*Discrete Mathematics and Theoretical Computer Science* **4**, 2001, 139–156

# *Analysis of Transmission Scheduling with Packet Fragmentation*

Nir Namman and Raphael Rom

*Department of Electrical Engineering*
*Technion - Israel Institute of Technology*
*Haifa 32000, Israel*
{*mnir@techunix, rom@ee*}*.technion.ac.il*

*received Dec 18, 2000*, *revised Aug 1, 2001*, *accepted Aug 10, 2001.*

We investigate a scheduling problem in which packets, or datagrams, may be fragmented. While there are a few applications to scheduling with datagram fragmentation, our model of the problem is derived from a scheduling problem present in data over CATV networks. In the scheduling problem, datagrams of variable lengths must be assigned (packed) into fixed length time slots. One of the capabilities of the system is the ability to break a datagram into several fragments. When a datagram is fragmented, extra bits are added to the original datagram to enable the reassembly of all the fragments.

We convert the scheduling problem into the problem of bin packing with item fragmentation, which we define in the following way: we are asked to pack a list of items into a minimum number of unit capacity bins, each item may be fragmented in which case overhead units are added to the size of every fragment. The cost associated with fragmentation renders the problem NP-hard; therefore, an approximation algorithm is needed. We define a version of the well-known Next-Fit algorithm, capable of fragmenting items, and investigate its performance. We present both worst case and average case results and compare them to the case where fragmentation is not allowed.

**Keywords:** scheduling, bin packing, fragmentation, CATV, algorithm, average case analysis

## 1 Introduction

We address the class of transmission scheduling problems in which packets of variable lengths must be assigned into fixed length time slots (see e.g., [3]). Our model of the problem is derived from a scheduling problem present in data over CATV (Community Antenna Television) networks. In particular we refer to Data-Over-Cable Service Interface Specification (DOCSIS), standard of the Multimedia Cable Network System (MCNS) standard committee (see [18] for a detailed description). 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 mini-slots. 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 datagram transmission in the best possible way.

The headend must consider two kinds of datagram allocations:

1. *Fixed Location* - allocations for connections with timing demands, such as a CBR (constant bit rate) connection. These connections must be scheduled so as to ensure delivering the guaranteed service. Fixed location datagrams are therefore scheduled in fixed, periodically located mini-slots.

2. *Free Location* - allocations for connections without timing demands, such as a best effort connection. Free location datagrams can use any of the mini-slots.

The headend therefore performs the allocation in two stages. In the first stage it schedules, or allocates, all fixed location datagrams. We assume that after the fixed allocations have been made, a gap of $U$ mini-slots is left between successive fixed allocations. In the second stage all free location datagrams are scheduled. The free allocations must fit into the gaps left by the fixed allocations. One of the capabilities of the system is the ability to break a datagram into smaller pieces called *fragments*. When a datagram is fragmented, i.e., transmitted over non successive mini-slots, extra bits are added to the original datagram to enable the reassembly of all the fragments at the headend. In a typical CATV network one mini-slot is added to every fragment.

We model the scheduling problem as a bin packing problem. The relation to the bin packing problem should be clear. The items are the free location datagrams that should be scheduled, each of which may require a different number of mini-slots. The bins are defined by the gaps between every two successive fixed allocations in the MAP. The goal is to use the available mini-slots in the MAP in the best way.

Because of its applicability to a large number of applications and because of its theoretical interest bin packing has been widely researched and investigated (see, e.g., [7], [11] and [2] for a comprehensive survey). Since the problem, as many of its derivatives, is NP-hard [8] many approximation algorithms have been developed for it (see, e.g., [12], [14] and [1] for a survey). To analyze the scheduling problem we introduce a new variant of bin packing which we call ***bin packing with item fragmentation***. We convert the scheduling problem into the problem of bin packing with item fragmentation and show that the two are strongly related. The subject of item fragmentation in bin packing problems has received almost no attention so far. This paper concentrates on aspects that were heretofore never researched, such as developing an algorithm for the problem and investigating its performance.

The cost associated with fragmentation renders the bin packing problem nontrivial. In the scheduling problem, where items correspond to datagrams, the cost is 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 bin packing problem. Hence, the problem is interesting with the middle-range costs. It has been shown in [16] that for non zero cost, the problem of bin packing with item fragmentation is NP-hard.

We present an analysis of the Next-Fit (NF) algorithm, which is perhaps the simplest algorithm for bin packing. The algorithm keeps only one open bin and packs items according to their order, into the open bin. When an item does not fit in the open bin, the bin is closed and a new bin is opened. The NF algorithm is very simple, can be implemented to run in linear time, and requires only one open bin (bounded space).

