Bin Packing Problems with Item Fragmentation

Nir Menakerman and Raphael Rom

November 7, 2001

Abstract

We investigate several variants of bin packing problems in which items may be fragmented into smaller size pieces called fragments. While there are a few applications to bin packing with item fragmentation, our model of the problem is derived from a scheduling problem present in data over CATV networks. Fragmenting an item is associated with a cost which renders the problem NP-hard. The first part of this paper is concerned with worst case analysis; We analyze the following problems: Bin packing with size increasing fragmentation, packing variable size bins with item fragmentation and the problem of size-preserving fragmentation, i.e., when fragmentation does not change the size of items. For each problem we investigate the performance of some well known algorithms such as Next-Fit and First Fit Decreasing, as well as of other algorithms. The second part of the paper is devoted to average case analysis. We present a new technique of average case analysis and use it to analyze the performance of the Next-Fit algorithm for the problem of bin packing problem with item fragmentation.

Contents

1 Introduction

Because of its applicability to a large number of applications and because of its theoretical interest bin packing has been widely researched and investigated. A comprehensive survey of bin packing problem can be found in [CGJ96]. In the classical one-dimensional bin packing problem, we are given a list of items $L = (a_1, a_2, ..., a_n)$, each with a size $s(a_i) \in (0, 1]$ and are asked to pack them into a minimum number of unit capacity bins. Since the problem, as many of its derivatives, is NP-hard many approximation algorithms have been developed for it (see e.g., [Ull76],[CGJ84]).

The bin packing problem presented in this paper is derived from a scheduling problem present in data over CATV networks. In particular we refer to Data-Over-Cable Service Interface Specification (DOCSIS), standard of the Multimedia Cable Network System (MCNS) standard committee. For a detailed description see [MCNS]. When using CATV networks for data communication the data subscribers are connected via a cable modem to the headend. The headend is responsible for the scheduling of all transmissions in the upstream direction (from the cable modem to the headend). Scheduling is done by dividing the upstream, in time, into a stream of numbered mini-slots. The headend receives requests from the modems for allocation of datagram transmission. The length of each datagram can vary and may require a different number of minislots. From time to time, the headend publishes a *MAP* in which it allocates mini-slots to one modem or a group of modems. The scheduling problem is that of allocating the mini slots to be published in the MAP, or in other words, how to order the datagrams transmission in the best possible way.

The headend must consider two kinds of allocations:

1. Allocations for connections with timing demands, such as a CBR connection (referred to as fixed allocations). These connections must be scheduled so as to ensure delivering the guaranteed service. For a CBR connection, these would typically be scheduled in fixed, periodically located minislots.

2. Allocations for connections without timing demands, such as a best effort connection. These connections can use any of the minislots.

The headend therefore performs the allocation in two stages: in the first stage it schedules, or allocates, minislots for connections with timing requirements. In the second stage all other allocations are performed. Because different CBR connections have different requirements (i.e., different periods) and because the size of the MAP is not necessarily related to the connections, the fixed allocations may appear in different locations in the different MAPs, which means that every MAP might have a completely different structure.

The relation to the bin packing problem should now be clear. The items are the datagrams that should be scheduled, each of which may require a different number of minislots. The bins are defined by the gaps between every two successive fixed allocations in the MAP. The goal is to use the available minislots in the MAP, in the best way.

One of the capabilities of the system is the ability to break a datagram into smaller pieces called *fragments*. When a datagram is fragmented, extra bits are added to the original datagram to enable the reassembly of all the fragments at the headend.

The subject of item fragmentation in bin packing problems received almost no attention so far. This paper concentrates on aspects that were heretofore never researched, such as developing algorithms for the problem and investigating their performance. We restrict our attention to practical bin packing algorithms, i.e., of low order polynomial running time, and

examine both *online* and *offline* algorithms. Online algorithms are applicable to cases where the items arrive in some order and must be assigned to a bin as soon as they arrive. Next-Fit and First-Fit are example of online algorithms. Offline algorithms assume the entire list of items is known before the packing begins, this knowledge may be used for sorting the items for example. First-Fit Decreasing and Best-Fit Decreasing are examples of offline algorithms. In addition to the classical definitions of bin packing, we also look into other environments such as bins of variable sizes and finite number of bins.

As indicated earlier, to make the problem nontrivial a cost must be associated with fragmentation. In the scheduling problem, where items correspond to datagrams, the cost may be due to the extra overhead bits that are added to each fragment for reassembly purposes. Other fragmentation costs can be those resulting from processing time or reassembly delay. It is interesting to note that when the cost associated with fragmentation is ignored the packing problem becomes trivial, and when the cost is very high, it does not pay to fragment items and we face the classical problem. Hence, the problem is interesting with the middle-range costs. It has been shown [MCG98] that for non zero cost the ability to fragment items does not reduce the complexity of the problem, that is, the problem of bin packing with item fragmentation is NP-hard.

The scheduling problem we described can also be modeled as a *Multiple Knapsack Problem* (MKP) [MT81]. In MKP we are given a pair (B, L) where B is a set of m bins (knapsacks) and L is a set of n items. Each bin $B_j \in B$ has a capacity $c(B_j)$, and each item $a_i \in L$ has a size $s(a_i)$ and a profit $p(a_i)$. The objective is to find a subset $S \subseteq L$ of maximum profit such that S has a feasible packing in B.

We are going to treat the Multiple Knapsack Problem in a future paper, where we deal with some aspects in which the scheduling problem differs from the classical bin packing problem:

1. **Variable size bins** - The bins may be of different sizes.
2. **A finite number of bins** - The number of bins a MAP defines is finite.
3. **Small bins** - Not all items are necessarily smaller than all bins (as assumed in the classical problem).

Another characteristic of the scheduling problem which we intend to explore in the future is the ability to modify bin sizes. Since a bin's size is a consequence of timing requirement it may be possible to slightly alter some of the fixed location. This results in a variant of bin packing where the bin sizes can be altered in a constrained manner.

The remainder of the paper is organized as follows. In Section 2 we address the problem of bin packing with size-increasing fragmentation. Section 3 explores the case of packing variable size bins. Section 4 discusses the problem of size-preserving fragmentation, i.e., when fragmentation does not change the size of items. In Section 5 we present an average case analysis of the NF algorithm for the problem of bin packing problem with item fragmentation.

2 Bin Packing with Size-Increasing Fragmentation

In this section we study the variant of bin packing with item fragmentation where fragmentation increases the size of an item. We define the problem similar to the classical bin packing problem. The classical bin packing problem deals with equal-sized (unit capacity) bins and a list of items each of which can fit in every bin. To handle fragmentation, we use a discrete version of the problem and add a fragmentation cost function that adds overhead units to each fragment. We proceed to formally define the problem.

Bin Packing with Size-Increasing Fragmentation (BP-SIF): We are given a list of items $L = (a_1, a_2, ..., a_n)$, each with a size $s(a_i) \in \{1, 2, ..., U\}$. The items must be packed into a minimum number of bins, which are all the size of U units. When packing a fragment of an item, one unit of overhead is added to the size of *every* fragment.

Example: Assume the bin size is U=10 and we are given three items of sizes 5, 6 and 7. We can pack the items in only two bins in the following way: We pack the items of sizes 5 and 7 one in a bin and fragment the item of size 6 into two fragments of sizes 2 and 4. When we pack the fragments we add one overhead unit to each, so their sizes become 3 and 5, respectively. We pack the fragment of size 5 with the item of size 5 and the fragment of size 3 with the item of size 7.

Performance Ratio: We use the same definition as is typically used in analyzing the classical problem. For a given list L and algorithm A, let $A(L)$ be the number of bins used when algorithm A is applied to list L, let $OPT(L)$ denote the optimum number of bins for a packing of L, and let $R_A(L) \equiv A(L)/OPT(L)$. The ***asymptotic worst case performance ratio*** R_A^∞ is defined to be:

$$R_A^\infty \equiv \inf\{r \geq 1 : \text{ for some } N > 0,\ R_A(L) \leq r \text{ for all } L \text{ with } OPT(L) \geq N\} \qquad (1)$$

The bin packing problem is known to be NP-hard in the strong sense [GJ79]. We show that the complexity of BP-SIF is the same.

Claim 2.1 *BP-SIF is NP-hard in the strong sense.*

Proof: We denote by D(BP-SIF) the decision version of BP-SIF and show that it is NP-Complete in the strong sense. We do so by reducing the 3-PARTITION problem to a restricted instance of D(BP-SIF). The 3-PARTITION problem (defined formally below) is known to be NP-Complete in the strong sense [GJ79].

3-PARTITION: given a list L of $n = 3m$ integers: $w_1, w_2, ..., w_n$ and a bound $B \in Z^+$ such that $B/4 < w_j < B/2$ for $j = 1, ..., n$ and $\sum_{j=1}^{n} w_j = m\,B$, can L be partitioned into m disjoint subsets $S_1, ..., S_m$ such that $\sum_{j \in S_i} w_j = B$ for $i = 1, ..., m$?

We define D(BP-SIF) as follows: given a list of items L, a size $s(a) \in Z^+$ for each item $a \in L$, a positive integer bin capacity U and a positive integer K, is there a feasible packing of L in K bins of size U? Any instance I of 3-PARTITION can be polynomially transformed into an equivalent instance I' of D(BP-SIF) by setting $U = B$ and $K = m$. To realize the two decision problems are equivalent note that, since the total size of all items is equivalent to the total capacity of all bins, in any "yes" instance of D(BP-SIF) all n items are packed without fragmentation and the packing is therefore also valid for 3-PARTITION. Clearly the packing of any "yes" instance of 3-PARTITION is also valid for D(BP-SIF). It follows that the "yes" and "no" instances of the two problems are equivalent. $\triangle$

When packing a list of n items into m bins, the maximum number of fragmentations possible is $n \cdot m$ (each item is fragmented over all bins). From the definition of the problem it is obvious that a good algorithm should try to perform the minimum number of fragmentations. Therefore we would only like to consider algorithms that do not fragment items unnecessarily.
Definition 1: An algorithm A is said to *prevent unnecessary fragmentation* if it follows the following two rules:

1. No unnecessary fragmentation: An item (or fragment of an item) is fragmented only if it is to be packed into a bin that cannot contain it. In case of fragmentation, the item (or fragment) is divided into two fragments. The first fragment must fill one of the bins. The second fragment is packed according to the packing rules of the algorithm.

2. No unnecessary bins: An item is packed in a new bin only if it cannot fit in any of the open bins used by A.

Algorithms that prevent unnecessary fragmentation have the following property.

Lemma 1 *For any algorithm A that prevents unnecessary fragmentations - $R_A^\infty \leq \frac{U}{U-2}$, for every $U > 2$.*

Proof: Assume that the number of bins used by the algorithm is $A(L) = m$. Since A prevents unnecessary fragmentation it can perform at most $m-1$ fragmentations while packing m bins (no fragmentation in the last bin), regardless of the size of the list. Each fragmentation adds 2 units of overhead at the most. Therefore, in the worst case, $2(m-1)$ units of overhead are added to the total size of all items. Note that in this case only the last bin may be left unfilled. Assuming the optimal packing does not fragment any item, the number of bins used by it satisfies:

$$OPT(L) \geq \frac{(m-1)U - 2(m-1) + 1}{U} = (m-1)\frac{U-2}{U} + \frac{1}{U} \tag{2}$$

The asymptotic performance ratio follows:

$$R_A(L) = \frac{A(L)}{OPT(L)} \leq \frac{mU}{(m-1)(U-2)+1} \quad \stackrel{m\to\infty}{\longrightarrow} \quad R_A^\infty \leq \frac{U-2}{U}$$

$\triangle$

Remark: For the more general case, where r units of overhead (instead of one) are added to the size of every fragment, it can be shown by similar arguments, that: $R_A^\infty \leq \frac{U}{U-2r}, \quad U > 2r$.

