered. If this total is really big, the problem becomes trivial again because it impacts the total sum of the items. The negative items could just cancel out the positive items and they can then all be put in one bin. So the sum of the sizes of the negative items cannot be too big as well.

Now let us take a look at an example. Take the set $L = (0.7, -0.2, 0.6, 0.5, 0.2, 0.3, -0.4)$. A nice way to tackle this problem is first to put in the positive items. Then we could get the following packing.



Now we still have the -0.2 and the -0.4, but these cancel out the 0.6 perfectly. So then we can merge that bin with the 0.7 to get Figure 3.1.

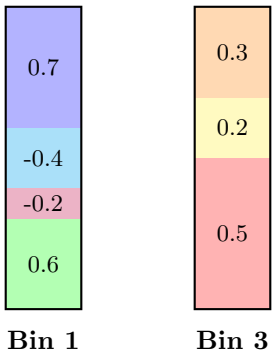


Figure 3.1: Example bin packing with negative items.

This packing is obviously optimal and it was not hard to find. But making a heuristic algorithm that repeats this progress is a lot harder. This algorithm needs to make use of the negative items in such a way that the packing needs fewer bins, but there are a lot of ways to divide the negative items over the positive items. To find the most optimal way to do this is hard. Before we try and find such an algorithm, let us first try to get an idea of how hard the bin packing problem with negative items is.
In the previous chapter, we saw that for bin packing there are no approximation algorithms with an approximation ratio smaller than 1.5. Does this change when negative items are added to the problem? Intuitively we might think that the approximation ratio can become lower as negative items can only lower the number of bins, but this is not the case. The set of problems for negative packing still contains the set of problems for positive packing and thus is at least as hard. So because finding an algorithm with an approximation ratio smaller than 1.5 for positive bin packing is not possible, it is also not possible for negative bin packing. The question now is, what is the best approximation ratio we can find? In the next section, we look at how the negative items affect the previously discussed heuristic algorithms.

3.2 Heuristic algorithms with negative items

The first heuristic algorithm we review is again the Next Fit algorithm. For standard bin packing, we saw that this was not the best algorithm, as it closes bins even if they can still be used. For bin packing with negative items, this does not change. We take at look a the following example where we apply NF on the set $L = (0.5, 0.6, -0.4, -0.1, 0.5, 0.6, 0.5, -0.2, 0.3, 0.5, 0.3, -0.2)$. This gives us the packing of Figure 3.2.

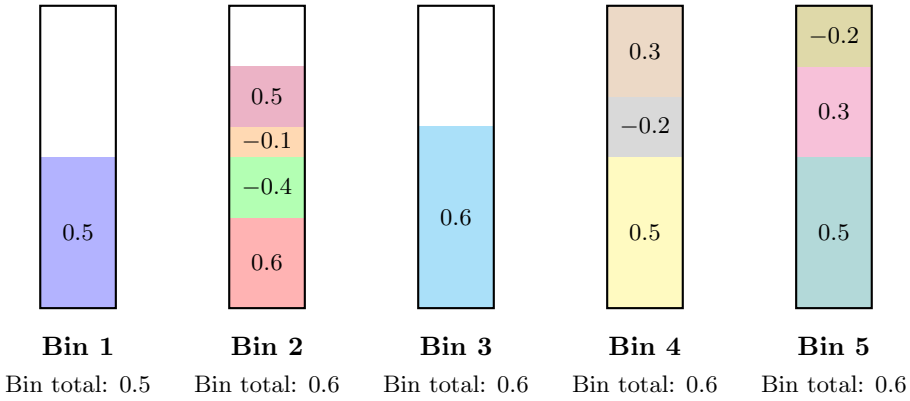


Figure 3.2: Next fit with negative items.

NF gives us a solution with 5 bins, but this is clearly not optimal. Bin 1 and bin 3 are closed too early, which leaves a lot of capacity. That capacity could be used for the items of bin 4 and bin 5. So the addition of negative items does not prevent the flaw of NF.
To fix this problem of the NF algorithm, we looked at the BF and FF algorithms. These algorithms did take the previous bins into account. We also saw that they both worked in very similar ways and that they had an approximation ratio of 1.75, which is a really nice bound. Let us now take a look at what happens with these algorithms when negative items are added to the problem. We will first take a look at the FF algorithm. Applying FF to the previous set L, we get Figure 3.3.