It is therefore interesting to investigate the performance of such an algorithm before considering other, more complicated, algorithms.

We point out that the practical scheduling problem may actually be somewhat more complicated than the problem we present here since the bins may not be of uniform size. This situation is likely to happen for example when the fixed location allocations are a result of several CBR connections with different intervals between successive allocations. In such cases we may also face a situation where some items are bigger than some of the bins. While the problem of variable size bins is beyond the scope of this paper, we do mention some results we obtained while analyzing this case. The analysis of variable size bins, as well as the analysis of algorithms other than NF, appears in [17].

Our work contains several contributions. We introduce the variant of bin packing with item fragmentation and relate it to scheduling problems where datagrams may be fragmented. Our analysis of the scheduling algorithm contributes new results to the literature of bin packing. Finally, we developed a new technique for average case analysis which has some important advantages over existing techniques.

The remainder of the paper is organized as follows. In Section 2 we formally define the problem. Section 3 presents worst case analysis of the scheduling algorithm. Section 4 is devoted to average case analysis.

## 2 Problem Statement and Definitions

In this section we formally define the problem of bin packing with item fragmentation and show its relation to the scheduling problem. We define the problem similar to the classical bin packing problem. 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. To handle fragmentation, we use a discrete version of the problem and add a fragmentation cost function that adds overhead bits to each fragment. We proceed to formally define the problem.

**Bin Packing with Item Fragmentation (BP-IF)**: We are given a list of $n$ items $L = (a_1, a_2, ..., a_n)$, each with a size $s(a_i) \in \{1, 2, ..., U\}$. 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.

The analysis of bin packing algorithms is traditionally divided into worst case analysis and average case analysis. In worst case analysis we are usually interested in the ***asymptotic worst case performance ratio.*** For a given list of items, $L$ 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* $R_A^\infty$ is defined to be

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

A different approach for estimating the performance of an algorithm is an average case analysis. In this case we assume that item sizes 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 ***asymptotic expected performance ratio*** is defined as follows:

$$\overline{R}_A^\infty(H) \equiv \lim_{n\to\infty} E[R_A(L_n)] = \lim_{n\to\infty} E\left[\frac{A(L_n)}{OPT(L_n)}\right] \quad (2)$$

**Schedule Efficiency**: To evaluate the performance of a scheduling algorithm $A$, we compare the channel utilization achieved by $A$ to that of the best possible schedule. Let $s(L)$ denote the total sum of all item sizes in $L$, the channel utilization of algorithm $A$ is $C_A(L) = \frac{s(L)}{A(L)\cdot U}$. The worst case schedule efficiency, which we denote by $\eta_A$, is exactly the inverse of the worst case performance ratio of $A$

$$\begin{aligned} \eta_A(L) &= \frac{s(L)/(A(L)\cdot U)}{s(L)/(OPT(L)\cdot U)} = (R_A(L))^{-1} \\ \eta_A^\infty &= (R_A^\infty)^{-1} \end{aligned} \tag{3}$$

The expected schedule efficiency is the inverse of the expected performance ratio of $A$: $\overline{\eta_A^\infty} = (\overline{R_A^\infty})^{-1}$.

In this paper we analyze the performance of the Next-Fit algorithm. We denote by NF$_f$ the version of NF capable of fragmenting items and define the NF$_f$ algorithm similarly to NF.

**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.

In the reminder of this paper we concentrate on the bin packing problem. We calculate both the worst case and the expected asymptotic performance ratio of the NF$_f$ algorithm and compare it to known results about the NF algorithm. The schedule efficiency of the algorithms is easily derived from the performance ratio using (3).

# 3 Worst Case Analysis

The NF algorithm is the least efficient among all standard bin packing algorithms. The asymptotic worst case performance ratio of the algorithm is the worst possible

$$R_{NF}^\infty = \frac{2U}{U+1}, \quad \text{for every } U \geq 2. \tag{4}$$

We now analyze the performance of the NF$_f$ algorithm for the problem of bin packing with item fragmentation. As we shell see, the ability to fragment items may considerably improve the performance of the NF algorithm.

**Theorem 1** *The asymptotic worst case performance ratio of algorithm NF$_f$ is $R_{NF_f}^\infty = \frac{U}{U-2}$, for every $U \geq 6$.*

*Proof*: We first prove the upper bound and then provide an example to establish the lower bound.

