UNLV Theses, Dissertations, Professional Papers, and Capstones

8-1-2012

A Survey of Classical and Recent Results in Bin Packing Problem

Yoga Jaideep Darapuneni
University of Nevada, Las Vegas, darapune@unlv.nevada.edu

Follow this and additional works at: https://digitalscholarship.unlv.edu/thesesdissertations

Part of the Computer Sciences Commons, and the Discrete Mathematics and Combinatorics Commons

Repository Citation
Darapuneni, Yoga Jaideep, "A Survey of Classical and Recent Results in Bin Packing Problem" (2012). *UNLV Theses, Dissertations, Professional Papers, and Capstones*. 1663.
https://digitalscholarship.unlv.edu/thesesdissertations/1663

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Theses, Dissertations, Professional Papers, and Capstones by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

A SURVEY OF CLASSICAL AND RECENT RESULTS IN BIN PACKING PROBLEM

by

Yoga Jaideep Darapuneni

A Thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
August 2012

THE GRADUATE COLLEGE

We recommend the thesis prepared under our supervision by

Yoga Jaideep Darapuneni

entitled

A Survey of Classical and Recent Results in Bin Packing Problem

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Wolfgang Bein, Committee Chair

Ajoy K. Datta, Committee Member

Ju-Yeon Jo, Committee Member

Zhiyong Wang, Graduate College Representative

Ronald Smith, Ph. D., Vice President for Research and Graduate Studies
and Dean of the Graduate College

May 2012

ABSTRACT

A SURVEY OF CLASSICAL AND RECENT RESULTS IN BIN PACKING PROBLEM

By

Yoga Jaideep Darapuneni

Dr. Wolfgang Bein, Examination Committee Chair

Professor, Department of Computer Science

University of Nevada, Las Vegas

In the classical bin packing problem one receives a sequence of *n* items 1, 2,…, *n* with sizes *s1*, *s2*, . . . ,*sn* where each item has a fixed *size* in (0, 1]. One needs to find a partition of the items into sets of size1, called *bins,* so that the number of sets in the partition is minimized and the sum of the sizes of the pieces assigned to any bin does not exceed its capacity. This combinatorial optimization problem which is N_P hard has many variants as well as online and offline versions of the problem. Though the problem is well studied and numerous results are known, there are many open problems. Recently bin packing has gained renewed attention in as a tool in the area of cloud computing. We give a survey of different variants of the problem like 2D bin packing, strip packing, bin packing with rejection and emphasis on recent results. The thesis contains a discussion of a newly claimed tight result for First Fit Decreasing by Dosa et.al. as well as various new versions of the problem by Epstein and others.

ACKNOWLEDGEMENTS

I would like to thank Dr. Wolfgang Bein for chairing my committee and advising this work. I am thankful for his continuous guidance and patiently clearing all my questions during the course of my thesis. I really enjoyed the thesis discussions we had and this helped in finishing my work in time. Without his generous help this thesis would not have had such a rich content. I am thankful to Dr. Ajoy K Datta for his support and guidance through my Masters program and help on my thesis. I would also like to specifically thank Dr. Ju-Yeon Jo and Dr. John Wang for serving on the committee. For this and for being generous with their time when I needed it, I am deeply indebted to them.

I would like to thank the faculty at the School of Computer Science, University of Nevada, Las Vegas for the formal education along with generous financial support. I would also like to extend my appreciation towards my family and friends for being there for me all the time and always encouraging me to strive for the best.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Class P

Set of decision problems or class of problems for which some algorithm can solve the problem in polynomial time. This means that the running time of the algorithm is bounded by a polynomial of input size. Let T(*n*) be the running size of the algorithm for input size *n*.

∃ a constant k such that the running time T(*n*) is $O(n^k)$

Example: sorting, minimum spanning tree.

1.2 Class Np

In formal terms class Np can be defined as the set of decision problems where the "yes" instances can be decided in polynomial time by a non-deterministic Turing machine.

Np is the class of decision problems, for which the "yes" answers have proofs verifiable in polynomial time by a deterministic Turing machine. The notation Np stands for "nondeterministic polynomial time", since originally Np was defined in terms of nondeterministic Turing machines (that is, machines that have more than one possible move from a given configuration).

Example: Subset sum problem, bin packing problem.

Class P and Class Np can be represented as P and Np respectively. So, P is the class of "easy to solve" problems, and NP is the class of "easy to check" problems. The class P in contained in class Np i.e. P $\subseteq$ Np. Does P = Np?

It is an open problem of major importance.

Is P=Np?

This is called the P vs Np problem. It is major unsolved problem in the field of computer science.

If P=Np, then it basically denotes the set of problems that can be verified in polynomial time (class Np) can also be solved in polynomial time (Class P).

If P $\neq$Np , it means that there are problems in Np(quickly verifiable) that are hard to solve than to verify. This gives rise to the concept of Class Np-Complete and Np-hard problems.

1.3 Np-Hard:

The set of problems is said to be in Np-hard if it contains the following property

- If there exists a polynomial time algorithm to solve one of these problems then there exists one for every problem in Np.

Note: Np-hard problems need not be in Np & need not be a decision problem.

1.4 Np-Complete:

A decision problem X is Np-complete iff

- X ϵ Np

- X is Np-hard (or) if every problem in Np can be reduced to X in polynomial time.

X can be shown to be in Np by showing that a candidate solution to X can be verified in polynomial time.

Np-complete problems are the hardest problems in Np. The importance of solving a Np-complete problem is that if we are able to find an algorithm to solve Np complete problem in polynomial time then we can solve every other Np problem in polynomial time.