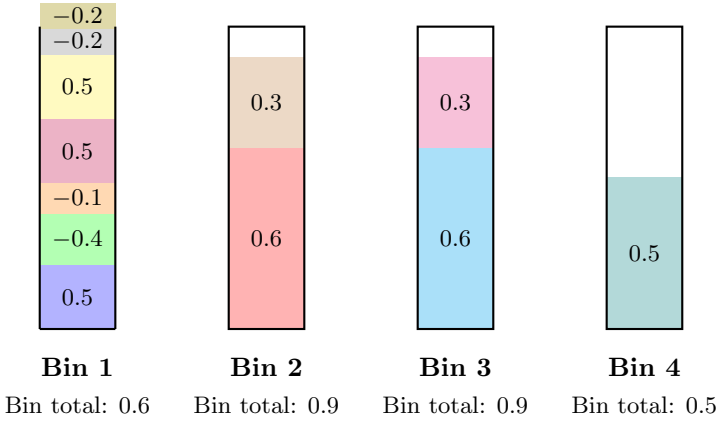


Figure 3.3: FF with negative items.

Again, this does improve the number of bins from 5 to 4, as some items of the previous bin 4 and bin 5 are divided over bins in the front. But in this example something stands out. All negative items are put in the first bin. Looking at how the FF algorithm works it becomes clear why. The FF algorithm checks from the first bin if an item fits and then puts it in if it does. But a negative item will always fit in a bin thus all the negative items will be put where it starts to check, which is the first bin.

Now let us take a look at the BF algorithm. Where we saw previously that BF and FF are very similar when considering only positive items, the negative items now cause a big change between them. The BF algorithm starts to check if an item fits in the bin with the biggest load. This means that the negative items are not all just put in the first bin but can end up in other bins. If BF is applied on the set $L = (0.5, 0.6, -0.4, -0.1, 0.5, 0.6, 0.5, -0.2, 0.3, 0.5, 0.3, -0.2)$ again, we get the packing of Figure 3.4.

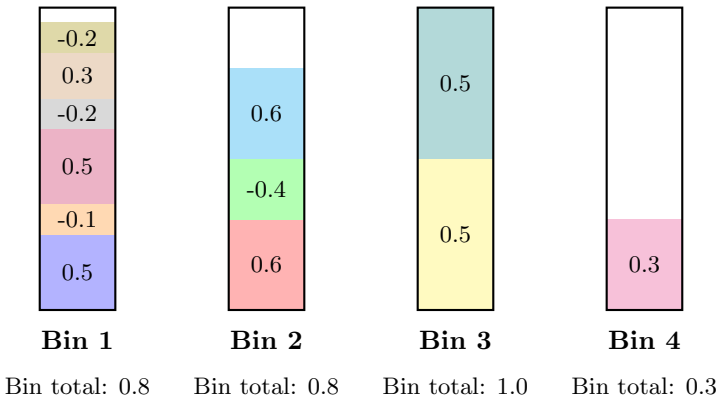


Figure 3.4: BF with negative items.

.

Indeed, now not all the negative items are put in the first bin. Although BF does not give a better number of bins for the set L, we could argue that it gives a better packing. This is because the -0.4 is more useful in the second bin than in the first bin. So this example hints that the BF algorithm could give some better results than the FF algorithm.
We saw in Chapter 3.2 that the order of the items in set L is very important. Applying the FF or BF algorithm to the same set of items but with a different order can give some better or worse results. We also saw that putting the items in decreasing order before applying FF or BF, it generally gave a better packing. The application of FFD and BFD on the set L can be seen in Figure 3.5 and Figure 3.6.

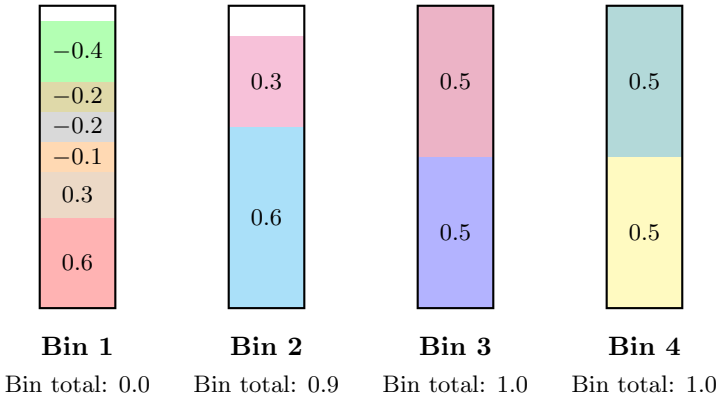


Figure 3.5: FFD with negative items.