We now have an upper bound on the performance ratio of any algorithm. In the remainder of this section we investigate specific algorithms to find their actual performance ratio. For a given algorithm A we define a version of the algorithm that allows item fragmentation and denote it by A_f. We investigate the worst case performance ratio of the following algorithms: NF_f, NFD_f (NFI_f) and FFD_f (BFD_f).

2.1 Next-Fit with item fragmentation - NF_f

The NF_f algorithm is defined similar to the NF algorithm.

Algorithm NF_f - In each stage there is only one open bin. The items are packed, according to their order in the list L, into the open bin. When an item does not fit in the open bin, it is fragmented into two parts. The first part fills the open bin and the bin is closed. The second part is packed into a new bin which becomes the open bin. Offline version of the algorithm sorts the items in decreasing (NFD_f) or increasing (NFI_f) order before packing the list.

The NF_f algorithm is very simple, can be implemented to run in linear time and requires only one open bin (bounded space). However, as we show next, similar to the classical problem, the performance ratio the algorithm achieves is the worst possible.

Theorem 1 *For algorithm* NF_f - $R^{\infty}_{NF_f} = \frac{U}{U-2}, \quad \forall\, U \geq 6.$

Proof: Lemma 1 provides an upper bound on the performance ratio of the algorithm. We present an example that proves the lower bound. Let us first consider the case where the bin size U is an even number. As a worst case example we choose the following list L: The first item is of size $U/2$, the next $\left(\frac{U}{2}-2\right)$ items are of size 1. The rest of the list repeats this pattern kU times. An example of such a list, for U=6, is given in Figure 1. The optimal packing avoids fragmentations by packing bins with two items of size $U/2$ or U items of size 1. The total number of bins used is: $OPT(L) = \frac{U}{2}k + \left(\frac{U}{2}-2\right)k = (U-2)\,k$. On the other hand, algorithm NF_f fragments each item of size $U/2$ (except for the first item). Overall $2(kU-1)$ units of overhead are added to the packing and therefore the number of bins used by NF_f is:

$$NF_f(L) = \left\lceil \frac{U}{2}k + 2\frac{kU-1}{U} + \left(\frac{U}{2}-2\right)k \right\rceil = \left\lceil Uk - \frac{2}{U} \right\rceil = Uk. \tag{3}$$

A worst case example for the case where U is an odd number is similar. The first item in L is of size $(U-1)/2$, the next $\left(\frac{U-1}{2}-1\right)$ items are of size 1. The rest of the list repeats this pattern kU times. It is easy to verify that, as in the previous example, $OPT(L) = (U-2)k$, while $NF_f(L) = kU$.

We conclude that for every $U \geq 6$ the asymptotic performance ratio is: $R^{\infty}_{NF_f} = \frac{NF_f(L)}{OPT(L)} = \frac{U}{U-2}$. △

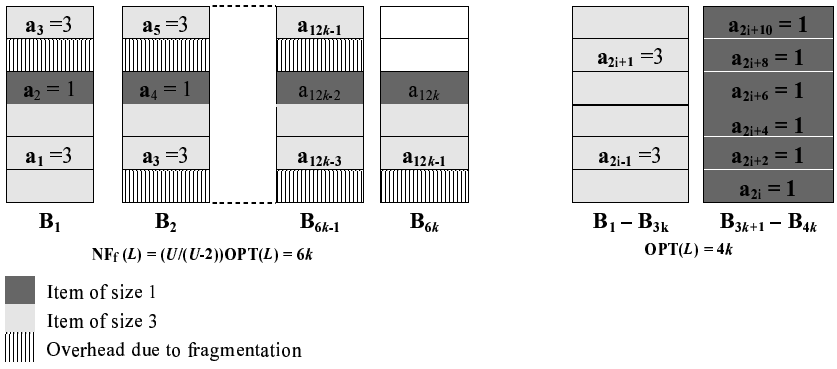

Figure 1: An example for the packing of NF_f for $U = 6$, the list is $L = \{3, 1, 3, 1....3, 1\}$.

When the bin size is very small the above proof does not hold. We now show that for the values $3 \leq U \leq 5$ the worst case asymptotic performance ratio is: $R^{\infty}_{NF_f} = \frac{3}{2}$.

- $U = 2$: $\overline{R}^{\infty}_{NF_f} = \frac{4}{3}$, for example $L = \{1, 2, 1, 2, ..., 1, 2\}$.
- $U = 3$: $\overline{R}^{\infty}_{NF_f} = \frac{3}{2}$, for example $L = \{1, 3, 1, 3, ..., 1, 3\}$.
- $U = 4$: $\overline{R}^{\infty}_{NF_f} = \frac{3}{2}$, for example $L = \{2, 3, 2, 1, ..., 2, 3, 2, 1\}$.
- $U = 5$: $\overline{R}^{\infty}_{NF_f} = \frac{3}{2}$, for example $L = \{3, 3, ..., 3, 2, 2, ..., 2\}$.

It is interesting to compare the NF_f algorithm to the classic NF algorithm for which the asymptotic worst case performance ratio is: $R^{\infty}_{NF} = \frac{2U}{U+1}, \quad \forall\, U \geq 1$. While the performance ratio of NF is increasing with U the performance ratio of NF_f is decreasing with U. This is intuitive since as the bin size gets larger the effective cost of fragmentation gets smaller.

2.1.1 NFD_f and NFI_f Algorithms

Given the poor performance of the NF_f algorithm, one may ask whether sorting the items before applying the NF_f packing, would yield better results. It turns out that algorithms NFD_f and NFI_f are very similar to the NF_f algorithm in their worst case performance. The actual performance ratio of these algorithms depends on the bin size U, but in all cases it is not far from the ratio of NF_f. To avoid dealing with each value of U separately, we present an example for a general value. The example serves as a lower bound on the performance ratio for any value of U, but as we shell see in some cases the ratio may be even worse. Our list of items is made of k items of size $U-2$ and k items of size 2. An example for such a list, for $U = 7$, is given in Figure 2. The optimal packing uses k bins where each bin contains two items, one of each kind. The NFD_f algorithm first packs k items, of size U-2, into k bins. Then k-1 items of size 2 are packed into $\lceil 2\,(k-1)\,/U \rceil$ bins if U is even, or $\lceil 2\,(k-1)\,/\,(U-1) \rceil$ bins if it is odd. The NFI_f algorithm will use the same number of bins, since the only difference is that the items are packed in reverse order. This simple example gives us the following lower bounds:

$$R^{\infty}_{NFD_f} = R^{\infty}_{NFI_f} \geq \frac{U+2}{U}, \quad \text{for any even } U \geq 6 \tag{4}$$

$$R^{\infty}_{NFD_f} = R^{\infty}_{NFI_f} \geq \frac{U+1}{U-1}, \quad \text{for any odd } U \geq 5 \tag{5}$$

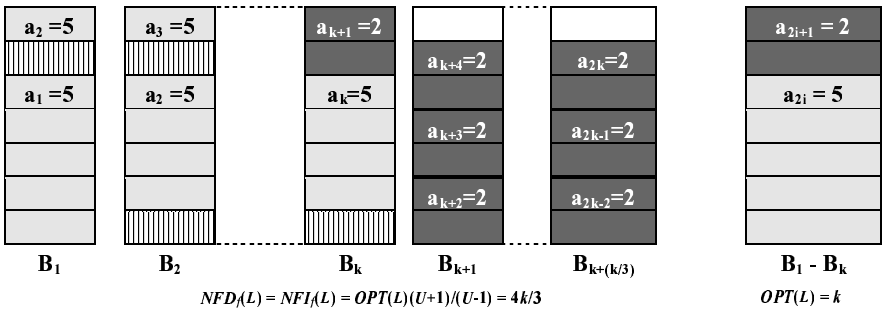

Figure 2: An example for the packing of NFD_f for $U = 7$, the list is $L = \{5, 5, ..., 5, 2, 2, ...2\}$.

The above example provides a lower bound. We now demonstrate that in some cases the NFD_f and NFI_f algorithms can perform just as bad as NF_f. The first example is for U=5

and a list $L = \{3, ..., 3, 2, ..., 2\}$, for which $R^{\infty}_{NFD_f} = R^{\infty}_{NFI_f} = R^{\infty}_{NF_f} = \frac{3}{2}$. To see that such examples are not restricted to low values of bin size, consider $U = 32$ and choose a list of $15k$ items of size 10 and $15k$ items of size 6. In the optimal packing the content of each bin is $\{10, 10, 6, 6\}$, therefore $OPT(L) = 7.5k$. The total number of bins used by the algorithms is: $NFD_f(L) = NFI_f(L) = 8k$, since all bins (save two) contain two units of overhead. The performance ratio in this case is the worst possible:

$$R^{\infty}_{NFI_f} = R^{\infty}_{NFD_f} \Big|_{U=32} = \frac{8k}{7.5k} = \frac{32}{30} = \frac{U}{U-2} \tag{6}$$

On the other hand for some values of bin size, NFD_f and NFI_f have a better performance ratio than NF_f. For example when $U = 6$, $R^{\infty}_{NFD_f} = R^{\infty}_{NFI_f} = \frac{4}{3}$, while $R^{\infty}_{NF_f} = \frac{3}{2}$.

2.2 First-Fit Decreasing with item fragmentation - FFD_f

We now develop an algorithm based on the FFD heuristic. Let us first describe algorithm FFD_f which packs items from a list L, into a *fixed* number of m bins.

Algorithm FFD_f - First-Fit Decreasing with item fragmentation: The algorithm packs the items in decreasing order. An item is packed into the lowest indexed bin into which it fits. If an item does not fit into any bin it is fragmented. When fragmenting an item the first fragment fills the lowest indexed bin that is as yet not full. If the second fragment can be packed without fragmentation, it is packed into the lowest indexed bin into which it fits, otherwise another fragmentation is performed according to the above rule.

Remark: Other definitions are possible. For example, upon fragmentation we may choose to insert the second fragment back to the list. Another possibility is to first go over the whole list and pack items without fragmentation and only then pack the remaining items into the available free space.

Note that FFD_f may not be able to pack all the items in L. To ensure all items are packed, we use the following iterative algorithm:

Algorithm FFD_f **Iterative** ($FFD_f - I$) - The $FFD_f - I$ algorithm tries to pack the list L into a fixed number of m bins. If it fails it increases m by one and tries again. Let $s(L)$ be the sum of all items in L, the first value of m is: $m_1 = \lceil s(L)/U \rceil$, which is the minimum number of bins possible.

The algorithm performs the following steps:

1. Set $m = m_1 = \lceil s\,(L)\,/U \rceil$.
2. Try to pack the list L into m bins using the FFD_f algorithm.
3. If all items were packed stop.
4. Otherwise set $m = m + 1$ and go to step 2.