No efficient algorithm for an NP-complete problem has ever been found; but nobody has been able to prove that such as algorithm does not exist. For many Np optimization problems, serious attempts are made to find the optimal solution in polynomial time but since it appears to be intractable we limit ourselves to approximate solutions using *r*-approximate algorithms. Now let us discuss about the *r*-approximate algorithm and its solution in the next section.

1.5 What is *r*-approximate solution?

For a given optimization problem *P*, there exists an algorithm *A* such that for any instance *I* it computes solution $m_{APPROX}(I)$, we say that *A* provides a *r*-approximation solution for problem *P* when for any instance *I*

$$\frac{1}{r} \leq \frac{m_{OPT}(I)}{m_{APPROX}(I)} \leq r$$

Where $m_{OPT}(I)$ is the optimal solution for instance *I* of problem *P*. Here *A* is said to be a *r*-approximate algorithm.

1.6 Class NPO:

Class NPO can be defined as a set of problems that allow polynomial time *r*-approximate algorithm. The existence of *r*-approximate algorithm for Np-hard problems helps in finding the approximate or the closest possible solution to the optimal.

1.7 Class APX:

APX belongs the class of all NPO problems where for some $r \geq 1$ there exist a *r*-approximate polynomial time algorithm.

So any problem which has *r*-approximate algorithm is said to be in class APX. Some of the problems which belong to the class APX are maximum satisfiability, maximum cut, minimum graph coloring restricted to planar graphs, minimum vertex cover, minimum bin packing and many more.

There are situations where for some NPO problems we cannot find *r*-approximate polynomial time algorithm unless P=Np, which can be interpreted as finding approximation algorithm is as hard as to determine optimal solution. This means that under the hypothesis P≠Np, class APX is strictly contained in class NPO i.e. APX $\subset$ NPO. Here in the given expression APX denotes class APX and NPO denotes class NPO.

Now as mentioned earlier in the above paragraph, we have situations where there are problems belonging to class NPO but does not belong to class APX. For example, minimum travelling salesperson problem is an optimization problem which does not have an *r*-approximate polynomial time algorithm. So it does not belong to class APX. Some other problems which do not belong to class APX are maximum clique and maximum independent set problem. Unfortunately for most of the problems in APX the performance ratio can only be approximated to a certain point, which means that a threshold exists *t* such that $r<t$ becomes computationally difficult.

1.8 Polynomial time Approximation Scheme (PTAS):

Let Q be an Np-hard optimization problem. An algorithm *A* is an approximation scheme for Q if for every $r > 0$, '*A*' returns a solution Q_{sol} such that

$Q_{sol} \leq (1 + r)\, Q_{opt}$ ------------ if Q is a minimization problem.

$Q_{sol} \geq (1 + r)\, Q_{opt}$ ------------ if Q is a maximization problem.

Q_{opt} means the optimal solution for the problem Q.

'*A*' will be called PTAS, if it runs in polynomial time of *n* and as we decrease *r*, the running time increases drastically. The dependency on *r* is exponential, so for example the running time can be of form $O(n^{\frac{1}{r}})$, $O(2^{\frac{1}{r}}n^{3})$ and many more.

Now for any NPO problem, let us suppose there exists a constant *k* and if its Np-hard to describe that for a given instance *I*, $m_{OPT}(I) \leq k$, then there is no PTAS for that problem and a polynomial time algorithm with $r < \frac{k+1}{k}$ exists only if P=Np.

1.9 Class PTAS:

Class PTAS can be defined as the set of problems that allow PTAS or has a PTAS. So any algorithm which contains PTAS is said to belong in class PTAS.

By definition class PTAS belongs to class PAX. So the problem which does not belong to class APX does not have PTAS too. Example: minimum travelling salesperson problem. So if P≠Np, then PTAS ⊂ APX where PTAS represent class PTAS and APX denotes class APX respectively.

The following picture depicts the relation between Class NPO, APX and PTAS. With the help of the figure below we can understand the relationship between these classes in a better way.

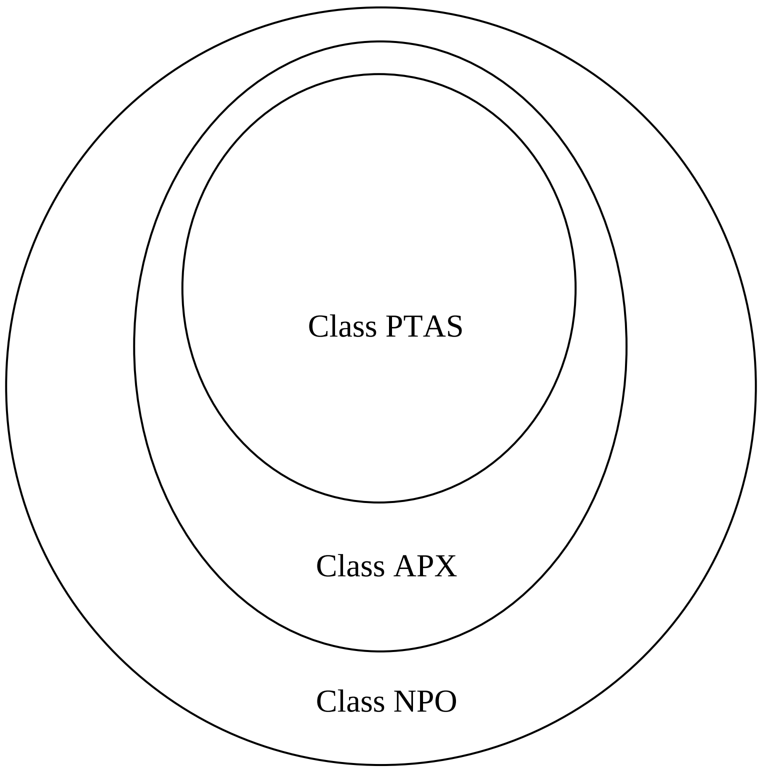


Fig 1: Relation between class PTAS, APX and NPO

Bin packing problem does not have PTAS. If P≠Np and r is the approximation ratio to bin packing, there is no r- approximate polynomial time algorithm for minimum bin packing problem for which $r \leq \frac{3}{2} - \varepsilon, \varepsilon > 0$.