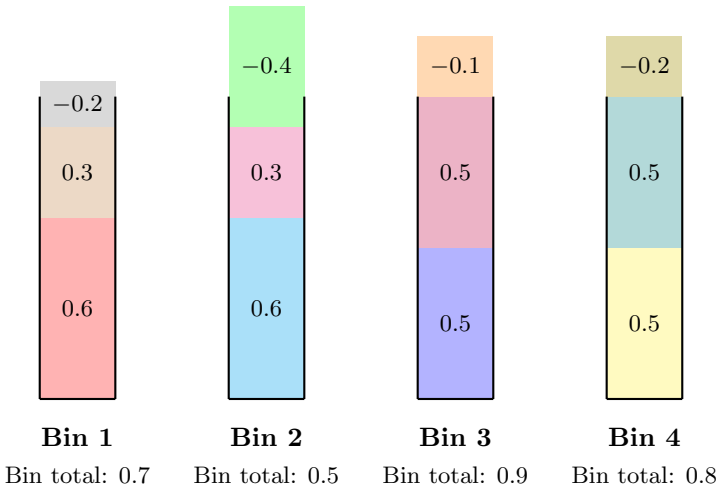


Figure 3.6: BFD with negative items.

From these examples, it becomes clear that the FFD and BFD algorithms do not perform better than the FF and BF algorithms. Because of the decreasing order, the negative items are all put in the back of the set L^*. This means that the negative items are the last items to put in, which is why the negative items are always on top of the positive items in each bin. Again we see the same difference as with FF and BF, where with FF all the negative items are put in the first bin and with BF the items are more divided over all bins.
But the negative items are now useless. By putting in the positive items first, we use a certain number of bins. Adding the negative now does not reduce this number of bins. But we can see clearly from Figure 3.5 that there is a way to use the negative items to reduce the number of bins as bin 1 and bin 2 can be combined. So even if we do not know for certain that there is a better number of bins, we know for sure that the FFD and BFD cannot find this improvement.

In the standard bin packing problem, we saw that FFD and BFD were the same as applying FF and BF on the best possible set order. But for bin packing with negative items, we see that FFD and BFD are the worst-case versions of FF and BF. This not only means that FFD and BFD are not good heuristic algorithms, but also that FF and BF are not ideal as well. In the next section, we try to find a way to make these algorithms work for bin packing with negative items as well.

3.3 Adapted heuristic algorithms for negative items

In the previous section, we looked at how negative items affected the existing heuristic algorithms for the bin packing problem. We concluded that just copying these algorithms on negative bin packing does not guarantee success because they do not necessarily make use of the negative items. So we need to find a way to make sure the negative items are incorporated into the algorithm.
To find such an algorithm, we start by analyzing the performance of the FF on negative bin packing previously discussed. Again note that analyzing the performance of BF is similar. We would like to get a certain approximation ratio because then at least we have a bound for the problem. We aim to get an approximation ratio of 2 by comparing the logic of proof theorem 3.2. For this proof we need that Propositions 3.1 and 3.2 hold.

For Proposition 3.1, which was $W \leq b^*$, negative items do not have an effect on the logic. Suppose again the total size of the items was bigger than the optimal number of bins. Then trying to fit all items in b^* number of bins will still result in some leftover items that do not go in any bin. This contradicts again the fact that it is an optimal solution. Thus, the addition of negative items does not change that and Proposition 3.1 still holds.
However, for Proposition 3.2, which was $b < 2W + 1$, the negative items do make a difference. For proving Proposition 3.2, we previously looked at the following claim: At most one bin is at most half full in the result of the First Fit (FF) algorithm. For bin packing with negative items this is not the case anymore. As we saw previously, all the negative items are put in the first bin. This can result in not only the last bin is at most half full but also the first one. As example look at Figure 3.3 again. If the set L were to contain one more extra negative item at the end, both bin 1 and bin 4 would be half the capacity, which contradicts the claim.