It is interesting to see if $FFD_f - I$ improves the performance ratio of NF_f. As we shell see the improvement is significant for small values of bin size.

Theorem 2 *The asymptotic worst case performance ratio of the $FFD_f - I$ algorithm satisfies:*

(i) $R^{\infty}_{FFD_f - I} \leq \frac{U}{U-1}$ *when* $U \leq 15$

(ii) $\frac{U}{U-1} < R^{\infty}_{FFD_f - I} < \frac{U}{U-2}$ *when* $U \geq 16$

Proof: To prove the theorem we use a property which we call the *border bin property*. We assume $FFD_f - I$ has m bins to pack and number the bins $B_1, ..., B_m$ according to the order they are opened by the algorithm. Looking at the level (used space) of the bins at a certain time during algorithm execution, we say that bin B_j, $j < m$, is a *border bin* at that time if its level satisfies: $L(B_j) \neq L(B_{j+1})$.

Claim 2.2 *At any time before the first item is fragmented, the packing of $FFD_f - I$ contains at most $2U$ border bins.*

Proof: The $FFD_f - I$ algorithm packs the items in decreasing size order. We consider the number of border bins before the first item is fragmented. Denote by $BR(k)$ the number of border bins, after k different sizes of items have been packed by $FFD_f - I$. Clearly $BR(1) \leq 2$ since items of the same size are packed in a similar way. Whenever each additional size is packed the number of border bins may increase by at most two, therefore $BR(k) \leq BR(k-1) + 2$. Since there are at most U different sizes, $BR(U) \leq 2U$, and the claim follows. $\triangle$

We now proceed to prove the theorem for each range separately.

(i) $U \leq 15$: We establish $\frac{U}{U-1}$ as an upper bound on the asymptotic performance ratio. It is clear that as long as the number of bins with two units of overhead is small, i.e., O(1), the asymptotic performance ratio cannot exceed $\frac{U}{U-1}$ (the proof is similar to that of Lemma 1). We make the following observation:

Claim 2.3 *The final packing of the $FFD_f - I$ algorithm **cannot** contain more than O(1) bins with 2 units of overhead, if one of the following conditions is met:*

1. *Before the first fragmentation occurs, the free space in the bins is 2 or less.*
2. *The fragmented items are of size 4 or less.*

Proof: Consider the moment before the first fragmentation is performed. The first condition is trivial since at most 1 unit of overhead can be packed when the free space in the bins is 2. To prove the second condition, knowing that the free space in the bins is at least 3, we need only test the case of items of size 4 (and bins with free space 3). When an item of size 4 is fragmented over two bins with free space of size 3, only one unit of overhead is packed in each bin, together with a fragment of size 2. The only way to get a bin with 2 units of overhead, is to fragment an item of size 4 over three bins which must have free space of sizes 2,3,3 or 3,2,3; in this case the last bin of each triplet contains 2 units of overhead. However, the number of such triplets is O(1), since each of them contains at least one border bin and according to the *border bins property*, there are at most $2U$ border bins. It follows that the number of bins containing 2 overhead units is less than $2U$, which is O(1) for $m \gg U$. $\triangle$

It is easy to verify that the conditions set by the above claim cannot be extended, since fragmenting items of size 5 over bins of size 3 results in one third of the bins containing 2 units of overhead. Therefore, in order to get a significant number of bins with 2 units of overhead, items of size 5 or more must be fragmented. This means that the list should contain items of size 6 or more. Note that if $FFD_f - I$ fragments items of sizes $s(a_i) \geq \frac{U}{2}$, they are also fragmented by the optimal packing. We conclude that in order to create a difference of more than one overhead unit, between the optimal packing and the packing of $FFD_f - I$, two items of size $s(a_i) \geq 6$ must be packed in a bin and leave a free space of at least 3. To do so, the

bin size must satisfy $U \geq 15$. However, in the case of U=15, items of size 5 are packed without fragmentation and therefore the value $\frac{U}{U-1}$ is an upper bound on the asymptotic performance ratio for $U \leq 15$.

(ii) $U \geq 16$: We first prove the lower bound and then the upper bound.

Claim 2.4 *For the $FFD_f - I$ algorithm -* $\frac{U}{U-1} < R^{\infty}_{FFD_f - I}$.

Proof : For each value $U \geq 16$, we present an example where the performance ratio exceeds $\frac{U}{U-1}$. We first consider the case where U is even and then the case where U is odd.

Case 1: Even $U = 2u$. We choose the following item list- L: k items of size $u-2$, then k items of size $u-3$ and finally k items of size 5. The optimal packing fills k bins each with one item of each size, $OPT(L) = k$. When $FFD_f - I$ is applied to L the first k bins contain all items except for the last $k/4$ items of size 5. In the best case (U is a multiple of 5) no more overhead is produced and $\frac{5k}{4U}$ more bins are used. The total number of bins required by the algorithm is: $FFD_f - I(L) \geq (k + 5k/4U)$. The performance ratio in this case is:

$$R^{\infty}_{FFD_f - I} \geq \frac{4U+5}{4U} > \frac{U}{U-1}, \quad \forall\, even\ U \geq 16. \tag{7}$$

Case 2: Odd $U = 2u+1$. This time we choose the following item list: k items of size $u-1$, then k items of size $u-3$ and finally k items of size 5. The packing is similar to the one in the last example and the total number of bins used by the algorithm is: $FFD_f - I(L) \geq (k + 7k/6U)$. The example for the odd case provides a slightly lower bound:

$$R^{\infty}_{FFD_f - I} \geq \frac{6U+7}{6U} > \frac{U}{U-1}, \quad \forall\, odd\ U \geq 17. \tag{8}$$

We now turn to the upper bound on the performance ratio. We show that for any bin size U, the performance ratio is smaller than $\frac{U}{U-2}$, which is the upper bound set by Lemma 1.

Claim 2.5 *For the $FFD_f - I$ algorithm -* $R^{\infty}_{FFD_f - I} < \frac{U}{U-2}$.

Proof : In order to prove the claim we show that the algorithm cannot produce a packing where each bin (maybe except for a negligible number) contains two units of overhead. The *border bins property* implies that just before the first item is fragmented, the bins are arranged in long sequences where the free space in all bins is equal. The items are also packed in long sequences. Let us assume the free space in a sequence is x and the size of the items packed is y. Obviously $x < y$ otherwise the items are not fragmented. Note that when an item is fragmented over more than two bins, only the first and last bins can contain two units of overhead. Therefore it is enough to consider only the case where an item is fragmented over two bins. Assume that item number k is fragmented over two bins such that bin number 1 contains a fragment of size α and bin number 2 contains a fragment of size $y - \alpha$. The next item (number $k+1$) is also fragmented and a fragment of size $x - y + \alpha - 2$ is packed in bin number 2. In order to create a repeated cycle of fragmentations the size of the first fragment of each item must be equal, that is $\alpha = x - y + \alpha - 2$. This is true for $y = x - 2$, but for that value the items are not fragmented in the first place. Since we can not create a repeated cycle of equal size fragmentations, the size of the first fragment packed in a bin increases, until it reach the size of the bin, in which case only one unit of overhead is packed in the bin. This

means that at least one of every y (the size of the fragmented item) bins contains only one unit of overhead. Since $y \leq U/2$ we can establish that:

$$R^{\infty}_{FFD_f-I} \leq \frac{U}{U-2+\frac{2}{U}} < \frac{U}{U-2} \tag{9}$$

This concludes the proof of Theorem 2. $\triangle$

The improvement of $FFD_f - I$ over NF_f is significant for low values of bin size ($U \leq 15$), which are the most meaningful values (since the performance ratio is decreasing with the bin size). Moreover, $FFD_f - I$ is superior for any value of U. We make the following remarks:

- For $U \geq 16$, equations (17) and (18) enable us to set a tighter bound on the performance ratio, than what is stated in the theorem.
- For each of the following values: $U \in \{7, 9, 10, 11, 13, 14, 15\}$, it is possible to find an example where the ratio is $\frac{U}{U-1}$, hence $R^{\infty}_{FFD_f-I} = \frac{U}{U-1}$. We show an example for $U = 7$. The list L contains k items of size 3 and $2k$ items of size 2. The optimal packing packs one item of size 3 with two items of size 2 in each bin, therefore $OPT(L) = k$. The $FFD_f - I$ algorithm packs $k/2$ bins with two items of size 3 and $2k/3$ bins with three items of size 2. There are no fragmentations, but each bin contains a wasted space of 1 unit, that cannot be used. The total number of bins is $FFD_f - I(L) = 7k/6$. It is not difficult to develop similar examples for other values of U in this range. The value $U = 12$ is an exception since the performance ratio is slightly lower. The worst list we could find is made of items of sizes 8, 3 and 2 for which we have: $R^{\infty}_{FFD_f-I} = \frac{16}{15}$.
- When $U \leq 5$, $R^{\infty}_{FFD_f-I} = 1$, however when $U \in \{6, 8\}$ $R^{\infty}_{FFD_f-I} > 1$. This is interesting, since for the classical problem the performance ratio of FFD is: $R^{\infty}_{FFD} = 1$. As an example for $U = 6$ we choose a list L with $15k$ items of size 4 and $12k$ items of size 3. The optimal packing fragments items of size 4 and uses $OPT(L) = 18k$ bins. The $FFD_f - I$ algorithm fragments items of size 3 and as a result $FFD_f - I(L) = 18.5k$. As an example for $U = 8$ choose a list with $7k$ items of size 6 and $4k$ items of size 4.
- We may define algorithm BFD_f-I in a similar way to $FFD_f - I$, only this algorithm is based on the Best-Fit Decreasing heuristic. We can show that the $BFD_f - I$ algorithm has the same performance ratio as $FFD_f - I$.

3 Packing Variable Size Bins with Item Fragmentation

In this section we look into a problem containing two important elements of the scheduling problem: bins of different sizes and a fixed number of bins. We are given a set B, of m variable size bins and a list L, of n items. The goal is to maximize the sum of items packed from the list.

The problem is a variant of the multiple knapsack problem (MKP), see e.g., [MT90]. In MKP each item $a_i \in L$ has a size $s(a_i)$ and a profit $p(a_i)$. The goal is to pack items of maximum profit into variable size bins. In the variant we define here, the profit of each item equals its size, i.e., $p(a_i) = s(a_i)$. The problem is also related to variants of variable size bin packing, although it is different from previously studied variants. Two known variants are particularly relevant. The first variant, presented in [FL86], defines the problem as follows: we are given a

list of items and a set of bin sizes. There is an infinite supply of bins of each size. The goal is to pack the items into a set of bins, such that the sum of all bin sizes is minimized. Our problem is different, since the set of bins is fixed. The second variant appears in [Lan84] and [FK88] where the problem is defined in a similar way to ours but the goal is to pack as many items as possible. This variant corresponds to the case where the profit of each item is $p(a_i) = 1$.

Bin packing problems (in contrast with knapsack problems) assume that all items must be packed and there is no effort to optimize the selection of a feasible subset. As a result the performance of any bin packing algorithm is going to be primarily effected by the subset selection. We make the following assumption to avoid the effect of item selection:

Feasible List Assumption: An optimal packing can pack all items in L into the set of bins B (the optimal packing may require to fragment some of the items).