1.10 Asymptotic Polynomial time Approximation Scheme (PTAS$^\infty$):

Let P be an NPO problem and let there exists a constant k. An algorithm A is said to be an asymptotic polynomial approximation scheme for any $r \geq 1$,if the algorithm A for the instance I returns a solution whose performance ratio is at most $r + \frac{k}{m_{OPT}(I)}$ where $m_{OPT}(I)$ denotes the optimal solution and algorithm A runs in polynomial time.

Asymptotic polynomial time approximation ratio (PTAS$^\infty$) is a weaker form of approximation when compared to PTAS. It is based on the idea that the performance ratio of the approximate solution (returned by the respective approximation algorithm) may improve as optimal solution becomes bigger.

Just like class PTAS we also have class PTAS$^\infty$ which is the set of all NPO problems that contain an asymptotic polynomial time approximation ratio(PTAS$^\infty$). So the relation among PTAS, APX and polynomial time approximation ratio(PTAS$^\infty$) can be given as

PTAS $\subseteq$ PTAS$^\infty$ $\subseteq$ APX.

CHAPTER 2

What is Bin-Packing?

Now coming to our problem, the bin packing problem is considered to be one of the combinatorial minimization problems. We receive a sequence of *n* items *L*= {1, 2,…, n} with sizes s_1, s_2, . . . , s_n and each item has a fixed *size* in (0, 1]. Now one needs to find a partition of the items into sets of size 1 (called *bins*) so that the number of sets in the partition is minimized and the sum of the sizes of the pieces assigned to any bin may not exceed its capacity. We say that an item that belongs to a given bin (set) is *packed* into this bin. A bin is *empty* if no item is packed into it, otherwise it is used. Since the goal is to minimize the number of bins used. Bin packing is N_P -hard, thus finding an exact solution for any given input can be done currently only in exponential time. Since it`s is an N_P-hard problem and the polynomial time optimization algorithm cannot be found unless $P=N_P$. A more reasonable approach would be finding an approximation algorithm *m* that runs in low-order polynomial time and for all instances *I*, m_{APPROX} *(I)* is close to $m_{OPT}(I)$. m_{APPROX} (*I*) represents the approximate solution for the given instance *I* . $m_{OPT}(I)$ represents the optimal solution for the instance *I.* m_{OPT}(I) can also be represented as $m^*(I)$ which means the same.

Our primary goal is to fit items into the bins such that the number of bins used is minimal. For this purpose, there are several algorithms developed which provides an approximate solution bounded by '*r*' (here '*r*' is the approximation ratio). These algorithms can be broadly classified into two categories

- Online bin packing Algorithms
- Offline Bin packing Algorithms

Online bin packing algorithms packs items in the bin as per the input sequence. These algorithms does not have knowledge of the next items in the input sequence whereas the offline algorithm has knowledge of the next item in the input sequence required for bin packing and can possibly arrange them in a particular order before packing the items in the bins.

Theorem: If P≠Np and for any $\varepsilon > 0$, we cannot find an *r*-approximate algorithm for bin packing problem whose approximation factor $r < \frac{3}{2} - \varepsilon$.

Proof: Consider a partition problem which is Np-Complete. This is a decision problem where for a given input of n numbers the problem is to decide if there is a way to partition or divide n number into two sets, such that each set is equal to $\frac{S_n}{2}$. Here S_n represents the sum of n numbers. This partition problem can be reduced to the bin packing problem where each number correspond to the items and these items should be packed into bins of size $\frac{S_n}{2}$. So for the given instance the answer to the decision problem is "yes" iff n items can be packed into two bins of size $\frac{S_n}{2}$. So if there is a $\frac{3}{2} - \varepsilon$ approximation algorithm then it will have to give an optimal packing and thereby solving the Np complete partition problem.

In the next section we discuss about the offline algorithms of bin packing problem. The First fit algorithm is one of the most basic algorithms for bin packing and can be used as both online and offline algorithm.

2.1 OFFLINE ALGORITHMS FOR BIN PACKING:

2.1.1 First Fit Algorithm:

Given an instance *x* of OFFLINE BIN PACKING, the algorithm First Fit returns a result **[1]** with value $m_{FF}(x)$ such that $m_{FF}(x) \leq 1.7\ m_{OPT}(x)+2$ where $m_{OPT}(x)$ denotes the optimal solution for an instance *x*. The numeric "2" represents the additive constant.

Algorithm

Consider bins b_j where $j \in (1, 2 \ldots, n)$

Consider an instance *x* containing items a_i where $i \in (1, 2 \ldots, n)$

```
Begin
    for $i$ := 1 to $n$ do
    for $j$ := 1 to $n$ do
    if item $a_i$ can fit in the bin $b_j$
        then
            Insert $a_i$ into the bin
            Break; //exit for $j$ loop
    //continue for $i$ loop
End
```

2.1.2 First Fit Decreasing Algorithm (FFD):

First Fit Decreasing is an enhanced algorithm with improved performance ratio and better approximation. In fact it is an offline algorithm where the items are first sorted in non-increasing order as per their size and then processes items as First Fit.