But suppose we adjust the algorithm so that after packing the items, it fuses the bin which load is at most half the capacity in such a way that the claim does hold. Then there would still be a problem. Using the claim we conclude that $\frac{1}{2}(b-1) < W$, but negative items break this piece of logic. This is because if there is indeed a bin that is filled to at most half its capacity, this bin could be filled with negative items. So it could be that the total for that bin is negative and this breaks the $\frac{1}{2}(b-1) < W$ inequality.
So for bin packing with negative items, Proposition 3.2 has the problem that the claim and the logic after that break. But there is a way to adjust the FF algorithm in such a way that is solves these problems and gives us the desired approximation ratio of 2. By putting all the negative items in the first bin and then applying the FF algorithm, we get the negative first fit (NFF) algorithm.

Theorem 3.3.1. *For the NFF algorithm it holds that $b \leq 2b^*$ and they thus have an approximation ratio of 2.*

Proof. We have already shown that Proposition 3.1 still holds. Thus, we only need to prove Proposition 3.2 holds for NFF. We will do this by proving our claim again and making sure that the logic after still holds.

If we put all the negative items in the first bin and then apply the FF algorithm, we get 2 situations. Either all items fit in the first bin, which means the packing is trivial, or there is more than 1 bin needed. In the second case, the claim holds again because we do not have any negative items anymore, so the algorithm works like the positive bin packing problem again. This means that is it impossible the FF results in more than 1 bin that is at most half full, so the claim holds.
Now the second piece of logic also still holds. Because more than 1 bin is needed, we know that the first bin, with all the negative items, must have a positive load. Furthermore all the negative items are used, which means that all other bins in the packing must have a positive load as well. Thus we can use the claim to conclude that $\frac{1}{2}(b-1) < W$ holds again, thus proving proposition 3.2.
□

For BF, FFD and BFD we can do something similar. If we apply BF, FFD or BFD after putting all negative items in the first bin we get the Negative Best Fit algorithm (NBF), Negative First Fit Decreasing (NFFD) and Negative Best Fit Decreasing algorithms (NBFD). These algorithms all have an approximation ratio of 2 as well which can be proven in a similar way. Applying NFF or NBF on the set $L = (0.5, 0.6, -0.4, -0.1, 0.5, 0.6, 0.5, -0.2, 0.3, 0.5, 0.3, -0.2)$ we get Figure 3.7 and applying NFFD or NBFD on set L we get Figure 3.8.

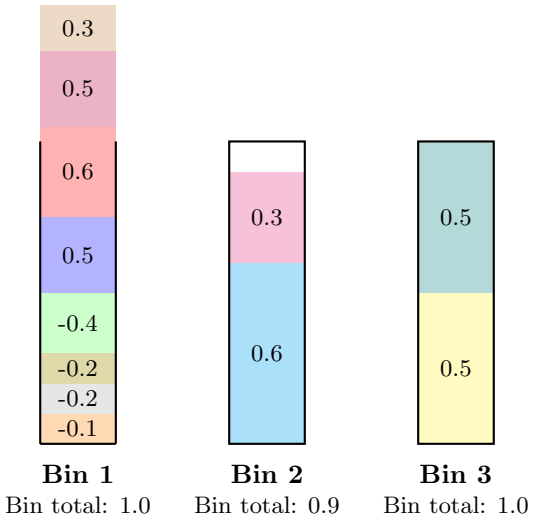


Figure 3.7: NFF and NBF.

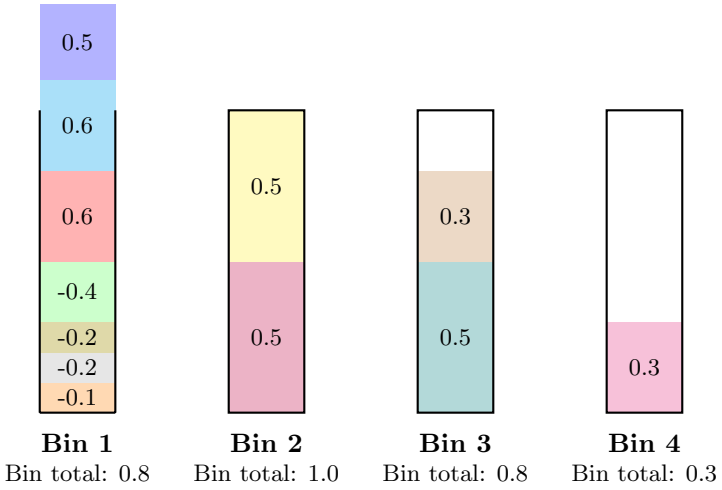


Figure 3.8: NFFD and NBFD.