Note that although the assumption tells us that optimally all items can be packed, a specific algorithm may not be able to pack all items. Another important assumption is that the algorithm is allowed to pack only part of an item. That is, packing one fragment of an item but leaving the other fragment unpacked is legal.

Problem Definition 2: We are given a list of m bins B and a *feasible list* of n items L. Each item $a_i \in L$ has a size $s(a_i) \in Z^+$ and each bin $B_j \in B$ has size $s(B_j) \in Z^+$. The goal is to maximize the total sum of items packed in B. When packing a fragment of an item one unit of overhead is added to the size of every fragment.

Performance Ratio: For a given list L and algorithm A, let $c(A, L)$ be the sum of items packed form list L by algorithm A, let $c(OPT, L)$ denote the sum of items packed from L by an optimal packing. Define the ratio: $R_A(L) \equiv c(OPT, L)/c(A, L)$. The asymptotic worst case performance ratio R_A^∞ is defined to be:

$$R_A^\infty \equiv \inf\{r \geq 1 : \text{ for some } N > 0,\ R_A(L) \leq r \text{ for all } L \text{ with } OPT(L) \geq N\} \qquad (10)$$

As in the previous section we are going to consider only algorithms that avoid unnecessary fragmentations (see Definition 1).

We found out in Section 2 that in the case of equal size bins, the performance ratio is a factor of the bin size. As we shell see, in the case of variable size bins the performance ratio is determined by the average bin size: $\overline{U} = \frac{1}{m}\sum_{j=1}^{m} s(B_j)$. We note that the average bin size is not necessarily an integer.

Lemma 2 *For any algorithm A that prevents unnecessary fragmentations:*

$$R_A^\infty \leq \frac{\overline{U}}{\overline{U} - 2}, \quad \overline{U} > 2.$$

Proof: Assume algorithm A produces the worst possible packing. When packing m bins there could be at most $m-1$ fragmentations. Each fragmentation may add at most two overhead units. The content of the bins after A is done is:

$$c(A, L) \geq \sum_{j=1}^{m} s(B_j) - 2(m-1) > m(\overline{U} - 2)$$

The best an optimal packing can do is to fill the bins without fragmentations:

$$c(OPT, L) \leq \sum_{j=1}^{m} s(B_j) = m\overline{U}$$

If we compare the worst possible packing of algorithm A, to the best possible packing, we get the ratio of Lemma 2:

$$R_A^\infty \leq \frac{m\overline{U}}{m(\overline{U}-2)} = \frac{\overline{U}}{\overline{U}-2}$$

$\triangle$

We now find the performance ratio of the NF_f and FFD_f algorithms. We point out that Lemma 2 holds for any value of $\overline{U}$, however we are going to consider only cases where $\overline{U}$ is an integer. We use Lemma 2 as an upper bound and establish lower bounds on the performance ratio of the algorithms. Note that the case of equal size bins is a special case of the current problem. Therefore we can use the examples of the previous section as worse case examples. This means the performance ratio of the algorithms can only worsen when the bins are of variable size.

Claim 3.1 *For the NF_f algorithm $R_{NF_f}^\infty = \frac{\overline{U}}{\overline{U}-2}$, $5 \leq \overline{U}$, $\overline{U} \in Z^+$.*

Proof : The upper bound is given by Lemma 2. To show that this is also the lower bound, we use the examples of the previous section. We showed an example for every $6 \leq U$ where each bin contains 2 units of overhead. As a worst case example for the case of $\overline{U} = 5$ choose m bins of size 5 and an item list with m items of size 3 followed by m items of size 2. Only the items of size 3 are packed by algorithm NF_f, therefore the performance ratio is: $R_{NF_f}^\infty \Big|_{\overline{U}=5} = \frac{5}{3} = \frac{\overline{U}}{\overline{U}-2}$.

$\triangle$

We now turn to the FFD_f algorithm. Here the worst case performance ratio is effected more severely.

Theorem 3 *For the FFD_f algorithm the performance ratio satisfy:*

$$\frac{\overline{U}}{\overline{U}-1.25} \leq R_{FFD_f}^\infty \leq \frac{\overline{U}}{\overline{U}-2}\,, \quad 8 \leq \overline{U},\ \overline{U} \in Z^+.$$

Proof : The upper bound is given by Lemma 2. To establish a lower bound we present the following example: Let there be $m = 2k$ bins, of which k are the size of $\overline{U}+2$ units and another k are the size of $\overline{U}-2$ units. Clearly the average bin size is $\overline{U}$. Let us assume the bins are arranged in decreasing order (similar examples can be developed for increasing order). As an item list we choose $3k$ items, k from each of the sizes $\overline{U}-2$, $\overline{U}-3$ and 5. In the optimal packing each bin of size $\overline{U}+2$ contains one item of size $\overline{U}-3$ and one of size 5. Therefore all the bins are full and we have $c(OPT, L) = 2k\overline{U}$. The FFD_f algorithm first packs each item of size $\overline{U}-2$ into a bin of size $\overline{U}+2$, then each item of size $\overline{U}-3$ into a bin of size $\overline{U}-2$. In the last stage of the packing items of size 5 are fragmented over the free space in the bins of size $\overline{U}+2$. The content of the bins, when the algorithm terminates, is: $c(FFD_f, L) = k(2\overline{U}-2.5)$. The example shows that $R_{FFD_f}^\infty \geq \frac{\overline{U}}{\overline{U}-1.25}$, $8 \leq \overline{U}$, $\overline{U} \in Z^+$. $\triangle$

Considering the cases of $\overline{U} \leq 7$, we find that the performance ratio is 1 for values $\overline{U} \leq 4$. For values $5 \leq \overline{U} \leq 7$ the performance ratio satisfies: $R_{FFD_f}^\infty \geq \frac{\overline{U}}{\overline{U}-1}$.

Remarks:

1. The difference between the cases of equal and variable size bins, is significant for small values of U. For example consider the case of $\overline{U} = 5$. For equal size bins ($U = 5$) the performance ratio is $R^{\infty}_{FFD_f} = 1$. For variable size bins it is: $R^{\infty}_{FFD_f} = \frac{5}{4}$.
2. The difference between the two cases is not as significant for values of $U > 15$, since in both cases the performance ratio is not far from upper bound.

4 Bin Packing with Size-Preserving Fragmentation

In this section we study a different fragmentation cost function in which fragmenting an item does not increase its size. Instead, we assume that packing an item is associated with a cost and fragmentation increases this cost. The cost of packing an item (or fragmenting it) depends on the application. As an example, take the scheduling problem but assume that fragmenting a datagram does not increase its size, because the format of the datagram already includes the fragmentation fields. On the other hand fragmentation requires additional resources from the system (CPU, memory) and takes longer to process. In other applications, such as in stock-cutting problems, it may simply cost to fragment an item (cut a piece of pipe for example) or put it back together. We proceed to formally define the problem.

Bin Packing with Size-Preserving Fragmentation (BP-SPF): We are given a list of n items $L = (a_1, a_2, ..., a_n)$, each with a size $s(a_i) \in \{1, 2, ..., U\}$ and a cost $c(a_i) \in Z^+$. The items must be packed into m identical bins, of size U. It is possible to fragment any item, in which case one unit is added to its cost but does not change its size. The goal is to minimize the total cost.

Denote by $s(L)$ and $c(L)$ the total size and cost of all items, respectively. To ensure all items can be packed, we assume $s(L) \leq mU$.

Performance: There are several ways to evaluate the performance of an algorithm for the problem. We observe that since the cost of fragmentation is not related to the size or cost of an item, the additional cost of an algorithm depends only on the number of fragmentations it performs. We therefore chose to evaluate the performance of an algorithm by its *overhead*. For a given list L and algorithm A, let $c(A, L)$ be the total cost of algorithm A, let $c(OPT, L)$ denote the optimal (minimal) cost and define the overhead of A as: $OH_A(L) \equiv c(A, L) - c(OPT, L)$. For the case of $OPT(L) = m$, we define the worst case overhead of algorithm A, OH^m_A, as:

$$OH^m_A \equiv \inf\{h : \ OH_A(L) \leq h \quad \text{for all } L \text{ with } OPT(L) = m\}. \tag{11}$$

We first show that the complexity of BP-SPF is similar to that of BP-SIF.

Claim 4.1 *BP-SPF is NP-hard in the strong sense.*

Proof: The proof is similar to that of BP-SIF. We define the decision version D(BP-SPF) as follows: given a list of items L, a size $s(a) \in Z^+$ and a cost $c(a) \in Z^+$ for each item $a \in L$, a positive integer bin capacity U and two positive integers, k and C, is there a feasible packing of L in k bins of size U such that the total cost is C? Any instance I of 3-PARTITION can be polynomially transformed into an equivalent instance I' of D(BP-SPF) by setting U=B, $K = m$ and $C = c(L)$ (which implies that for any "yes" instance all n items are packed without fragmentation). △

We consider only algorithms that prevent unnecessary fragmentation (see Definition 1). Such algorithms have the following property.

Lemma 3 *For any algorithm A, that prevents unnecessary fragmentations* - $OH_A^m \leq m-1$.

Proof: Since A prevents unnecessary fragmentations, it may perform at most $m-1$ fragmentations when packing m bins. The maximum cost of algorithm A is therefore: $c(A,L) = c(L) + (m-1)$. Clearly $c(OPT,L) \geq c(L)$, which means that for any list L: $OH_A(L) \leq m-1$. $\triangle$

We now examine the performance of the NF_f and FFD_f algorithms (defined in Sections 2.1 and 2.2, respectively). We show that the performance of the NF_f algorithm is the worst possible while FFD_f performs better.

Theorem 4 *The overhead of algorithm NF_f for every $m \geq 2$ is* - $OH_{NF_f}^m = m-1, \quad \forall 2 \leq U$.

Proof: Lemma 3 provides an upper bound. We show that this is also the lower bound. As a worst case example choose the following list of items: one item of size $U-1$ followed by $m-1$ items of size U. The optimal packing causes no fragmentations by packing each item in a bin. The NF_f algorithm fragments all the items of size U. The total overhead is therefore $m-1$. $\triangle$

We now turn to the FFD_f algorithm. We expect FFD_f to perform better than NF_f and this is indeed true when the bin size U is small. However, we show that if the bin size is not bounded, the worst case overhead of FFD_f is the maximum possible, that is, there exist a list L for which, for any value of m, $c(FFD_f,L) - c(OPT,L) = m-1$.

Claim 4.2 *For the FFD_f algorithm, for every $m \geq 2$ and $2m+16 < U$* - $OH_{FFD_f}^m = m-1$.