**Claim 3.1** $R_{NF_f}^\infty \leq \frac{U}{U-2}$ *, for every $U > 2$.*

*Proof*: The NF$_f$ algorithm may pack at most two fragments in each bin (one fragment when the bin is opened and another when the bin is closed). There are therefore at most 2 overhead units in a bin which means that the number of bins used when the algorithm is applied to list $L$ is at most $\lceil s(L)/(U-2)\rceil$. The optimal packing of $L$ requires at least $\lceil s(L)/U\rceil$ bins and the claim follows. □

**Claim 3.2** $R^{\infty}_{NF_f} \geq \frac{U}{U-2}$ *, for every* $U \geq 6$*.*

*Proof*: 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 $\frac{U}{2} - 2$ items are of size 1. The rest of the list repeats this pattern $kU$ times. 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 $OPT(L) = (U-2)k$. 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 the number of bins used by NF$_f$ is therefore

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

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 $\frac{U-1}{2} - 1$ items are of size 1. The rest of the list repeats this pattern $kU$ times. It is easy to verify that this list produces the same ratio. It follows from the examples that: $R^{\infty}_{NF_f} = \frac{NF_f(L)}{OPT(L)} \geq \frac{U}{U-2}$, $U \geq 6$. □

The combination of the above claims proves the theorem. □

There is a considerable difference between the worst case performance ratios of NF and NF$_f$. Since both algorithms work in a similar way, we conclude that the improved performance ratio of NF$_f$ is due to its ability to fragment items. Note that the performance ratio of any algorithm that adopts the simple rule of filling a bin whenever an item is fragmented, cannot be worse than that of NF$_f$.

Finally, we point out a few more properties of the worst case performance ratio of the NF$_f$ algorithm. The results are taken from [17] and we present them here without a proof.

- When the bin size is very small Theorem 1 does not hold (because items of size 2 are not fragmented). For the values $3 \leq U \leq 5$ we show that the worst case asymptotic performance ratio of NF$_f$ is $R^{\infty}_{NF_f} = \frac{3}{2}$.
- For a more general case, where $r$ (instead of 1) units of overhead are added to the size of every fragment, it can be shown by similar arguments that $R^{\infty}_{NF_f} = \frac{U}{U-2r}$, for every $U > 4r + 2$.
- As we mentioned, in the scheduling problem the bins may be of different sizes. We denote by $\overline{U}$ the average bin size and show that for the case of variable size bins the worst case performance ratio of NF$_f$ is $R^{\infty}_{NF_f} = \frac{\overline{U}}{\overline{U}-2}$, for every $\overline{U} \geq 5$.
- The worst case results do not change if we allow the items to be of any size, that is, we assume that the size of some items may be larger than the size of some (or all) of the bins.

# 4 Average Case Analysis

The worst case analysis we presented, provides an upper bound on the performance ratio of the NF$_f$ algorithm. However, from a practical point of view it may be too pessimistic, since the worst case may rarely occur. To learn about the typical behavior of the algorithm we present an average case analysis.

Since the results of an average case analysis depend on the item-size distribution, it is desirable to be able to calculate results for any given distribution. We therefore consider some general item-size distribution assuming only that the items are independent, identically distributed (i.i.d). Let us define the problem in a formal way.

**Average case analysis of BP-IF**: We are give a list of $n$ items $L = (a_1, a_2, ..., a_n)$. The size of each item is 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 [5] who showed that the asymptotic expected performance ratio for continuous uniform distribution is $\overline{R}^{\infty}_{NF} = \frac{4}{3}$. For discrete uniform distribution, in the range $s(a_t) \in \{1, 2, ..., U\}$, it has been shown in [3] that the NF algorithm has the following asymptotic expected performance ratio:

$$\overline{R}^{\infty}_{NF} = \frac{2(2U+1)}{3(U+1)}. \tag{5}$$

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 [5], [10] and [13]). 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.

## 4.1 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 $1 \leq N_t \leq U$, $1 \leq t \leq n$. 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

$$I^n_{av}(A) \equiv E\left[\frac{1}{n}\sum_{t=1}^{n}(s(a_t) + oh_t)\right] \tag{6}$$

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

$$I_{av}(A) \equiv \lim_{n\to\infty} I^n_{av}(A) \tag{7}$$

Since $I_A(H)$ accounts for both the used and wasted space in the bins, we can state the following relation between $I_A(H)$ and $A(L)$:

$$I_A(H) = \lim_{n\to\infty} E\left[\frac{U \cdot A(L_n)}{n}\right] \tag{8}$$

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 $n$ [19], 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 [6]. Therefore the asymptotic expected performance ratio is given by

$$\overline{R}_A^\infty = \lim_{n\to\infty} E\left[\frac{A(L_n)}{OPT(L_n)}\right] = \lim_{n\to\infty} \frac{E\left[A(L_n)\right]}{E\left[OPT(L_n)\right]} = \lim_{n\to\infty} \frac{\frac{U}{n}\,E\left[A(L_n)\right]}{\frac{U}{n}\,E\left[OPT(L_n)\right]} = \frac{I_{av}(A)}{I_{av}(OPT)} \tag{9}$$