It would make sense that after putting all negative items in, we apply the best algorithm for positive bin packing. However, we see here that applying FF or BF here gives better results. This is different compared to positive bin packing. We concluded that FFD and BFD must give a result that is at least equal to or better than the standard bin packing problem, but this example shows that this is not the case for NFFD and NBFD when applied to the problem with negative items. We can also not improve the approximation ratio of 2.

Theorem 3.3.2. *For the NFF, NBF, NFFD and NBFD there is no smaller approximation ratio then 2.*

Proof. To prove this, we need to construct a special set. For $n = 2k$, $k \in \mathbb{N}$ and $d \in (0, 0.5)$ define the set Z_n^d as a set that contains n positive items with size $\frac{1}{2} + d$ and n negative items with size $-d$. As an example, we take a look at the set $Z_{n=4}^{0.1} = (0.6, 0.6, 0.6, 0.6, -0.1, -0.1, -0.1, -0.1)$. From this example it is clear that the optimal solution pairs the 0.6 and the -0.1 together to get 4 items of size 0.5 which can be put into 2 bins perfectly. Applying NFF on this set gives us Figure 3.9.

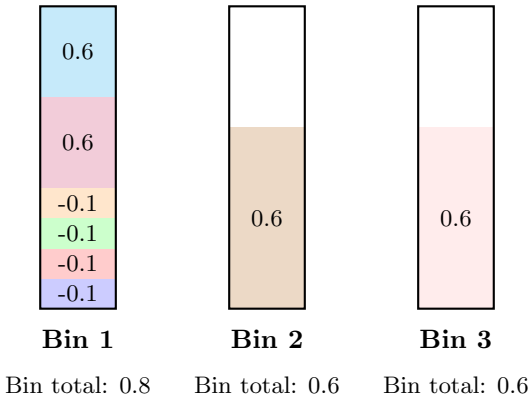


Figure 3.9: NFF on Z_4^1

The algorithm gives a number of bins of 3. So from this set we conclude that the NFF algorithm has an approximation ratio of at least $\frac{4}{3}$. To obtain a general approximation bound for the NFF algorithm, we need a general formula for the optimum number of bins for the set Z_n^d and a general formula for the number of bins that NFF gives for this set.
The formula for the optimal number of bins is clear from the example. For each set Z_n^d, adding the $\frac{1}{2} + d$ size items with the $-d$ size items results in n items of size $\frac{1}{2}$. Then combine this to get $\frac{1}{2}n$ bins.
The second formula is a bit harder but still possible. First, take a look at the first bin. This bin is first filled with a total of $-nd$. Then a j number of $\frac{1}{2} + d$ size items will be added to that bin. The remaining $n - j$ items of size $\frac{1}{2} + d$ will each be put in their own bin resulting in a packing consisting of $1 + n - j$ bins. Now we need to figure out how many $\frac{1}{2} + d$ size items fit into the first bin with the n $-d$ size items (j). An item fits in a bin if the load of the bin is less than the capacity minus the size of the item. The load first starts at $-nd$. If we add the maximum of j items to the bin, the following inequality should hold.

$$-nd + j(\frac{1}{2} + d) \leq 1 - \frac{1}{2} + d \quad \Rightarrow \quad -nd + \frac{1}{2}j + jd - \frac{1}{2} - d \leq 0.$$

If we now take the limit of d to 0 we end up with

$$\frac{1}{2}j - \frac{1}{2} \leq 0$$

The biggest j for which this inequality holds is 2. We have proven here that no matter how big our n is, if the d is small enough, we can fit at most 2 items of size $\frac{1}{2} + d$ into the first bin with all the negative items. Define $\epsilon > 0$, then we conclude that the NNF algorithm applied on the set Z_n^ϵ always gives $n - 1$ bins. We can now calculate a least bound for the approximation ratio by the following formula.

$$\frac{n-1}{\frac{1}{2}n} \quad = \quad 2 - \frac{2}{n}$$

Now take the limit from n to ∞ and we get the following.