Proof: We choose a bin size satisfying: $U > 2m+16$. The list, L, is made of k repetitions of the following set: $L' = \{U/2+2,\ U/2+1,\ U/4+2,\ U/4+1,\ U/4-3,\ U/4-3\}$. Two bins of size U are needed to pack L', therefore $m = 2k$ bins are needed to pack L. The optimal packing cause no fragmentations by packing k bins with items of size $U/2+2$, $U/4+1$ and $U/4-3$, the other k bins are packed with items of size $U/2+1$, $U/4+2$ and $U/4-3$ (see Figure 3). The FFD_f algorithm packs the items in the following way: The first $5k$ items are packed without fragmentations. The free space in the first k bins is $U/4-4$, in the remaining k bins the free space is 1 (see Figure 3). Since the free space in all bins is smaller than the size of the items, the items are fragmented. Each item except the last is fragmented over two bins. When packing the last item, bins B_1,...B_{k-1} are full, bin B_k has free space of $\frac{U}{4}-k-3$ and each bin B_{k+1},...,B_{2k} has free space of 1. As a result, the last item is fragmented over the remaining free space, causing k more fragmentations. The total number of fragmentations is $m-1$ and the overhead is therefore $OH_{FFD_f}^m = m-1$. $\triangle$

To hold for every value of m the above claim requires an unbounded bin size. Since we are mainly interested in asymptotic behavior, i.e., $U \ll m$, this is clearly not a practical assumption. For the more reasonable cases where $m > U$ the overhead of algorithm FFD_f is less than $m-1$.

Claim 4.3 *For any bin size U there is $N > 0$ for which the overhead of FFD_f satisfies* - $OH_{FFD_f}^m < m-1, \quad \forall N < m$.

Proof: We show that if $U \ll m$ then $c(A,L) < c(L) + (m-1)$. Recall that when analyzing the $FFD_f - I$ algorithm (subsection 2.2) we proved the border bin property. The property tells us that before the first item is fragmented the bins are ordered in long sequences of equal content bins. Note that by increasing m while keeping U constant, we can create sequences of

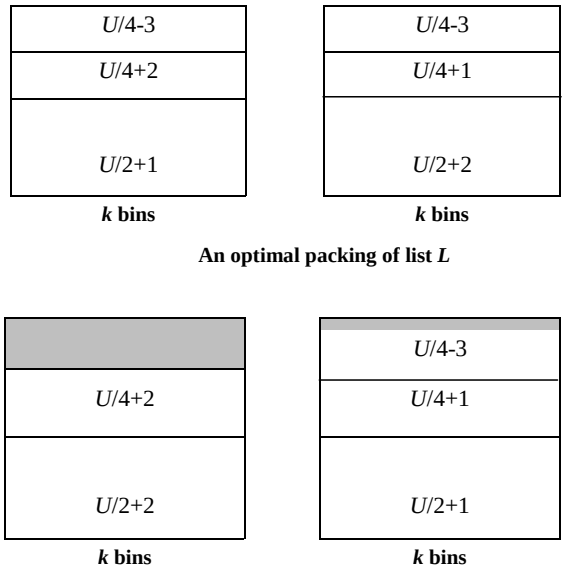

Figure 3: Worst case example for algorithm FFD_f.

any length. Consider the moment before FFD_f fragments the first item. Let us assume the free space in a sequence is x and the size of the items packed is y, where $x < y$. Clearly at least one out of every $x \cdot y$ bins is closed without fragmentation. The overhead is therefore always smaller than $m - 1$. $\triangle$

5 Average Case Analysis

In Section 2 we presented worst case analysis of several algorithms for the problem of bin packing with size increasing fragmentation. Worst case analysis provides an upper bound on the performance ratio of an algorithm, but from a practical point of view it may be too pessimistic, since the worst case may happen very rarely. A different approach for estimating the performance of an algorithm, is an average case analysis. In this case we assume the items are taken from a given distribution H and we try to estimate the performance ratio of an algorithm, when it is applied to a list taken from that distribution. For a given algorithm A and a list of n items L_n, generated according to distribution H, the expected performance ratio is defined as follows:

$$\overline{R}_A^n(H) \equiv E\left[R_A(L_n)\right] = E\left[\frac{A(L_n)}{OPT(L_n)}\right] \tag{12}$$

The asymptotic expected performance ratio is defined as:

$$\overline{R}_A^\infty(H) \equiv \lim_{n\to\infty} \overline{R}_A^n(H) \tag{13}$$

In this section we present average case analysis of the NF_f algorithm. Let us define the problem in a formal way.

Average case analysis of BP-SIF: We are give a list of n items $L = (a_1, a_2, ..., a_n)$, each independently chosen from the finite set $s(a_t) \in \{1, 2, ..., U\}$. The probability to choose an item of size i is h_i, i.e., for all t: $h_i = Pr(s(a_t) = i)$. The goal is to pack the items into a minimum number of bins of equal size- U. When packing a fragment of an item, one unit of overhead is added to the size of *every* fragment.

The first average case analysis of the NF algorithm was done by Coffman, So, Hofri and Yao [CSH80] who showed that the asymptotic expected performance ratio for continuous uniform distribution is: $\overline{R}_{NF}^\infty = \frac{4}{3}$. For discrete uniform distribution, in the range $s(a_t) \in \{1, 2, ..., U\}$, it has been shown in [CHJ93] that the NF algorithm has the following asymptotic expected performance ratio:

$$\overline{R}_{NF}^\infty = \frac{2(2U+1)}{3(U+1)}. \tag{14}$$

Note that the result for the continuous uniform distribution is reached when $U \to \infty$.

The above mentioned results were achieved by using different techniques, all of which are fairly complicated (see, for example [CSH80], [Hof84] and [Kar82]). We present here, what we believe to be a much easier method of calculating the asymptotic expected performance ratio. We first use this method to repeat the analysis of the NF algorithm. We next apply it to the new problem of bin packing with item fragmentation, to find the expected performance ratio of the NF_f algorithm.

5.1 Average Case Analysis of the NF Algorithm

We use a Markov chain to describe the packing of the algorithm. The state of the algorithm, which we denote by N_t, is the content of the open bin after t items were packed. Since the bin size is U and there are n items to pack, the possible states of the algorithm are $1 \le N_t \le U$, $1 \le t \le n$. The probability distribution for N_{t+1} is completely determined by the value of N_t,

which renders the process a Markov chain. We consider only the cases where the Markov chain describing the algorithm is ergodic. Note that this is very reasonable since the chain is finite and for most item size distributions it would also be irreducible and acyclic, hence ergodic. If the chain is not ergodic, it is necessary to apply the analysis we present on the irreducible part of the chain which is accessible from the initial state (empty bins).

Assume $N_{t-1} = j$ and the algorithm now packs item a_t. If the open bin cannot contain the item, i.e., $j + s(a_t) > U$, the item is packed in a new bin. The previous open bin contains $U - j$ unused units which we call overhead units. We say that the overhead units "increased" the size of a_t and define its ***combined-size*** to be the actual size of the item, plus the overhead units it created. For example, say the algorithm is in state $N_t = 2$ and the next item is of size U. The overhead in this case is $U - 2$ units and we say the *combined size* of the item is: $U + U - 2$.

Denote by oh_t the overhead added to the size of item a_t. For an algorithm A and a list L_n of n items, we define the expected average combined size of all items to be:

$$I_{av}^{n}(A) \equiv E\left[\frac{1}{n}\sum_{t=1}^{n}(s(a_t) + oh_t)\right] \tag{15}$$

We define the expected *asymptotic average combined size* of all items as:

$$I_{av}(A) \equiv \lim_{n\to\infty} I_{av}^{n}(A) \tag{16}$$

We can express the asymptotic expected number of bins required by A as:

$$E\left[\lim_{n\to\infty} A(L_n)\right] = \frac{n \cdot I_{av}(A)}{U} \tag{17}$$

We now use a property of the optimal packing that ensures that for any item size distribution the tails of the distribution of $OPT(L_n)$ decline rapidly enough with n [RT87], so that as $n \to \infty$, $E[A(L_n)/OPT(L_n)]$ and $E[A(L_n)]/E[OPT(L_n)]$ converge to the same limit [CL91]. Therefore the asymptotic expected performance ratio is given by:

$$\begin{aligned} \overline{R}_A^{\infty} &= \lim_{n\to\infty} E\left[\frac{A(L_n)}{OPT(L_n)}\right] = E\left[\frac{\frac{U}{n}\lim_{n\to\infty} A(L_n)}{\frac{U}{n}\lim_{n\to\infty} OPT(L_n)}\right] \\ &= \frac{E\left[\frac{U}{n}\lim_{n\to\infty} A(L_n)\right]}{E\left[\frac{U}{n}\lim_{n\to\infty} OPT(L_n)\right]} = \frac{I_{av}(A)}{I_{av}(OPT)} \end{aligned} \tag{18}$$

To find the asymptotic expected performance ratio of the NF algorithm, we must calculate both $I_{av}(OPT)$ and $I_{av}(NF)$. Since bin packing is NP-hard, we can not expect to find $I_{av}(OPT)$ for all item size distributions. Fortunately, we do know that for several important distributions, including the uniform distribution, the overhead of the optimal packing can be neglected [CGJ96]. For such distributions we have:

$$I_{av}(OPT) = \sum_{i=1}^{U} i \cdot h_i \tag{19}$$

To find $I_{av}(NF)$ we use the Markov chain describing the algorithm. Denote by P the transition matrix of the Markov chain and by $\Pi = (\Pi_1, ...\Pi_U)$ the equilibrium probability

vector satisfying $\Pi = \Pi P$. Assume NF packs a long list of n items; denote by n_j the number of visits in state j during the packing. Since we consider ergodic chains, we have the following property:

$Pr\left(\lim_{n\to\infty} \frac{n_j}{n} = \Pi_j\right) = 1$, which is usually written as: $\lim_{n\to\infty} \frac{n_j}{n} = \Pi_j$, *a.s. (almost surely).*

We now denote by $n_{j,i}$ the number of items of size i packed when the algorithm is in state j. The probability for the next item in the list to be of size i, h_i, is unrelated to the state of the algorithm. Therefore we can use the Law of large numbers to establish the following property of $n_{j,i}$:

$$\lim_{n\to\infty} \frac{n_{j,i}}{n} = \lim_{n\to\infty} \frac{n_j}{n} \cdot h_i = \Pi_j \cdot h_i\,, \quad a.s. \tag{20}$$

The overhead added to each item is related to both the state of the algorithm and the size of the item. We denote by $oh_i(j)$ the overhead added to an item of size i which is packed when the algorithm is in state j. We calculate the average combined size of the items in the following way:

$$\begin{aligned} I_{av}(NF) = \lim_{n\to\infty} I^n_{av}(NF) &= \lim_{n\to\infty} E\left[\frac{1}{n}\sum_{j=1}^{U}\sum_{i=1}^{U} n_{j,i}\cdot(i+oh_i(j))\right] \\ &= E\left[\sum_{j=1}^{U}\sum_{i=1}^{U}\lim_{n\to\infty}\frac{n_{j,i}}{n}\cdot(i+oh_i(j))\right] \end{aligned} \tag{21}$$

Substituting (20) we get:

$$I_{av}(NF) = \sum_{j=1}^{U}\sum_{i=1}^{U} \Pi_j \cdot h_i \cdot (i + oh_i(j)) \tag{22}$$

To simplify (22) we use the following definitions:

$$\begin{array}{ll} \overline{h} \equiv \sum_{i=1}^{U} i \cdot h_i & \text{average size of items (without overhead)} \\ OH(j) \equiv \sum_{i=1}^{U} h_i \cdot oh_i(j) & \text{average overhead in state } j \\ \overline{OH} \equiv \sum_{j=1}^{U} \Pi_j \cdot OH(j) & \text{average overhead size} \end{array} \tag{23}$$

Equation (22) now becomes:

$$\begin{aligned} I_{av}(NF) &= \sum_{j=1}^{U}\Pi_j \cdot \sum_{i=1}^{U} i\cdot h_i + \sum_{j=1}^{U}\Pi_j \cdot \sum_{i=1}^{U} h_i \cdot oh_i(j) \\ &= \overline{h} + \sum_{j=1}^{U}\Pi_j \cdot OH(j) = \overline{h} + \overline{OH} \end{aligned} \tag{24}$$

The expression in (24) is very intuitive: the asymptotic average combined size of the items is made of the average size of the items plus the average size of the overhead. To calculate the expected performance ratio we must find two components:

1. The equilibrium probabilities of the Markov chain, Π. We find Π by constructing the transition matrix P and calculating the equilibrium probability vector satisfying: $\Pi = \Pi P$.