Given an instance x of BIN PACKING, the algorithm First Fit Decreasing**[7]** returns a result with value $m_{FFD}(x)$ such that $m_{FFD}(x) \leq \frac{11}{9} m_{OPT}(x) + 4$ where $m_{OPT}(x)$ denotes the optimal solution for an instance x.

Let us take an example to illustrate the distribution of items using first fit decreasing algorithm and optimal packing.

Consider an instance I of $5n$ items

The classifications of $5n$ items (input sequence of items) is as follows

- n items of size $½ + \varepsilon$,
- n items of size $¼ + 2\varepsilon$,
- n items of size $¼ + \varepsilon$,
- $2n$ items of size $¼ - 2\varepsilon$.

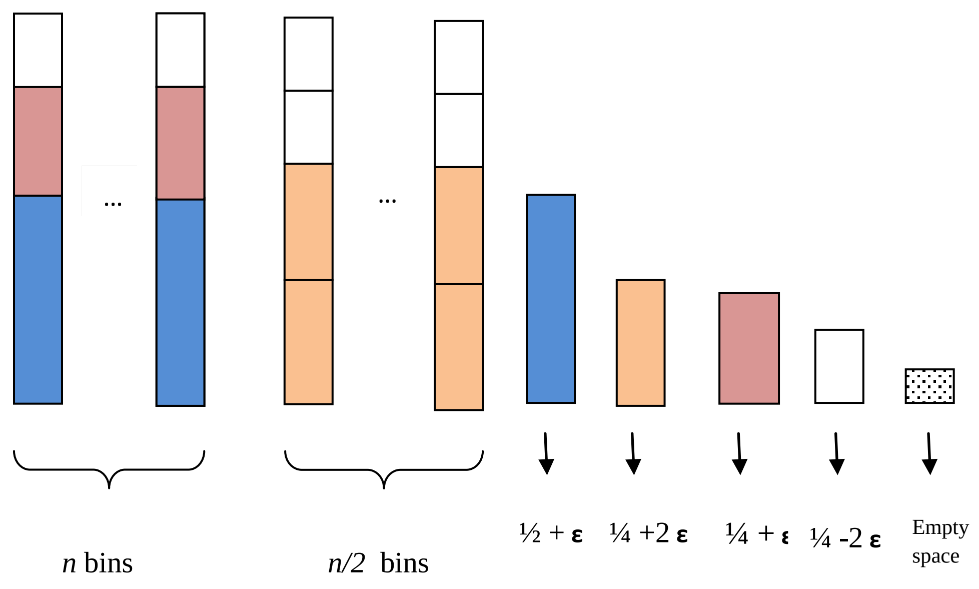
Using FFD, these $5n$ items can be filled in $11n/6$ bins ($m_{FFD}(I)= 11n/6$) .The distribution is done in this way, Initially as per the algorithm all the $5n$ items of different sizes are arranged in non-increasing order. After the arrangement, the items are filled into the bins as per the first fit algorithm. Now that we have non increasing sequence of input items, the first(largest) n items of size $½ + \varepsilon$ are filled in n different bins since we have unit size bins and two items of size $½ + \varepsilon$ would exceed the size of the bin . Now the next sequence of n items of size $¼ + 2\varepsilon$ is filled in the existing n bins as there is enough space for them to fit. After filling n items of size $½ + \varepsilon$, we deal with next sequence of items i.e. n items of size $¼ + \varepsilon$, now since these items cannot fit in the existing bins, new bins are opened and these n items are filled in $n/3$ bins with each bin containing 3 items of size $¼ + \varepsilon$. Now we are left with final sequence i.e. $2n$ items of size $¼ - 2\varepsilon$, Since these items cannot fit in any of the existing opened bins, new bins are opened and these 2*n* items and these items are filled in *n*/4 bins with each bin containing 4 items of size ¼ - 2**ε**. Hence, in this way the items are filled using the first fit decreasing algorithm. In short, the distribution can be described as follows

- *n* bins contain each item of size ½ + **ε** and **,** ¼ + 2 **ε** in one bin .
- *n*/3 bins contain three ¼ + **ε** size item in each bin.
- *n/2* bins contain four ¼ - 2 **ε** size items in each bin.

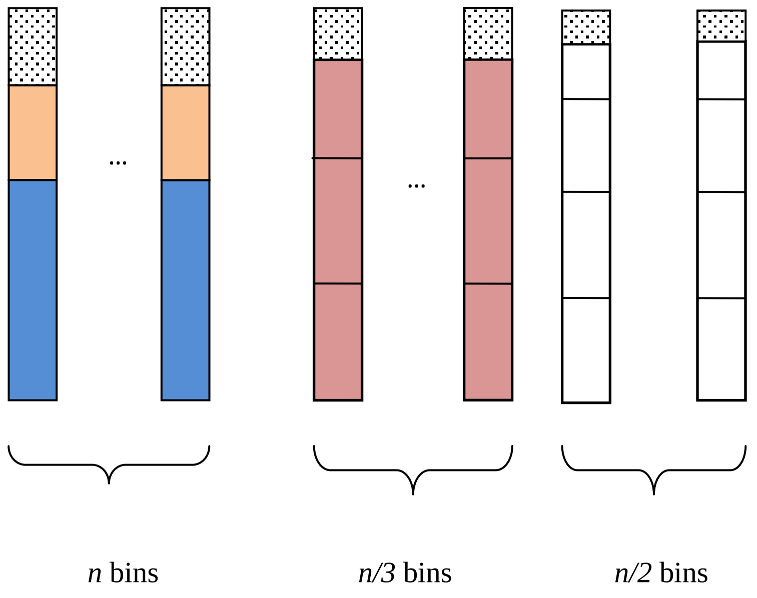
The optimal bin distribution can be detailed as follows

- *n* bins contain each item of size ½ + **ε,** ¼ - 2 **ε** and ¼ + **ε** in one bin.
- *n/2* bins contain two ¼ + 2**ε** size items and two ¼ - 2 **ε** size item in each bin.