$$\lim_{n\to\infty} 2 - \frac{2}{n} \quad = \quad 2$$

□

Although we have found a nice approximation ratio with these negative adjusted algorithms, we still have a problem when we look at the resulting packings these algorithms give. Putting all negative items in the first bin does not make sense in a real-life scenario. There might be improvement possible in getting more logical packing results. To try and find this improvement, we look at the problem from a different angle. Let us take a look at the big items again, which we defined as the items whose sizes are larger than half the capacity. In Chapter 3.1, we concluded that the total amount of big items must be a lower bound for the optimal number of bins because all these big items need to be in separate bins. So, if we want to use the negative items to try and reduce the number of bins used for a set L, we can try this by reducing the number of big items in that set L. The question now becomes how to divide the negative items over the positive items to achieve this. Successfully dividing the negative items allows us to eliminate the big items. Essentially, we are transforming the packing problem with negative items into a standard packing problem. So after the division, we should just apply a good algorithm we know for the bin packing problem without negative items and should get a good result.
So how can we divide the negative items heuristically? First off all, we introduce the Decreasing Division (DD). This means that we take the biggest positive item and the smallest negative item and pair them. Secondly, we take a look at the second-biggest items and second-smallest negative items and pair them and continue this until all negative items are paired. The paired items can now be seen as a single item, with their combined size. So we end up with a new set. If the sizes of some of these pairs are negative apply DD on this new set again. If applying DD over and over again results in a set only containing negative items, the problem is trivial. Otherwise it will result in a new set with only positive items. On this set we apply our best-known algorithm, FFD.

Theorem 3.3.3 (DD)**.** *Using DD until all negative items are gone and then applying FFD has an approximation ratio of 2 for bin packing with negative items.*

Proof. Because of DD, we transform the negative packing problem into a positive one. Because we apply FFD to this positive problem, we can again follow theorem 3.1, which results in an approximation ratio of 2. □

If we apply DD on the set $L = (0.5, 0.6, -0.4, -0.1, 0.5, 0.6, 0.5, -0.2, 0.3, 0.5, 0.3, -0.2)$, we get the set $L^* = ((0.5, -0.2), (0.6, -0.4), (0.5, -0.1), (0.6, -0.2), 0.5, 0.3, 0.5, 0.3)$, which is the same set as $L = (0.3, 0.2, 0.4, 0.4, 0.5, 0.3, 0.5, 0.3)$. Applying FFD on the set L after DD we get Figure 3.10 which in our original problem translates to Figure 3.11.

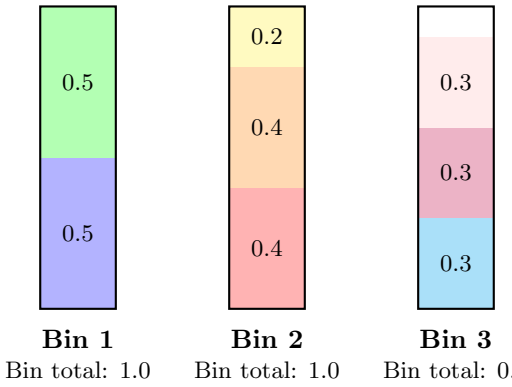


Figure 3.10: FFD applied on a set transformed by using DD

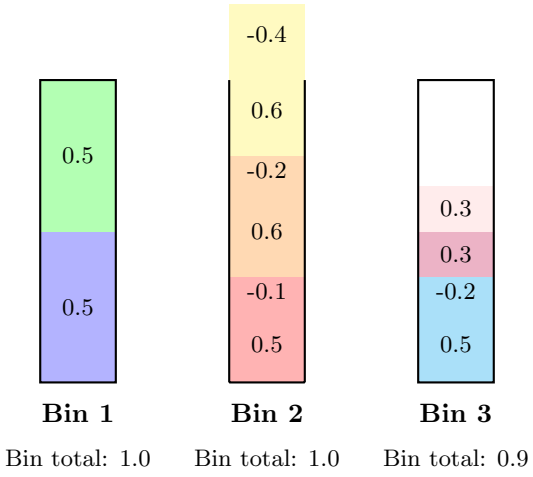


Figure 3.11: The real result of Figure 3.10

This gives the optimal number of bins of 3 again, like we have seen before, so for this example it works really well again. But unfortunately there are cases where it does not.