2. The overhead components $oh_i(j)$. This is easily obtained from the packing rules of the algorithm.

Our technique of average case analysis has two advantages: it is suitable for analyzing any (i.i.d) item size distribution, both discrete and continuous, and the calculation is relatively easy. These properties are important since in most real-world applications of bin packing the items are drawn from a *finite* set, which is rarely uniform.

In the next subsection we calculate specific results for the case of discrete uniform distribution.

5.1.1 Discrete Uniform Distribution

Discrete uniform distribution means that $h_i = \frac{1}{U}$, $\forall i$. It is easy to see that in this case the Markov chain is ergodic. An important characteristic of the discrete uniform distribution is that the overhead of the optimal packing is negligible. To state it formally, let L_n be a list of n items drawn from a discrete uniform distribution H and let $s(L_n)$ be the total size of all items in L_n. The expected wasted space of the optimal packing has the following property [CCG00]:

$$\overline{W}^n_{OPT}(H) = E\left[U \cdot OPT(L_n) - s(L_n)\right] = O\left(\sqrt{n}\right)$$

Based on the above result it is clear that $\lim_{n\to\infty} E\left[OPT(L_n)/s(L_n)\right] = 1$, which means we can neglect the overhead of the optimal packing in calculating the asymptotic expected performance ratio. Therefore for the optimal packing we have:

$$I_{av}(OPT) = \sum_{i=1}^{U} i \cdot h_i = \frac{1}{U}\sum_{i=1}^{U} i = \frac{U+1}{2} \tag{25}$$

We use (24) to find the average combined size of the items. We first find the equilibrium probabilities of the Markov chain. Let P be the $U \times U$ transition matrix describing the chain and let $\Pi = (\Pi_1, ..., \Pi_U)$ be the equilibrium probability vector satisfying: $\Pi = \Pi P$. There is a symmetry in the lines of the transition matrix P, in a sense that line j and line $U - j$ are identical. For $j \le \left\lfloor \frac{U}{2} \right\rfloor$ we have:

$$P_{j,k} = \frac{1}{U} \cdot \left\{ \begin{array}{ll} 0 & 1 \le k \le j \\ 1 & j < k \le U - j \\ 2 & U - j < k \le U \end{array} \right. \quad 1 \le j \le \left\lfloor \frac{U}{2} \right\rfloor \tag{26}$$

The last line is: $P_{U,k} = \frac{1}{U},\ 1 \le k \le U$

As an example we present the matrix P for the case of $U = 10$:

$$P = \frac{1}{10} \cdot \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{27}$$

The simple structure of the matrix P enables an easy solution to the set of equations $\Pi = \Pi P$.

$$\Pi_j = Pr(N = j) = \frac{2j}{U(U+1)} \tag{28}$$

Next we compute the overhead component $OH(j)$. It is easy to verify that for the NF algorithm the average overhead in state $N = j$ is:

$$OH(j) = \sum_{i=1}^{U} h_i \cdot oh_i(j) = \sum_{i=U-j+1}^{U} \frac{1}{U} \cdot (U - j) = \frac{j(U-j)}{U} \tag{29}$$

We now use (28) and (29) to find the average combined size of the items:

$$\begin{aligned} I_{av}(NF) &= \frac{U+1}{2} + \sum_{j=1}^{U} \Pi_j \cdot OH(j) = \frac{U+1}{2} + \sum_{j=1}^{U} \frac{2j}{U(U+1)} \cdot \frac{j(U-j)}{U} \\ &= \frac{U+1}{2} + \sum_{j=1}^{U} \frac{2j^2\,(U-j)}{U^2\,(U+1)} = \frac{U+1}{2} + \frac{U-1}{6} = \frac{2U+1}{3} \end{aligned} \tag{30}$$

We use $I_{av}(NF)$ and $I_{av}(OPT)$ to obtain the asymptotic expected performance ratio:

$$\overline{R}_{NF}^{\infty} = \frac{I_{av}(NF)}{I_{av}(OPT)} = \frac{(2U+1)/3}{(U+1)/2} = \frac{2(2U+1)}{3(U+1)} \tag{31}$$

The result for the asymptotic expected performance ratio is in accordance with known results (see [CHJ93]). The asymptotic worst case performance ratio is:

$$R_{NF}^{\infty} = \frac{2U}{U+1}, \quad \text{for every } \; U \geq 2.$$

We compare the two, for several values of U, in Table 1.

In both cases the performance ratio increases with the bin size. There is a dramatic difference between the worst case ratio and the expected ratio. The average case results are therefore not as bad as the worst case analysis indicates.

Table 1: Expected and worst case asymptotic performance ratio of the NF algorithm.

Bin Size U	**Worst case ratio** R_{NF}^{∞}	**Average case ratio** $\overline{R}_{NF}^{\infty}$
3	1.5	1.166...
4	1.6	1.2
6	1.714...	1.238...
10	1.818...	1.272...
100	1.980...	1.326...
∞	2	1.3333...

5.2 Average Case Analysis of the NF_f Algorithm

We now use the same method we used for analyzing the NF algorithm, to analyze the case where item fragmentation is allowed, i.e., the NF_f algorithm. In this section we assume the items are taken from a discrete uniform distribution, that is: $h_i = \frac{1}{U}$, $\forall i$. Note that the overhead this time is due to fragmentation or an unused free space (if the content of a closed bin is $U-1$).

The first stage in our analysis of NF_f, is to find the equilibrium probabilities of the Markov chain. Let us describe the components of the transition matrix P:

$$P_{j,k} = \frac{1}{U} \cdot \begin{cases} 0 & k \leq 2, \quad k \leq j \leq U-k \\ 2 & j \in \{k-2, k-1\} \quad 3 \leq k \\ 1 & else \end{cases} \tag{32}$$

As an example we present the matrix P for the case of $U = 10$:

$$P = \frac{1}{10} \cdot \begin{bmatrix} 0 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 & 2 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 2 & 2 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{33}$$

The set of equations defined by $\Pi = \Pi P$ is:

$$\Pi_1 = \frac{1}{U}\Pi_U \tag{34}$$

$$\Pi_2 = \frac{1}{U}(\Pi_1 + \Pi_{U-1} + \Pi_U) \tag{35}$$

$$\Pi_j = \frac{1}{U}(1 + \Pi_{j-1} + \Pi_{j-2}), \quad 3 \leq j \leq U \tag{36}$$

Note that the solution to (36) (if it were the only equation) is: $\Pi_j = \frac{1}{U-2}$. Unlike the case of NF, the solution to the set of equations (34)-(36) is not simple. We therefore defer the calculation of a closed form solution to subsection 5.2.1 and proceed to calculate the average overhead in state $N = j$ - $OH(j)$. Note that when an item is fragmented over two bins, 2 units of overhead are added to it. In state $N = U - 1$ all items of size 2 or more are packed in the next bin, so only 1 unit of overhead is added to them. The average overhead in state $N = j$ is therefore:

$$OH(j) = \frac{1}{U} \cdot \begin{cases} 2\,j & 1 \leq j \leq U-2 \\ U-1 & j = U-1 \\ 0 & j = U \end{cases} \tag{37}$$

We can now express the average combined size of the items:

$$I_{av}\,(NF_f) = \frac{U+1}{2} + \sum_{j=1}^{U} \Pi_j \cdot OH(j) = \frac{U+1}{2} + \frac{U-1}{U}\Pi_{U-1} + \sum_{j=1}^{U-2} \Pi_j \cdot \frac{2j}{U} \tag{38}$$

Similar to (25), the overhead of the optimal packing is negligible: $I_{av}(OPT) = \frac{U+1}{2}$.
The asymptotic expected performance ratio is therefore:

$$\overline{R}^{\infty}_{NF_f} = \frac{I_{av}\,(NF_f)}{I_{av}\,(OPT)} = 1 + \frac{2}{U+1}\left(\frac{U-1}{U}\,\Pi_{U-1} + \sum_{j=1}^{U-2} \Pi_j \cdot \frac{2j}{U}\right) \tag{39}$$

At this point we do not have a closed form solution to the equilibrium probabilities and therefore we cannot present the expected performance ratio in closed form. It is easy, however, to find a numerical solution for every value of U. In Table 2 we present the expected asymptotic performance ratio and the worst case ratio, for several values of U.

Table 2: Expected asymptotic performance ratio of the NF_f algorithm.

Bin Size U	**Worst case ratio** $R^{\infty}_{NF_f}$	**Average case ratio** $\overline{R}^{\infty}_{NF_f}$
3	1.5	1.1666...
4	1.5	1.1961...
5	1.5	1.2097...
10	1.25	1.1676...
20	1.1111...	1.0938...
100	1.0204...	1.0198...
∞	1	1

Recall that, according to Theorem 1, the worst case performance ratio of NF_f is: $R^{\infty}_{NF_f} = \frac{U}{U-2}$. Figures 4 and 5 present the asymptotic expected performance ratio of the NF_f and

NF algorithms together with the worst case performance ratio of NF_f. We observe that the difference between the worst case and the average case for NF_f is not as significant as in NF, that is, the expected performance ratio of the NF_f algorithm is not far from its worst case performance ratio. This is obvious for large values of U since the worst case ratio converges to one, but it also indicates that even under a uniform distribution the NF_f algorithm produces almost the worst possible packing. An interesting question, which we leave open, is whether other, more efficient, algorithms can produce a better packing. In this respect we note that in the case where fragmentation is not allowed, there is a big difference between the performance of NF and the performance of other online algorithms, such as First-Fit and Best-Fit, for which the asymptotic expected performance ratio, for any value of U, is [CGJ96]: $\overline{R}^{\infty}_{FF} = \overline{R}^{\infty}_{BF} = 1$.

Finally we note that it is not difficult to repeat the analysis we presented for a general item size distribution. We begin by calculating the equilibrium probabilities. To that end we construct the transition matrix P and then find the equilibrium probability vector by solving the equation $\Pi = \Pi P$. The next step is to calculate the overhead component $oh_i(j)$ (overhead added to an item of size i packed in state j). The expression for $oh_i(j)$ is given by:

$$oh_i(j) = \begin{cases} 0 & j+i \leq U \text{ or } j = U \\ 2 & j+i > U,\ j \leq U-2 \\ 1 & j+i > U,\ j = U-1 \end{cases} \tag{40}$$

We can now use (24) to calculate the average combined size of the items:

$$I_{av}(NF_f) = \sum_{j=1}^{U} \Pi_j \cdot \sum_{i=1}^{U} i \cdot h_i + \sum_{j=1}^{U} \Pi_j \cdot \sum_{i=1}^{U} h_i \cdot oh_i(j) \tag{41}$$

The key problem in deriving the expected performance ratio for a general item size distribution is that, unlike the uniform distribution, $I_{av}(OPT)$ is not always known. In cases where $I_{av}(OPT)$ is not known we can choose to calculate the ratio between the number of bins used by the algorithm, $NF_f(L)$ and $s(L)/U$.