In the below figure case (a) represents optimal bin distribution and case (b) shows the FFD bin distribution.



CASE (a)



CASE (b)

Fig 2: Optimal bin distribution and distribution using FFD algorithm

Thus the optimal solution for the instance of 5n items can be filled in 3n/2 bins (m_{OPT} (x) = 3n/2) and for FFD 5n items can be filled in 11n/6 bins (m_{FFD} (x) = 11n/6). So by the definition of performance ratio, we can calculate the performance ratio for a given instance L by max ($\frac{m_{APPROX}(L)}{m_{OPT}(L)}, \frac{m_{OPT}(L)}{m_{APPROX}(L)}$) where $m_{APPROX}(L)$ is the solution returned by the approximation algorithm for a given instance L and $m_{OPT}(L)$ represents the optimal solution. In the above case $m_{APPROX}(L)$ can be replaced by $m_{FFD}(x)$ since our approximation algorithm is First fit decreasing and the instance we are dealing with is x. Hence by substituting the actual values we get the performance ratio i.e. max ($\frac{11}{9}, \frac{9}{11}$) = $\frac{11}{9}$. Hence the above example not only illustrates the distribution of first fit decreasing algorithm but also shows that for given instance x the bound $\frac{11}{9}$ is tight and cannot get smaller than that.

In this doctoral thesis, D.S.Johnson [1] showed $m_{FFD}(x) \leq \frac{11}{9} m^*(x) + 4$, he proved that the performance ratio for FFD cannot get better than $\frac{11}{9}$. Though for an instance x, $m_{FFD}(x) \leq \frac{11}{9} m_{OPT}(x) + 4$ has tight bound and works considerably well for higher values of $m_{OPT}(x)$ ($m_{OPT}(x) > 10$) , work has been going on to find the closest asymptotic additive constant (like 4) which is required to find better approximations for smaller instances. In this process, after the D.S.Johnson, B.S.Baker[2] proved that additive constant can be reduced to 3. Later in 1991, Yue Minyi [3] proved that additive constant cannot be lesser than 1 i.e. $m_{FFD}(x) \leq \frac{11}{9} m^*(x) + 1$ but the proof is difficult to understand. Later in 1997, L. Rongheng, M. Yue [4] furthur tried to reduce the additive constant to $\frac{7}{9}$ but they did not prove the statement but gave a draft about it. They also conjectured that the tight additive constant can be $\frac{5}{9}$ (which proves to be an incorrect result). Finally in November 2011 Gyorgy Dósa, Rongheng Li, Xin Han and Zsolt Tuza [5] claimed that the lower bound for the additive constant is $\frac{6}{9}$, but the proof is 30 page long and considers a lot of test cases which makes it difficult to understand (not sure about the correctness of the proof).

So Gyorgy Dósa, Rongheng Li, Xin Han and Zsolt Tuza claim $\forall\ x,\ m_{FFD}(x) \leq \frac{11}{9} m_{OPT}(x) + \frac{6}{9}$ bound is tight. So the FFD guarantees that it is never more than 22 percent worse than optimal.

Now to illustrate the tightness (with regards to additive constant $\frac{6}{9}$) of the expression $m_{FFD}(x) \leq \frac{11}{9} m_{OPT}(x) + \frac{6}{9}$, let us consider an example where the instance is *x* and it can be described as follows

- 4 items of size ½ + **ε**
- 4 items of size ¼ + 2**ε**
- 4 items of size ¼ + **ε**
- 8 items of size ¼ - 2**ε**

So for an optimal solution, the items in instance *x* can be filled in 6 bins ($m_{OPT}(x) = 6$) and the distribution is done as follows

- 4 bins containing 1 item of size ½ + **ε,** ¼ + **ε** and ¼ - 2**ε** in each bin.
- 2 bins containing 2 items of size ¼ + 2**ε** and ¼ - 2**ε** in each bin.