Theorem 3.3.4. *For applying FFD after DD, there is no smaller approximation ratio than 2.*

Proof. For this proof, we need to construct a special set again. For $n = 2k$, $k \in \mathbb{N}$ and $\epsilon > 0$, define the set X_n as a set $(\frac{1}{2}+(n+1)\epsilon, \frac{1}{2}+n\epsilon, \ldots, \frac{1}{2}+\epsilon, -n\epsilon, -(n-1)\epsilon, \ldots, -\epsilon)$. Again, we need a general formula for the optimum number of bins for the set X_n and a general formula for the number of bins that DD combined with FFD gives for this set.
For the general optimum number of bins, it is clear that we need to pair the items of size $\frac{1}{2}+n\epsilon$ with the items of size $-n\epsilon$. This way we end up with n items of size $\frac{1}{2}$ and 1 item of size $\frac{1}{2}+(n+1)\epsilon$, which can fit in less bins than $\frac{1}{2}n+1$. Unfortunately, DD performs poorly on this set, resulting in n+1 items that are bigger than $\frac{1}{2}$, thus needing n+1 bins.

We can now calculate a least bound for the approximation ratio by the following formula.

$$\frac{n+1}{\frac{1}{2}n+1}$$

Now take the limit from n to ∞ and we get the following

$$\lim_{n\to\infty} \frac{n+1}{\frac{1}{2}n+1} \quad = \quad 2$$

□

Take for example the set $L = (0.9, -0.3, 0.8, -0.2, 0.7, -0.1, 0.6)$. When we apply DD on this set, we end up with $L^* = (0.6, 0.6, 0.6, 0.6)$. The items in this set are all big items and thus need to go in separate bins. If the 0.8 and 0.7 are paired with -0.3 and -0.2, we can get a solution that uses 3 bins. This example shows that DD does not reduce the number of big items, which was our goal. To achieve this more effectively, we introduce Increasing Division (ID). When applying ID, pair the smallest negative items with the smallest big item, the second-smallest negative items with the second-smallest big item and so on. If there are no more big items, we pair the next negative items to the biggest items again. This way we get a higher guarantee that, if possible, a big item is removed. If this does not happen, then there are not enough negative items to achieve this. Applying ID on $L = (0.9, -0.3, 0.8, -0.2, 0.7, -0.1, 0.6)$ we get the set $L^* = ((0.6, -0.3), (0.7, -0.2), (0.8, -0.1), 0.9)$, which is the same set as $L = (0.3, 0.5, 0.7, 0.9)$. Applying FFD on the set L after DD we get Figure 3.12 which in our original problem translates to Figure 3.13.

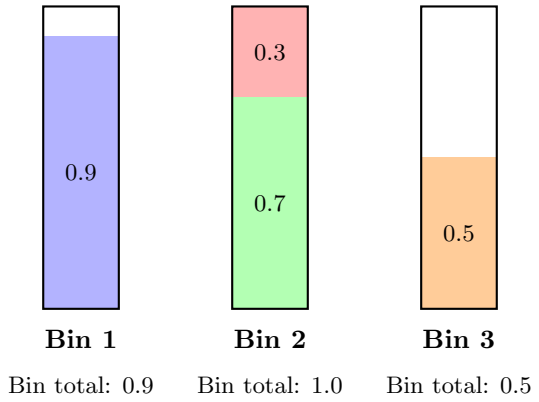


Figure 3.12: FFD applied on a set transformed by using ID

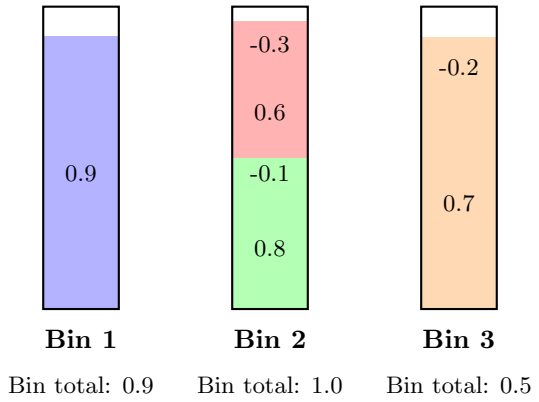


Figure 3.13: The real result of Figure 3.10

Where DD gave a solution of 4 bins, ID now gives us the optimal solution with 3 bins. We can clearly see from this example that, although ID does not always divide the negative digits as optimally as DD, it does reduce the number of big items more consistently, thus ID is better sort of general form of division. In a similar way as in Theorem 3.3.3 it holds that applying ID until there are no negative items again and then applying FFD has an approximation ratio of 2. But the proof of Theorem 3.3.4 breaks now, so we may find a better approximation ratio.