5.2.1 Calculating the Expected Performance Ratio

In this section we derive a closed form solution of the expected performance ratio. Since this closed form solution is rather complex, we also provide an approximation of the expected performance ratio. The approximation is much easier to use and the approximation error, for all but small values of U, is insignificant.

We substitute $\overline{N} = E[N] = \sum_{j=1}^{U} j \cdot \Pi_j$ in (39), to express $\overline{R}^{\infty}_{NF_f}$ in the following way:

$$\overline{R}^{\infty}_{NF_f} = 1 + \frac{2}{U+1}\left(\frac{2}{U}\overline{N} - \frac{U-1}{U}\Pi_{U-1} - 2\Pi_U\right) \tag{42}$$

To calculate the value of $\overline{N}$ we use a generating function: $\Pi(z) = \sum_{j=1}^{U} \Pi_j\, z^j$.

We now use the equilibrium equations (34) - (36) to get:

$$\Pi(z) \;=\; \sum_{j=1}^{U} \Pi_j\, z^j = \Pi_1\, z + \Pi_2\, z^2 + \sum_{j=3}^{U} \frac{1}{U}\left(1 + \Pi_{j-1} + \Pi_{j-2}\right) z^j \tag{43}$$

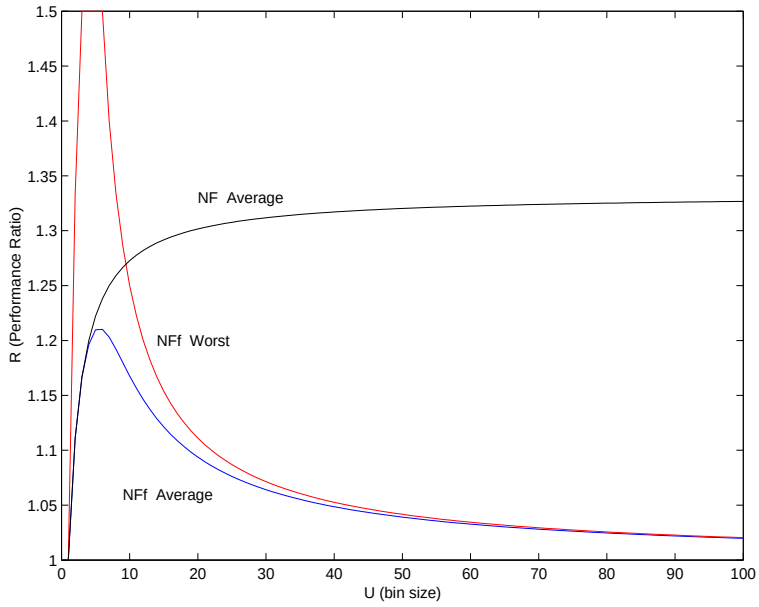

Figure 4: Expected and worst case performance ratio of the NF_f and NF algorithms.

$$= \frac{1}{U}\Pi_U\, z + \frac{1}{U}\left(\Pi_{U-1} + \frac{U+1}{U}\Pi_U\right) z^2 + \sum_{j=3}^{U} \frac{1}{U}\left(1 + \Pi_{j-1} + \Pi_{j-2}\right) z^j$$

Arranging the above expression we get:

$$\left(U - z - z^2\right) \cdot \Pi\left(z\right) = -\,\Pi_U\, z^{U+2} - \left(\Pi_U + \Pi_{U-1}\right) z^{U+1} + \left(\Pi_U + \Pi_{U-1}\right) z^2 + \Pi_U\, z + \sum_{j=3}^{U} z^j$$

$$\Pi\left(z\right) = \frac{\Pi_U\, z^{U+2} + \left(\Pi_U + \Pi_{U-1}\right) z^{U+1} - \left(\Pi_U + \Pi_{U-1}\right) z^2 - \Pi_U\, z - \sum_{j=3}^{U} z^j}{z^2 + z - U} \tag{44}$$

To find $\overline{N}$ we take derivative of the generating function:

$$\begin{aligned} \frac{d\,\Pi(z)}{d\,z} &= \frac{1}{\left(z^2 + z - U\right)^2}\Big[\left(z^2 + z - U\right)\Big[(U+2)\Pi_U\, z^{U+1} \\ &+ (U+1)(\Pi_U + \Pi_{U-1})\, z^U - 2(\Pi_U + \Pi_{U-1})\, z - \Pi_U - \sum_{j=3}^{U} j\, z^{j-1}\Big] \\ &- \left(\Pi_U\, z^{U+2} + \left(\Pi_U + \Pi_{U-1}\right) z^{U+1} - \left(\Pi_U + \Pi_{U-1}\right) z^2 - \Pi_U\, z - \sum_{j=3}^{U} z^j\right)(2z+1)\Big] \end{aligned} \tag{45}$$

The mean is found when $z = 1$:

$$\overline{N} = \frac{d\Pi\left(z\right)}{dz}\Big|_{z=1} = \frac{U\left(U+1\right) - 4U\Pi_U - 2\left(U-1\right)\Pi_{U-1}}{2\left(U-2\right)} \tag{46}$$

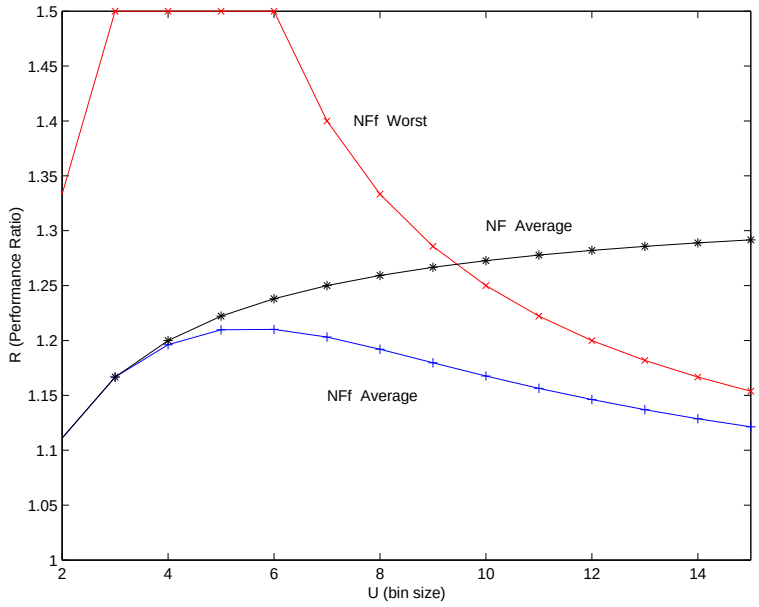

Figure 5: Expected and worst case performance ratio of the NF_f and NF algorithms for values of $U \leq 15$.

Substituting (46) in (42) we get an expression for the asymptotic expected performance ratio:

$$\begin{aligned} \overline{R}^{\infty}_{NF_f} &= 1 + \frac{2}{U+1}\left(\frac{2}{U}\overline{N} - \frac{U-1}{U}\Pi_{U-1} - 2\Pi_U\right) \\ &= 1 + \frac{2}{U+1}\left(\frac{U(U+1) - 4U\Pi_U - 2(U-1)\Pi_{U-1}}{U(U-2)} - \frac{U-1}{U}\Pi_{U-1} - 2\Pi_U\right) \\ &= 1 + \frac{2}{U-2} - \frac{4U^2\Pi_U + 2U(U-1)\Pi_{U-1}}{(U+1)U(U-2)} \end{aligned} \tag{47}$$

We now have an expression which is a function of Π_{U-1} and Π_U:

$$\overline{R}^{\infty}_{NF_f} = \frac{U}{U-2} - \frac{4U\Pi_U + 2(U-1)\Pi_{U-1}}{(U+1)(U-2)} \tag{48}$$

Observe that the first part of the expression is equivalent to the worst case performance ratio. The second part constitutes the difference between the worst case and the average case.

To find the expected performance ratio we must now calculate the probabilities Π_U, Π_{U-1}. We do so by exploring the roots of the generating function given in (44). Note that the denominator is a square polynomial with two roots. Since the generating function is analytic for any value of z, the roots of the denominator are necessarily roots of the numerator also. This information provides two equations from which Π_U and Π_{U-1} can be found. Denote by z_1 and z_2 the roots of the denominator:

$$z_1 = -\frac{1}{2} + \sqrt{U + \frac{1}{4}}, \qquad z_1 = -\frac{1}{2} - \sqrt{U + \frac{1}{4}}$$

Substituting z_1 in the numerator we get:

$$\Pi_U z_1^{U+2} + (\Pi_U + \Pi_{U-1}) z_1^{U+1} - (\Pi_U + \Pi_{U-1}) z_1^2 - \Pi_U z_1 - \sum_{j=3}^{U} z_1^j = 0 \tag{49}$$

Simplifying the above equation we obtain:

$$z_1 (1 + z_1) \left(z_1^U - 1\right) \Pi_U + z_1^2 \left(z_1^{U-1} - 1\right) \Pi_{U-1} = \left(z_1^{U+1} - z_1^3\right) \Big/ (z_1 - 1) \tag{50}$$

We get the same equation if we substitute z_2 in the numerator.

$$z_2 (1 + z_2) \left(z_2^U - 1\right) \Pi_U + z_2^2 \left(z_2^{U-1} - 1\right) \Pi_{U-1} = \left(z_2^{U+1} - z_2^3\right) \Big/ (z_2 - 1) \tag{51}$$

Using (50) and (51) it is now a straightforward algebraic exercise to find Π_U and Π_{U-1}.

$$\Pi_U = \frac{U}{(2-U)} \frac{\sqrt{4U+1}\left((-U)^{U-1} + 1\right) + z_2^{U-2}(2U + z_2(1-U)) + z_1^{U-2}(z_1(U-1) - 2U)}{\sqrt{4U+1}\left((-U)^U - 1\right) + (z_1^U - z_2^U)(U+1)} \tag{52}$$

$$\Pi_{U-1} = \frac{1}{(2-U)} \frac{(z_2 - 1)\left(z_1^{U+1} - z_1^3\right)\left(z_2^U - 1\right) - (z_1 - 1)\left(z_2^{U+1} - z_2^3\right)\left(z_1^U - 1\right)}{\sqrt{4U+1}\left((-U)^U - 1\right) + (z_1^U - z_2^U)(U+1)} \tag{53}$$

To find the expected performance ratio we substitute (52) and (53) in (48).