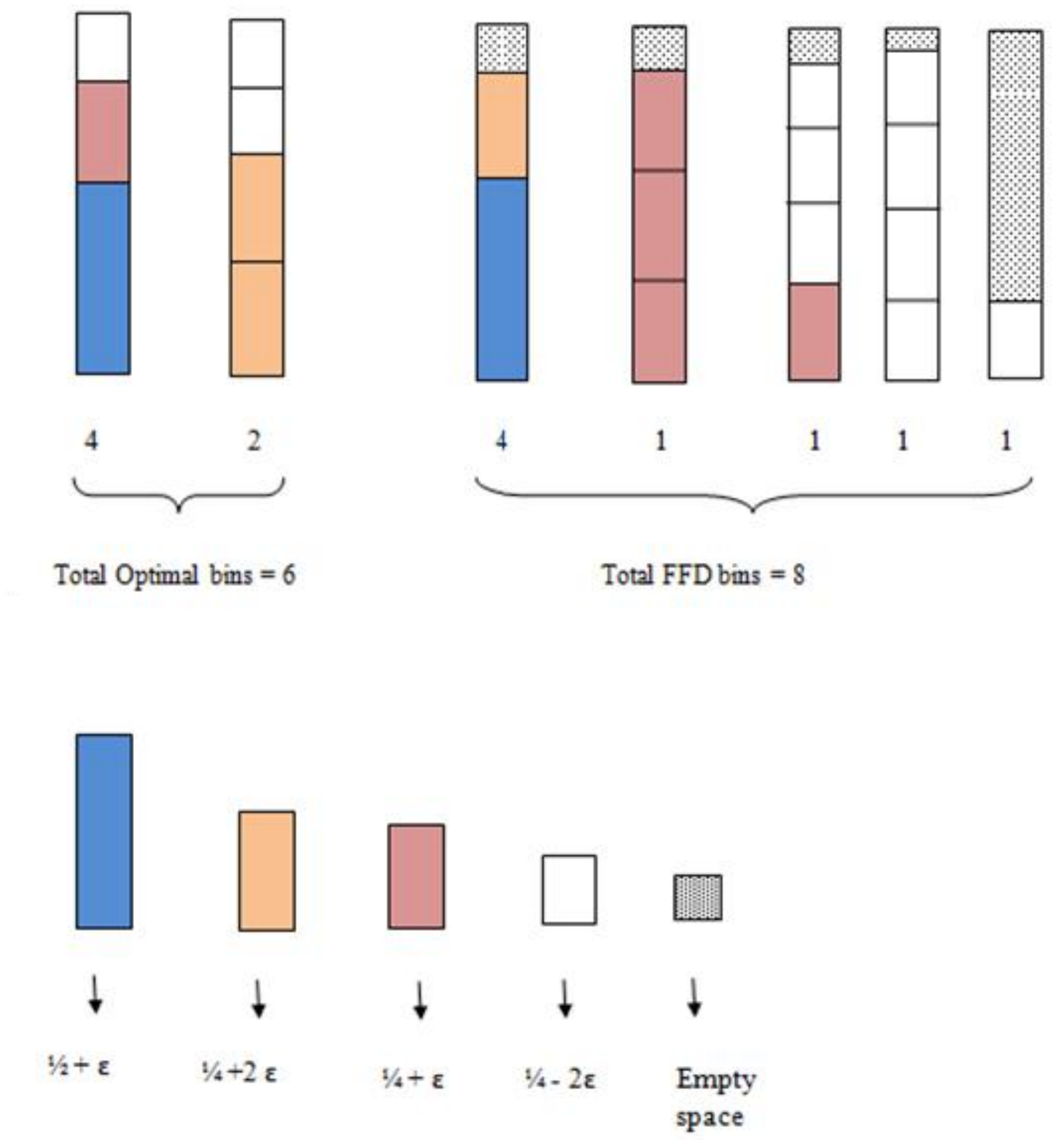


Fig 3: Another example for optimal bin distribution and distribution using FFD algorithm

And now using the first fit decreasing algorithm the items in the same instance *x* can be filled in 8 bins ($m_{FFD}(x) = 8$) and its distribution is as follows

- 4 bins containing 1 item of size ½ + **ε,** ¼ + 2**ε** in each bin.
- 1 bin containing 3 items of size ¼ + **ε**

- 1 bin containing 3 items of size ¼ - 2**ε** and 1 item of size ¼ + **ε**
- 1 bin containing 3 items of size ¼ - 2**ε**
- 1 bin containing 1 item of size ¼ - 2**ε**

So for the given instance *x*, if $m_{OPT}(x) = 6$ then $m_{FFD}(x) = 8$. So this example follows the statement $m_{FFD}(x) \leq \frac{11}{9} m_{OPT}(x) + \frac{6}{9}$.

Using the above result we can construct a corollary which can be illustrated as follows

Corollary:

We know that for an instance *x*, $m_{FFD}(x) \leq \frac{11}{9} m_{OPT}(x) + \frac{6}{9}$, Using this let $m_{OPT}(x) = M$ and $m_{FFD}(x) = N$ such that ∀M we can deduce maximum of N using the above statement.

In the given table below, M represents the number of bins used for optimal packing and N represents the maximum possible number of bins used by first fit decreasing algorithm. So for example let us consider an instance for which number of bins used for optimal packing is 5 i.e. M=5 then using $m_{FFD}(x) \leq \frac{11}{9} m_{OPT}(x) + \frac{6}{9}$ we can give N=6 i.e. the maximum possible number of bins used by first fit decreasing is 6 and this value cannot be bigger(for M=5). The table below shows the tightness of the asymptotic additive constant.

In the table below we give maximum values of N for different M values. Without the tight upper bound, we could not know the maximum value of N in many cases. Thus the table is as follows

$m_{OPT}(x)$ = M	$m_{FFD}(x)$= N	N-M
1	1	0
2	3	1
3	4	1
4	5	1
5	6	1
6	8	2
7	9	2
8	10	2
9	11	2
10	12	2
11	14	3
12	15	3
13	16	3
14	17	3
15	19	4
16	20	4
17	21	4
18	22	4
19	23	4
20	25	5
	And so on…	

Table 1: Maximum possible values of N for different M using $m_{FFD}(x) \leq \frac{11}{9}\, m_{OPT}(x) + \frac{6}{9}$

As we have discussed, we know that the bin packing problem does not have a PTAS. But we also need to know that this problem has an asymptotic polynomial time approximation scheme ($PTAS^{\infty}$). So let us describe the asymptotic polynomial time approximation scheme ($PTAS^{\infty}$) for bin packing.