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, finding $I_{av}(OPT)$ for certain item size distributions may require exponential time in $n$. Fortunately, we do know that for several important distributions, including the uniform distribution, the overhead of the optimal packing can be neglected [2]. For such distributions we have

$$I_{av}(OPT) = \sum_{i=1}^{U} i \cdot h_i = \overline{h} \tag{10}$$

In cases where $I_{av}(OPT)$ is not known we can still find the channel utilization of the algorithm by replacing $I_{av}(OPT)$ with $\overline{h}$, i.e., the average item size of the given distribution.

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 $\Pi = (\Pi_1, \ldots \Pi_U)$ 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 $n_{j,i}$:

$$\lim_{n\to\infty} \frac{n_{j,i}}{n} = \lim_{n\to\infty} \frac{n_j}{n} \cdot h_i = \Pi_j \cdot h_i\,,\ a.s. \tag{11}$$

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:

$$\begin{aligned} I_{av}(NF) &= \lim_{n\to\infty} I^n_{av}(NF) = \lim_{n\to\infty} E\left[\frac{1}{n}\sum_{j=1}^{U}\sum_{i=1}^{U} n_{j,i} \cdot (i + oh_i(j))\right] \\ &= E\left[\sum_{j=1}^{U}\sum_{i=1}^{U} \lim_{n\to\infty} \frac{n_{j,i}}{n} \cdot (i + oh_i(j))\right] \end{aligned} \tag{12}$$

Substituting (11) we get

$$I_{av}(NF) = \sum_{j=1}^{U}\sum_{i=1}^{U} \Pi_j \cdot h_i \cdot (i + oh_i(j)) \tag{13}$$

To simplify (13) we use the following definitions:

$$\begin{array}{rll} \overline{h} \equiv & \sum_{i=1}^{U} i \cdot h_i & \text{average size of items (without overhead)} \\ OH(j) \equiv & \sum_{i=1}^{U} h_i \cdot oh_i(j) & \text{average overhead in state } j \\ \overline{OH} \equiv & \sum_{j=1}^{U} \Pi_j \cdot OH(j) & \text{average overhead size} \end{array} \tag{14}$$

Equation (13) now becomes:

$$I_{av}(NF) = \sum_{j=1}^{U} \Pi_j \cdot \sum_{i=1}^{U} i \cdot h_i + \sum_{j=1}^{U} \Pi_j \cdot \sum_{i=1}^{U} h_i \cdot oh_i(j) = \overline{h} + \sum_{j=1}^{U} \Pi_j \cdot OH(j) = \overline{h} + \overline{OH} \tag{15}$$

The expression in (15) 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, $\Pi$. We find $\Pi$ 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.

### 4.1.1 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 [4]:

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

From the above result we conclude that $\lim_{n\to\infty} E\left[\frac{U \cdot OPT(L_n)}{s(L_n)}\right] = 1$. We therefore neglect the overhead of the optimal packing in calculating the asymptotic expected performance ratio. Calculating $I_{av}(OPT)$ is now trivial

$$I_{av}(OPT) = \sum_{i=1}^{U} i \cdot h_i = \frac{1}{U}\sum_{i=1}^{U} i = \frac{U+1}{2} \tag{16}$$

We use (15) 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 $\Pi = (\Pi_1, \ldots, \Pi_U)$ 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 $j \leq \lfloor \frac{U}{2} \rfloor$ we have

$$P_{j,k} = \frac{1}{U} \cdot \left\{ \begin{array}{cc} 0 & 1 \leq k \leq j \\ 1 & j < k \leq U - j \\ 2 & U - j < k \leq U \end{array} \right| \quad 1 \leq j \leq \left\lfloor \frac{U}{2} \right\rfloor \tag{17}$$

The last line is $P_{U,k} = \frac{1}{U}$, $1 \leq k \leq U$.