The expression we obtained for the expected performance ratio enables a calculation for any value of U but does not provide too much insight. To get a better understanding we note that (36) gives us a very good approximation: $\Pi_U = \Pi_{U-1} = \frac{1}{U-2}$. The approximation is getting better the larger U is. Using the approximation we get:

$$\begin{aligned} \overline{R}_{NF_f}^{\infty} &= \frac{U}{U-2} - \frac{4U\Pi_U + 2(U-1)\Pi_{U-1}}{(U+1)(U-2)} \\ &\cong \frac{U}{U-2} - \frac{4U\frac{1}{U-2} + 2(U-1)\frac{1}{U-2}}{(U+1)(U-2)} = \frac{U}{U-2} - \frac{6U-2}{(U+1)(U-2)^2} \end{aligned} \tag{54}$$

Comparing the exact value of the expected asymptotic performance ratio to the approximation, we find that for $U = 7$ the difference is about 0.3%, for $U = 10$ the difference is less than 0.003% and for larger values of U the approximation error is insignificant.

5.2.2 The NF_f Algorithm with r Units of Overhead

In this section we examine a more general case of bin packing with item fragmentation. We now assume that r units of overhead are added to every fragment of an item. The original definition is therefore a special case where $r = 1$. A reasonable assumption is that the bin size is much bigger than the overhead, that is $r \ll U$. We note that the asymptotic worst case performance ratio of NF_f in this case is $R_{NF_f}^{\infty} = \frac{U}{U-2r}$. As we shell see, the average case analysis is very similar to the case where $r = 1$. Before we begin we would like to point the following properties of the problem:

- Items of size $2r$ or less are not fragmented since the size of each fragment would be bigger than the original size of the item.
- If a bin contains $U-2r$ or more units and the next item can not fit in the bin, the bin is closed and the item is packed in a new bin.

The first stage of our analysis is to find the equilibrium probabilities of the Markov chain. To construct the transition matrix P we assume the state is $N_{t-1}=j$ and the next item is of size i, $1 \le i \le U$. There are three possibilities:

1. If $j+i \le U$ the item fits in the bin and the next state is $N_t = j+i$.
2. If $j+i > U$ and $j < U-2r$ the item does not fit in the bin and is therefore fragmented over two bins. The next state is $N_t = j+i+2r-U$.
3. If $j+i > U$ and $j \ge U-2r$ the item does not fit in the bin but it is not fragmented. The next state is $N_t = i$.

Based on the above rules, we can construct the transition matrix P and numerically calculate the equilibrium probability vector Π.

If $U > 4r$ we can describe P in more details. We note the following properties:

1. States $1, ..., 2r$ are exceptional. State $N_t = k$ where $k \le 2r$ can be reached from state $N_{t-1}=j$ only if $k > j$ or $j > U-k$. The transition probability is $\frac{1}{U}$.
2. Any other state $k > 2r$ can be reached from all states. The probability of $N_t = k$ if $N_{t-1}=j$ is $\frac{2}{U}$ if $k-2r \le j \le k-1$, it is $\frac{1}{U}$ otherwise.

The components of the matrix P have the following format:

$$P_{j,k} = \frac{1}{U} \cdot \begin{cases} 0 & k \le 2r, \quad k \le j \le U-k \\ 2 & k-2r \le j \le k-1, \quad 2r < k \\ 1 & else \end{cases} \tag{55}$$

An example for such a matrix for $r=2$ and $U=10$ is given bellow:

$$P = \frac{1}{10} \cdot \begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 & 2 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 & 2 & 2 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{56}$$

The set of equations defined by $\Pi = \Pi P$ is:

$$\Pi_j = \frac{1}{U}\left(\sum_{i=1}^{j-1}\Pi_i + \sum_{i=U-j+1}^{U}\Pi_i\right), \quad 1 \le j \le 2r \tag{57}$$

$$\Pi_j = \frac{1}{U}\left(1 + \sum_{i=j-2r}^{j-1}\Pi_i\right), \quad 2r < j \le U \tag{58}$$

Note that the solution to equation (58) (if it was the only equation) is: $\Pi_j = \frac{1}{U-2r}$. This means that the probabilities $\Pi_1, ..., \Pi_{2r}$ serve as initial values, from which the other steady state probabilities converge to the stable solution. As in the previous analysis, it is simple to get a numeric solution.

We now calculate the average overhead in state $N = j$ - $OH(j)$. Note that when an item is fragmented over two bins, $2r$ units of overhead are added to it. In state $N > U - 2r$ all items of size $2r$ or more are packed in the next bin, therefore the overhead of each item is $U - j$. The average overhead in state j is therefore:

$$OH(j) = \frac{1}{U} \cdot \begin{cases} 2\,r\,j & 1 \le j \le U - 2r \\ \\ j(U-j) & U - 2r < j \le U \end{cases} \tag{59}$$

We can now find the average combined size of the items:

$$I_{av}(NF_f) = \frac{U+1}{2} + \sum_{j=1}^{U-2r}\Pi_j \cdot \frac{2\,j\,r}{U} + \sum_{j=U-2r+1}^{U}\Pi_j \cdot \frac{j(U-j)}{U} \tag{60}$$

For the optimal packing $I_{av}(OPT) = \frac{U+1}{2}$. The asymptotic expected performance ratio is therefore:

$$\overline{R}^{\infty}_{NF_f} = \frac{I_{av}(NF_f)}{I_{av}(OPT)} = 1 + \frac{2}{U+1}\left(\sum_{j=1}^{U-2r}\Pi_j \cdot \frac{2\,j\,r}{U} + \sum_{j=U-2r+1}^{U}\Pi_j \cdot \frac{j(U-j)}{U}\right) \tag{61}$$

We do not have a closed form solution for the equilibrium probabilities and therefore we can not present a closed form expression of the asymptotic expected performance ratio. However, it is easy to compute the expected performance ratio for any value of U and r. In Figure 6 we present the asymptotic expected performance ratio for several values of overhead units r. The top curve is the asymptotic expected performance ratio of the NF algorithm that does not use fragmentation. We can see that when U is smaller than approximately $4r$ the performance ratio of NF and NF_f are the same. We conclude that when $U \le 4r$ fragmentation does not significantly improve the performance of the algorithm. When U becomes larger the performance ratio of improves. As we expect the performance ratio of NF_f is increasing with r but is never more than the performance ratio of the NF algorithm.

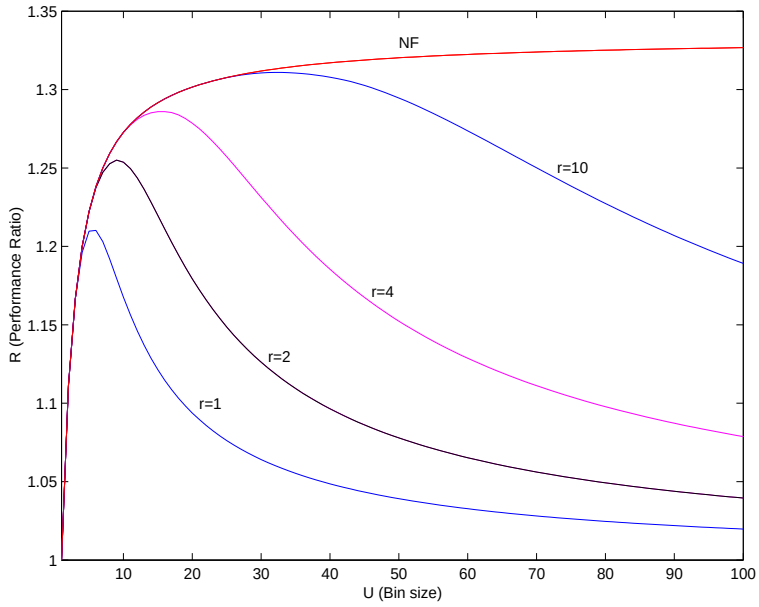

Figure 6: Expected performance ratio of NF_f for several values of r.

References

[CCG00] E. G. Coffman, Jr., C. A. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, and M. Yannakakis. Bin packing with discrete item sizes, Part I: Perfect packing Theorems and the average case behavior of optimal packings. Siam J. Discrete Math., vol. 13 pp. 384-402, 2000.

[CGJ84] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin-packing: An updated survey. In G. Ausiello, M. Lucertini, and P. Serafini, editors, Algorithm Design for Computer System Design, pages 49-106. Springer-Verlag, Wien, 1984.

[CGJ96] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum (ed), PSW publishing, Boston. Approximation Algorithms for NP-Hard Problems, pages 46-93. 1996.

[CHJ93] E. G. Coffman Jr., Shlomo Halfin, Alain Jean-Marie and Philippe Robert. Stochastic analysis of a slotted FIFO communication channel. IEEE Trans. on Info. Theory, vol. 39, No. 5, pp. 1555-1566, 1993.

[CL91] E. G. Coffman, Jr. and G. S. Lueker. Probabilistic Analysis of Packing and Partitioning Algorithms. Wiley, New York, 1991

[CSH80] E. G. Coffman Jr., Kimming So and Micha Hofri. A Stochastic model of bin packing. Information and Control, Vol. 44, pp. 105-115, 1980

[FK88] D. K. Friesen and F. S. Kuhl. Analysis of a hybrid algorithm for packing unequal bins. SIAM J. of Computing, Vol. 17, pp. 23-40, 1988.

[FL86] D. K. Friesen and M. A. Langston. Variable sized bin packing. SIAM J. Comput., 15:222-230, 1986.

[FL91] D. K. Friesen and M. A. Langston. Analysis of a compound bin packing algorithm. SIAM J. Disc. Math, vol. 4, pp.61-79, 1991.

[GJ79] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., San Francisco, 1979.

[Joh74] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. of Computing, 3:299-325, 1974.

[Joh74a] D. S. Johnson. Fast algorithms for bin packing. Journal of computer and system Science. vol. 8, pp. 272-314, 1974.

[Hof84] Micha Hofri. A Probabilistic analysis of the Next-Fit bin packing algorithm. Journal of Algorithms, Vol. 5, pp. 547-556, 1984.

[Kar82] N. Karmarkar. Probabilistic analysis of some bin packing algorithms. Proc. 23^{rd} IEEE FOCS, pp. 107-111, 1982.

[KK82] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In Proc. 23rd Ann. Symp on Foundations of Computer Science, pp. 312-320, 1982.

[Lan84] M. A. Langston. Performance of heuristics for a computer resource allocation problem. SIAM J. Alg. Discrete Methods, 5, pp. 154-161, 1984.

[LL85] C. C. Lee and D. T. Lee. A simple on-line packing algorithm. J. ACM, vol. 32, pp. 562-572, 1985.

[MCG98] Mandal-CA, Chakrabarti-PP and Ghose-S. Complexity of fragmentable object bin packing and an application. Computers and Mathematics with Applications. vol.35, no.11, pp. 91-7, 1998.

[MCNS] Multimedia Cable Network System Ltd., "Data-Over-Cable Service Interface Specifications - Radio Frequency Interface Specification", July 2000. http://www.cablelabs.com.

[MT81] S. Martello and P. Toth. Heuristics algorithms for the Multiple Knapsack Problem. Computing, 27, pp. 93-112, 1981.

[MT90] S. Martello and P. Toth. Knapsack Problems: Algorithms and computer Implementations. Jhon Wily and Sons, Ltd., New York, 1990.

[RT87] T. Rhee. and M. Talagrand. Martingale inequalities and NP-complete problems. Mathematical Operations Research , vol. 12, pp. 177-181, 1987.

[Ull76] J. D. Ullman. Complexity of sequencing Problems. Computer and Job-Shop Scheduling Theory. Wiley and Sons. 1976.