Chapter 4

Conclusion & Discussion

4.1 Conclusion

In this thesis we analyzed the bin packing problem with negative items. Although there was already much known about bin packing with positive items only, the problem including negative items was relatively unresearched. That led us to the research question: *Can we find methods that provide solutions with a guaranteed desired quality?* This question was answered by looking at the already known and analyzed standard bin packing problem. After that, we tried to utilize this research on the problem with negative items added.
In Section 2.1 we set a foundation for the research and defined the bin packing problem mathematically. Then in Section 2.2 we looked at a multitude of possible heuristic algorithms. We concluded that the FF and BF algorithms had an approximation ratio of 1.75, which means they already performed well. The order of the items in the set was also important, as the FFD and BFD algorithms even had a better approximation ratio of 1.5. We ended this section with the conclusion that 1.5 was impossible to improve, assuming $P \neq NP$.
In Section 3.1 we defined the addition of the negative items. We concluded that this problem is at least as hard as the standard bin packing problem, so we cannot find a lower approximation ratio than 1.5 again. We continued in Section 3.2 with applying the heuristic algorithms from Section 2.2 on negative bin packing. We observed that the FFD and BFD algorithms became worse than the FF and BF algorithms with negative items because these algorithms did not utilized their potential. But also FF and BF did not have an approximation ratio of 2 anymore. We needed a way to ensure that the algorithm made more use of the negative items. We saw in Section 3.3 that by putting all negative items in the first bin and then applying a method seen before, we solved this problem, restoring the 2 approximation ratio again. We also introduced the Decreasing Division and the Increasing Division transformations. These methods transformed packing problems with negative items into packing problems with only positive items. Then FFD could be applied again an this combination has an approximation ratio of 2 again.

4.2 Discussion

The results obtained during this research project must be viewed within its limitations. First off all, we concluded that putting all negative items in the first bin and then applying an algorithm seen before was an efficient way to get a good packing result and it even got an approximation ratio of 2. However, we did not review the resulting packings of this method and what they mean in practice. The context for bin packing with negative items we discussed in the introduction was a machine with a certain energy capacity. In this case the bins were the energy capacity of a single machine, positive items were tasks that drained energy and negative items were charging moments where energy was gained. In this context, it would make more sense to divide these charging moments over the machines instead of putting them all on one machine. So even though we have an approximation ratio of 2, it would make more sense to apply the DD or ID transformations to make sure the negative items are more spread out, resulting in more logical packing results.
After obtaining the results of this research, it is time to explore if more research can be done about the bin packing problem with negative items. Although we have proven that the approximation ratio of 2 for putting all negative items in the first bin or the DD transformation cannot be lowered, it may still be possible to lower it for the ID transform. But besides proving that this method may have a better performance, there is also more research to be done in the division techniques. To get the most use of the negative items, we wanted to reduce the number of items whose sizes were more than half the capacity by dividing these negative items using DD or ID. There could be a better way to do this by using integer linear programming and trying to minimize the number of such items with a size bigger than half the capacity.
Lastly we saw that there are a lot of variations of the standard bin packing problem. Take, for example, the bin packing problem with conflicts, talked about in more detail by Klaus Jansen in An Approximation Scheme for Bin Packing with Conflicts [3]. Another variation considers a fixed number of bins, rather than minimizing the number of bins used, as discussed by Jansen, Kratsch, Marx, and Schlotter in Bin Packing with Fixed Number of Bins Revisited [4]. Combining these variations with negative items leads the more questions and problems that can be analyzed.

Bibliography

[1] David Simchi-Levi. New worst-case results for the bin packing problem. *Naval Research Logistics*, 1994.

[2] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, 1979.

[3] Klaus Jansen. An approximation scheme for bin packing with conflicts. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1999.

[4] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[5] Bin packing problem. `https://en.wikipedia.org/wiki/Bin_packing_problem`. Accessed: 2025-06-16.