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

$$\Pi_j = \frac{2j}{U(U+1)} \tag{18}$$

Next we compute the overhead component $OH(j)$. When the $NF$ algorithm is in state $j$ any item bigger than $U - j$ creates an overhead of $U - j$ units; all other items are packed without overhead. Hence, the average overhead in state $j$ is

$$OH(j) = \sum_{i=1}^{U} h_i \cdot oh_i(j) = \sum_{i=U-j+1}^{U} \frac{U-j}{U} = \frac{j(U-j)}{U} \tag{19}$$

We now use (18) and (19) to find the average combined size of the items

$$\begin{aligned} I_{av}(NF) &= \overline{h} + \sum_{j=1}^{U} \Pi_j \cdot OH(j) = \frac{U+1}{2} + \sum_{j=1}^{U} \frac{2j}{U(U+1)} \cdot \frac{j(U-j)}{U} \\ &= \frac{U+1}{2} + \sum_{j=1}^{U} \frac{2j^2(U-j)}{U^2(U+1)} = \frac{2U+1}{3} \end{aligned} \tag{20}$$

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

$$\overline{R}_{NF}^{\infty} = \frac{I_{av}(NF)}{I_{av}(OPT)} = \frac{(2U+1)/3}{(U+1)/2} = \frac{2(2U+1)}{3(U+1)} \tag{21}$$

The result we obtain for the asymptotic expected performance ratio is in accordance with results reported in [3]. The asymptotic worst case performance ratio is given in (4). We compare the two, for several values of $U$, in Table 1. We observe that in both cases the asymptotic performance ratio increases with the bin size. There is, however, a dramatic difference between the worst case performance ratio and the expected performance ratio. The average case results are therefore not as bad as the worst case analysis indicates.

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

| **Bin Size** $U$ | **Worst case ratio** $R_{NF}^{\infty}$ | **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... |
| $\infty$ | 2 | 1.3333... |

## 4.2 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 subsection 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 $P$

$$P_{j,k} = \frac{1}{U} \cdot \begin{cases} 0 & k \le 2, \quad k \le j \le U-k \\ 2 & j \in \{k-2,\, k-1\} \quad 3 \le k \\ 1 & else \end{cases} \tag{22}$$

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

$$\Pi_1 = \frac{1}{U}\Pi_U \tag{23}$$

$$\Pi_2 = \frac{1}{U}(\Pi_1 + \Pi_{U-1} + \Pi_U) \tag{24}$$

$$\Pi_j = \frac{1}{U}(1 + \Pi_{j-1} + \Pi_{j-2}), \quad 3 \le j \le U \tag{25}$$

Note that the solution to (25) (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 (23)-(25) is not simple. We therefore defer the calculation of a closed form solution to subsection 4.2.1 and proceed to calculate $OH(j)$, the average overhead in state $N = j$. 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

$$OH(j) = \frac{1}{U} \cdot \begin{cases} 2j & 1 \le j \le U-2 \\ U-1 & j = U-1 \\ 0 & j = U \end{cases} \tag{26}$$

We can now express the average combined size of the items

$$I_{av}(NF_f) = \frac{U+1}{2} + \sum_{j=1}^{U} \Pi_j \cdot OH(j) = \frac{U+1}{2} + \frac{U-1}{U}\Pi_{U-1} + \sum_{j=1}^{U-2} \Pi_j \cdot \frac{2j}{U} \tag{27}$$

Similar to (16), the overhead of the optimal packing is negligible $I_{av}(OPT) = \frac{U+1}{2}$.
The asymptotic expected performance ratio is therefore

$$\overline{R}^{\infty}_{NF_f} = 1 + \frac{2}{U+1}\left(\frac{U-1}{U}\Pi_{U-1} + \sum_{j=1}^{U-2} \Pi_j \cdot \frac{2j}{U}\right) \tag{28}$$

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 asymptotic expected and worst case performance ratio of NF$_f$ for several values of $U$.

**Tab. 2:** Expected and worst case 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... |
| $\infty$ | 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 1 and 2 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