2.1.3 Asymptotic Polynomial time Approximation Scheme for bin packing problem:

Since we know that the bin packing problem has an asymptotic polynomial time approximation ratio($PTAS^{\infty}$).Let us discuss about the algorithm which is an asymptotic PTAS.So the following algorithm was given by Fernandez de la Vega, W., and Lueker, G.S. [19] in 1981.

We have asymptotic PTAS for the bin packing problem. So the algorithm [19] for asymptotic PTAS consists of the following 5 steps

1. Eliminate small items from the instance which needs to be packed.
2. Group the remaining items into a constant number of size values.
3. Find optimal solution of the resulting instance.
4. Ungroup the items.
5. Re-insert small items.

Let us define certain variables and constants before each step is explained in detail

c – Integer constant (c>0) denotes number of different sizes of items.

δ – Constant ($\delta \leq 1$)

B – Size of the bin.

Let K be the instance of bin packing and for any rational constant $\delta \in (0, \frac{1}{2}]$

K_δ – instance obtained by eliminating all items whose sizes are less than δB.

Now, each step is explained in detail

Step 1:

In this step we eliminate small items (size< δB) from the instance K to obtain K_δ**.**

Step 2:

In this step, we group the remaining items in K_δ into groups of constant size values.

Procedure:

Given an instance K_δ, firstly arrange the items in a non-increasing order. Let n represent the number of items in a given instance and let p be a constant.

Consider $p \leq n$, let $m = \frac{n}{p}$, and partition the n items into m+1 groups.

Now we define a new instance $K_{\delta,g}$ with the same bin size B and size of all items in the i^{th} (for i= 2,3,…,m+1) group are made equal to the largest item in that respective partition or group. The distribution of items in the new instance can be well understood by the following example. Consider the instance x containing 11 items and whose sizes are {9,8,8,7,7,6,5,4,4,3,3} .Let p=3, So we can have four groups (since $m = \frac{n}{p}$ =3) .Let the four groups be G_1 = { it_1,it_2,it_3} G_2 = {it_4,it_5,it_6} G_3={it_7,it_8,it_9} and G_4 = {it_{10},it_{11}} respectively.

Now the new instance x_g has 8 items arranged in 3 groups, three items of size 7(corresponding to items in group G_2), three items of size 5(corresponding to items in group G_3) and two items of size 3(corresponding to items in group G_4).

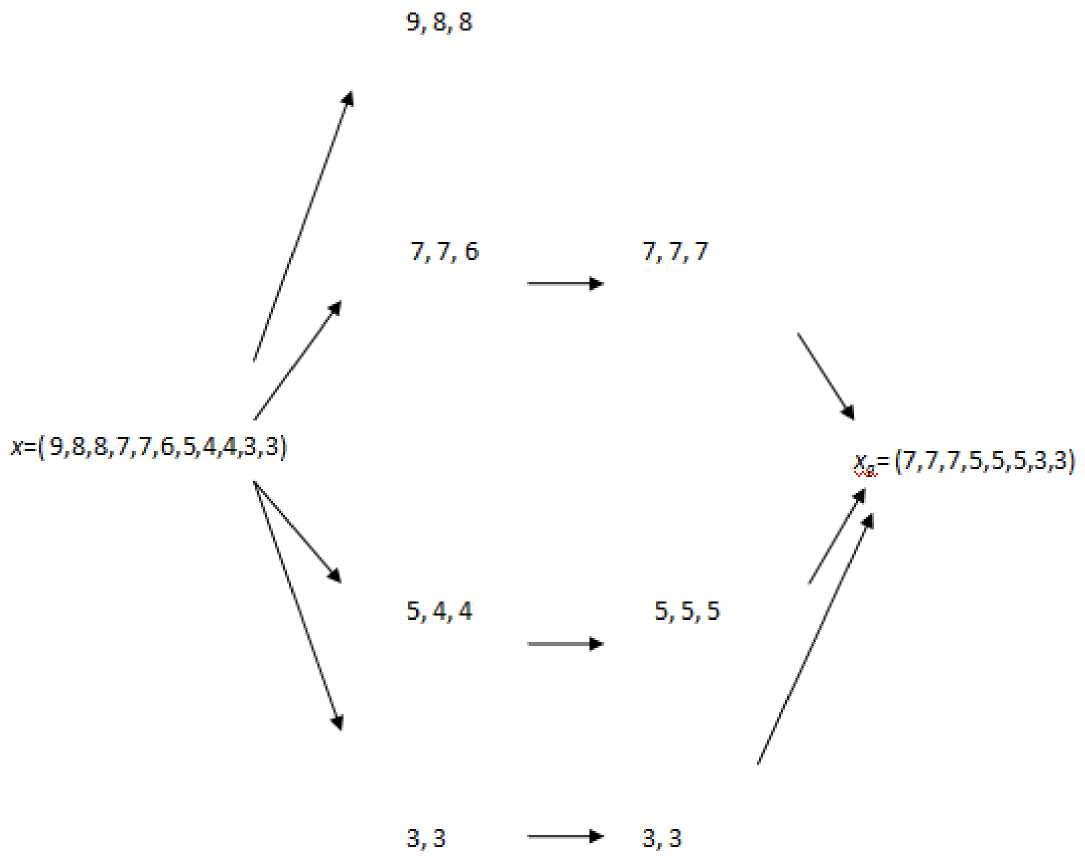


Fig 4: Example to explain the distribution and arrangement in step 2

So now for obtaining x_g, we eliminate the last group G_1 and then substitute each item in the group with the largest or the highest item in that group. Moreover, we can obtain x from x_g by simply adding p bins which can insert the p items (which were removed).