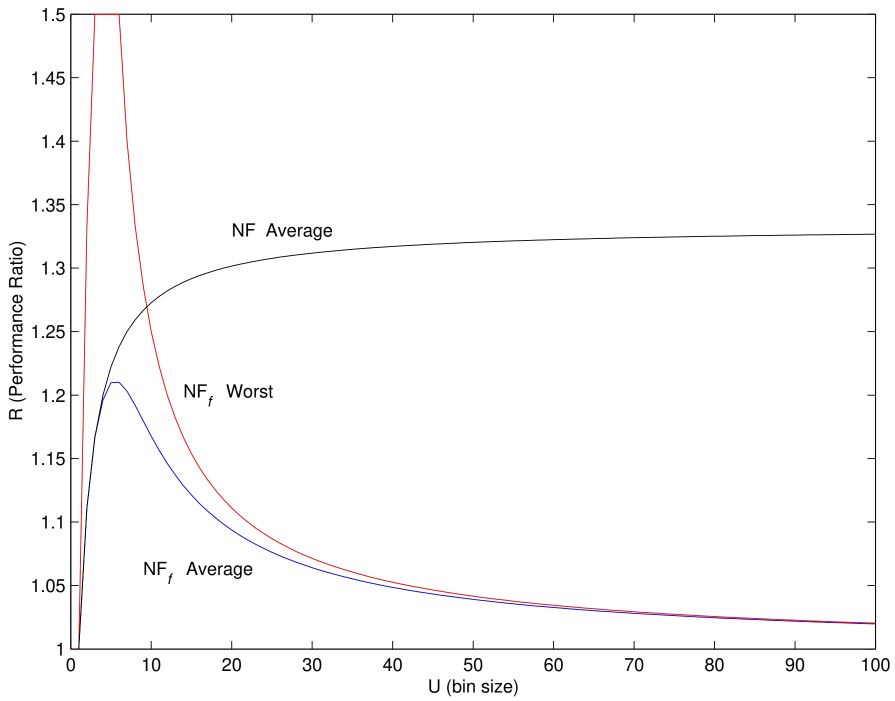


**Fig. 1:** Expected and worst case performance ratio of the NF$_f$ and NF algorithms.

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 [2]: $\overline{R}_{FF}^{\infty} = \overline{R}_{BF}^{\infty} = 1$.

Finally we note that it is not difficult to repeat the analysis we presented, for the general case, where $r$ units of overhead are added to the size of every fragment. We show in [17] that in this case the equivalent of (28) is

$$\overline{R}_{NF_f}^{\infty} = 1 + \frac{2}{U+1}\left(\sum_{j=1}^{U-2r} \Pi_j \cdot \frac{2jr}{U} + \sum_{j=U-2r+1}^{U} \Pi_j \cdot \frac{j(U-j)}{U}\right) \tag{29}$$

### 4.2.1 Calculating the Expected Performance Ratio

In this subsection 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 (28), to express $\overline{R}_{NF_f}^{\infty}$ in the following way:

$$\overline{R}_{NF_f}^{\infty} = 1 + \frac{2}{U+1}\left(\frac{2}{U}\overline{N} - \frac{U-1}{U}\Pi_{U-1} - 2\Pi_U\right) \tag{30}$$

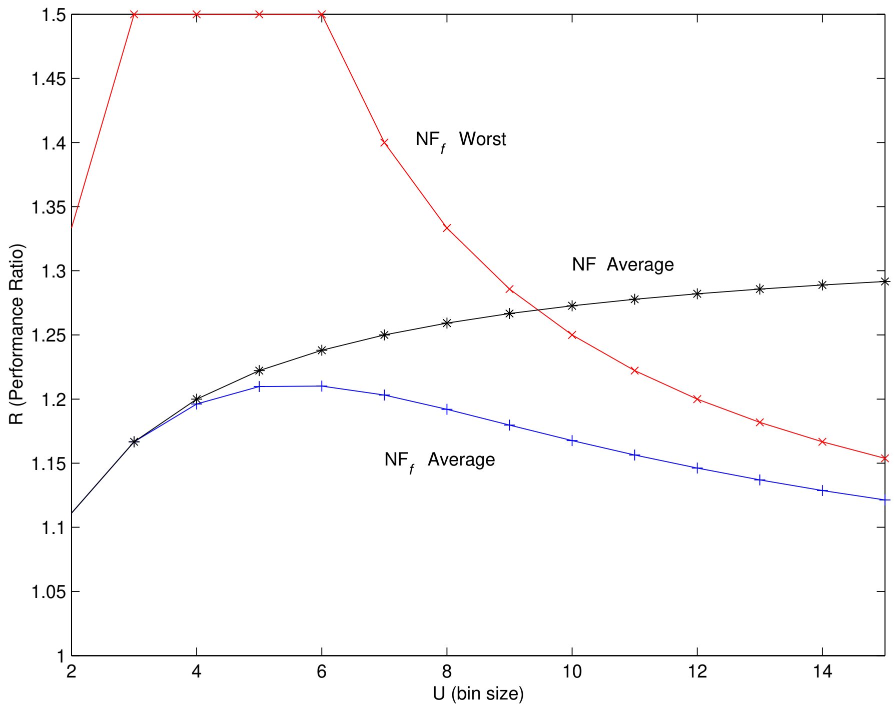


**Fig. 2:** Expected and worst case performance ratio of the NF$_f$ and NF algorithms for values of $U \le 15$.

To calculate the value of $\overline{N}$ we use a generating function: $G(z) = \sum_{j=1}^{U} \Pi_j z^j$. We use the equilibrium equations (23) - (25) to get

$$\begin{aligned} G(z) &= \Pi_1 z + \Pi_2 z^2 + \sum_{j=3}^{U} \frac{1}{U} (1 + \Pi_{j-1} + \Pi_{j-2}) z^j \qquad (31) \\ &= \frac{1}{U} \Pi_U z + \frac{1}{U} \left( \Pi_{U-1} + \frac{U+1}{U} \Pi_U \right) z^2 + \sum_{j=3}^{U} \frac{1}{U} (1 + \Pi_{j-1} + \Pi_{j-2}) z^j \end{aligned}$$

Arranging the above expression we get

$$G(z) = \frac{\Pi_U z^{U+2} + (\Pi_U + \Pi_{U-1}) z^{U+1} - (\Pi_U + \Pi_{U-1}) z^2 - \Pi_U z - \sum_{j=3}^{U} z^j}{z^2 + z - U} \qquad (32)$$

To find $\overline{N}$ we calculate the derivative of the generating function at $z = 1$

$$\overline{N} = \frac{dG(z)}{dz}\Big|_{z=1} = \frac{U(U+1) - 4U\Pi_U - 2(U-1)\Pi_{U-1}}{2(U-2)} \qquad (33)$$

Substituting (33) in (30) we get an expression for the asymptotic expected performance ratio

$$\overline{R}^{\infty}_{NF_f} = 1 + \frac{2}{U+1}\left(\frac{2}{U}\overline{N} - \frac{U-1}{U}\Pi_{U-1} - 2\Pi_U\right) = \frac{U}{U-2} - \frac{4U^2\Pi_U + 2U(U-1)\Pi_{U-1}}{(U+1)U(U-2)} \tag{34}$$

We now have an expression which is a function of $\Pi_{U-1}$ and $\Pi_U$

$$\overline{R}^{\infty}_{NF_f} = \frac{U}{U-2} - \frac{4U\Pi_U + 2(U-1)\Pi_{U-1}}{(U+1)(U-2)} \tag{35}$$

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 $\Pi_U, \Pi_{U-1}$. We do so by exploring the roots of the generating function given in (32). 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 $\Pi_U$ and $\Pi_{U-1}$ can be found. Denote by $z_1$ and $z_2$ the roots of the denominator

$$z_1 = -\frac{1}{2} + \sqrt{U + \frac{1}{4}}, \qquad z_2 = -\frac{1}{2} - \sqrt{U + \frac{1}{4}}$$

Substituting $z_1$ in the numerator we get

$$\Pi_U z_1^{U+2} + (\Pi_U + \Pi_{U-1})(z_1^{U+1} - z_1^2) - \Pi_U z_1 - \sum_{j=3}^{U} z_1^j = 0 \tag{36}$$

We get the same equation if we substitute $z_2$ in the numerator. Using the two equations it is now a straightforward algebraic exercise to find $\Pi_U$ and $\Pi_{U-1}$.

$$\Pi_U = \frac{U}{(2-U)} \frac{\sqrt{4U+1}\left((-U)^{U-1}+1\right) + z_2^{U-2}\left(2U + z_2\left(1-U\right)\right) + z_1^{U-2}\left(z_1\left(U-1\right) - 2U\right)}{\sqrt{4U+1}\left((-U)^{U}-1\right) + \left(z_1^U - z_2^U\right)\left(U+1\right)} \tag{37}$$

$$\Pi_{U-1} = \frac{1}{(2-U)} \frac{\left(z_2-1\right)\left(z_1^{U+1} - z_1^3\right)\left(z_2^U - 1\right) - \left(z_1-1\right)\left(z_2^{U+1} - z_2^3\right)\left(z_1^U - 1\right)}{\sqrt{4U+1}\left((-U)^{U}-1\right) + \left(z_1^U - z_2^U\right)\left(U+1\right)} \tag{38}$$

To find the expected performance ratio we substitute (37) and (38) in (35).

The expression we obtained for the expected performance ratio enables exact calculation for any value of $U$ but does not provide too much insight. To get a better understanding we note that (25) 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

$$\begin{aligned} \overline{R}^{\infty}_{NF_f} &= \frac{U}{U-2} - \frac{4U\Pi_U + 2(U-1)\Pi_{U-1}}{(U+1)(U-2)} \cong \frac{U}{U-2} - \frac{4U\frac{1}{U-2} + 2(U-1)\frac{1}{U-2}}{(U+1)(U-2)} \\ &= \frac{U}{U-2} - \frac{6U-2}{(U+1)(U-2)^2} \end{aligned} \tag{39}$$

Comparing the exact value of the asymptotic expected 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.

### 4.3 General Item Size Distribution

In this subsection we demonstrate how the analysis can be applied to any item size distribution. We assume the items are i.i.d and the probability to draw an item of size $i$ is $h_i$. As we mentioned earlier, since finding $I_{av}(OPT)$ may be difficult, we calculate the bin utilization which requires finding $I_{av}(NF_f)$ only. We use (15) to calculate the average combined size of the items. The construction of the transition matrix and the calculation of the equilibrium probabilities is similar to the one presented in the previous section. The calculation of the overhead component $oh_i(j)$ (overhead added to an item of size $i$ which is packed in state $j$) is simple

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

**Example**: We present an example using typical parameters of a data over CATV network. We assume a mini-slot is 25 microsecond and 16 bytes are transmitted in each mini-slot. In data over CATV networks a cable modem transfers IP datagrams using Ethernet packets. We therefore assume the datagram have typical Ethernet distribution with datagram sizes of 4, 8, 16, 64, and 94 mini-slots, and with the following probabilities: $h_4 = 0.5$, $h_8 = 0.1$, $h_{16} = 0.05$, $h_{64} = 0.15$, and $h_{94} = 0.2$. The average item size of the given distribution is $\overline{h} = 32$. We assume the bin size is $U = 100$ mini-slots, which corresponds to 2.5 millisecond.

We are interested in the channel utilization of algorithms NF and NF$_f$. Using our average case analysis we find that $I_{av}(NF) = 40.5$ and $I_{av}(NF_f) = 32.6$. The channel utilization of NF is therefore $\overline{C}^{\infty}_{NF} = 0.79$. The channel utilization of NF$_f$ is considerably better $\overline{C}^{\infty}_{NF_f} = 0.981$. The corresponding worst case results for the example are $C^{\infty}_{NF} = 0.505$ and $C^{\infty}_{NF_f} = 0.98$.

## 5 Conclusions

We studied a scheduling problem in which datagrams my be fragmented. Such scheduling problem is present in data over CATV networks as well as in several other applications (e.g., [3], [16]). The analysis we presented can also serve as a tool for evaluating the benefits of fragmentation in other systems. For example, several reservation-based satellite systems present a similar scheduling problem to that of data over CATV (see e.g., [9], [15]) but fragmentation is not currently implemented in such systems.

To analyze the scheduling problem we introduced a new variant of bin packing that allows item fragmentation. We converted the scheduling problem into the problem of bin packing with item fragmentation and showed that the two problems are strongly related. We defined the NF$_f$ algorithm and performed both worst case and average case analysis to evaluate the schedule efficiency of the algorithm. We developed a new technique, based on calculating the overhead produced during the packing, to derive our average case results. This technique may prove useful in analyzing other problems.

We found that fragmentation can considerably improve the schedule efficiency of an algorithm. An important characteristic is that the schedule efficiency is increasing with the bin size (the gap between fixed allocations). This means that for large bin sizes, the efficiency of NF$_f$ is not far from the optimum. While NF$_f$ is not very efficient, it may still be chosen as a practical scheduling algorithm since it has some important advantage over other algorithms; NF$_f$ is very simple and is suitable for online and bounded-space scheduling. Moreover, it keeps a first-in-first-out order of transmissions.

There are several possible extensions to the work presented in this paper; some of them are addressed in [17]. From a practical point of view, other bin packing algorithms, should also be considered. We expect algorithms such as First-Fit (FF) and First-Fit Decreasing (FFD) [11], to perform better than NF. The improved performance has to be weighed against the advantages of the $NF_f$ algorithm. Another interesting extension is to consider the case where the fixed allocations in the MAP create variable size gaps. This time the scheduling problem is modeled as a variant of variable size bin packing.

## References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

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

[8] 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.

[9] A. Guntsch. Analysis of the ATDMA/PRMA++ protocol in a mobile satellite system environment. In proceedings of the IEEE 46th Vehicular Technology Conference 96', New York, USA, pp. 1225-9, 1996.

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

[11] 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, vol. 3, pp. 299-325, 1974.

[12] D. S. Johnson. Fast algorithms for bin packing. Journal of Computer and System Science, vol. 8, pp. 272-314, 1974.

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

[14] 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.

[15] H. Koraitim and S. Tohme. Performance analysis of multiple access protocols for multimedia satellite networks. IEEE Journal on Selected Areas in Communications. vol.18, no.9, pp. 1751-63, 2000.

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

[17] Nir Menakerman and Raphael Rom. Bin packing problems with item fragmentation. Technical report, Technion EE publication CITT #342, April 2001.

[18] Multimedia Cable Network System Ltd., "Data-Over-Cable Service Interface Specifications - Radio Frequency Interface Specification (status: interim)", April 2000.

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