This gives,

$m^*(x_g) \leq m^*(x) \leq m^*(x_g) + p$

Where $m^*(x_g)$ is the optimal solution for instance x_g which indicates the filled bins and $m^*(x)$ is the optimal solution for instance x.

So if we are able to optimally solve x_g then we can find solution for x whose absolute error is at most p. We can generalize the above expression for the instance K_δ and it can be shown as $m^*(K_{\delta,g}) \leq m^*(K_\delta) \leq m^*(K_{\delta,g}) + p$ where p is the absolute error.

Step 3:

In this step we solve the instance ($K_{\delta,g}$) which we got from step 2. The procedure to solve the instance $K_{\delta,g}$ is as follows

We have instance $K_{\delta,g}$ and it can be re- written as $I = \{s_1 : n_1 , s_2 : n_2, \dots , s_c : n_c\}$ where c denotes number of different sizes of items and s_1,n_1 represents the size and the number of the items of size s_1 respectively.

For example: $I = \{ 3 : 4 , 5 : 2 , 7 : 1 \}$ contains 4 items of size 3, 2 items of size 5 , 1 item of size 7 and c=3

For bin packing of instance $K_{\delta,g}$ let each bin can be represented by a vector $\vec{b} = (b_1, b_2, \dots ,b_c)$ where $0 \leq b_i \leq n$ such that $\sum_{i=1}^{c} b_i\, s_i \leq B$. This implies that the packing of items in the bin should not exceed B.

We also draw an important result, so for each bin

$\sum_{i=1}^{c} b_i \leq \frac{1}{\delta} \sum_{i=1}^{c} b_i \frac{si}{B} \leq \frac{1}{\delta}$

The above result implies that the sum of number of different items in each bin should not be more than $\frac{1}{\delta}$.So we need to choose at most $\frac{1}{\delta}$ number of items for a bin from c (different size of items) types to fill in a bin and it is equal to

$$q = \binom{c + \frac{1}{\delta}}{\frac{1}{\delta}}$$

The value q denotes the possible bin types and depends on c and δ and does not depend on n.

Let us consider the example where the given instance is x_g

x_g = (7,7,7,5,5,5,3,3) now this instance for step3 can be re written as x_g = { 3 : 7 , 3 : 5 , 2 : 3 } and as we can observe c=3 and let us assume $\frac{1}{\delta} = 2$ then

$q = \binom{5}{2}$ = 10 ways

Indeed we can have 10 different possible ways to fill a bin(bin types), using the b vector to represent the bin types the result is as follows, (0,0,0), (0,1,0), (1,0,0), (0,0,1), (1,1,0), (0,1,1), (1,0,1), (2,0,0), (0,2,0), (0,0,2) but out of all the possible 10 solutions only 8 solutions are feasible because the other 2 bins violate the bound given by the size of the bin.(i.e., . $\sum_{i=1}^{c} b_i s_i \leq B$)

So (1, 1, 0) and (2, 0, 0) are the 2 bin types which are not feasible. Now one of the feasible solutions can be 2 bins of type (0,1,1) , 3 bins of type (1,0,0) and 1 bin of type (0,1,0) and the optimal solution can be 1 bin of type (0,2,0) , 2 bins of type (1,0,1) , 1 bin of type (1,0,0) and 1 bin of type (0,1,0). It is evident that the number of feasible solutions is bounded by $O(n^q)$ which implies that the instance can be solved in $O(n^q p(n))$ where p is a polynomial by exhaustively generating all these feasible solutions.

Step 4:

This step primarily deals with ungrouping the items. Now by using the expression

$m^*(K_{\delta,g}) \leq m^*(K_\delta) \leq m^*(K_{\delta,g}) + p$ we can obtain packing of items for K_δ by simply adding p bins in which we can insert the first p items(eliminated in step2).

Since we know the value of $m^*(K_{\delta,g})$, we can find $m^*(K_\delta)$from the above expression i.e., $m^*(K_\delta) \leq m^*(K_\delta) \leq m^*(K_{\delta,g}) + p$.

The result of $m^*(K_\delta)$ concludes step 4.

Step 5:

In this step we insert small items that were removed in step 1. Now using the first fit algorithm small items are inserted to the instance K_δ. Let us suppose items in K_δ instance are filled in M bins. So if the small items fit in the existing M bins then the packing is done, otherwise

- $M' \geq 1$ new bins have been created. So we can show all bins except at most one have an empty space i.e., at most δB.

This results in the expression

$$(1 - \delta)\,(M + M' - 1) \leq \sum_{i=1}^{n} \frac{s(u_i)}{B} \leq m^*(K)$$

$m^*(K)$ – optimal solution for instance K

$\sum_{i=1}^{n} \frac{s(u_i)}{B}$ - Sum of items in the given instance/ size of the bin

The above expression gives

$$M + M' \leq \frac{1}{1-\delta} m^*(K) + 1 \leq (1 + 2\delta)\, m^*(K) + 1$$

So, given a packing of instance K_δ with M bins we can find in polynomial time a solution for K instance whose measure is at most

$$\text{Max}\ (M, (1 + 2\delta) m^*(K) + 1)$$

Here $r = 1 + 2\delta$, $p = \frac{(r-1)^2}{2}$ and Max (M, $rm^*(K)$ +1) gives solution for packing

Another observation is that if $r \geq 2$, then First fit algorithm achieves the desired performance ratio. So PTAS for bin packing is restricted to $r < 2$. Now elaborating on M, from step4 it is understood that $m^*(K_\delta) \leq m^*(K_{\delta,g}) + p$ where $m^*(K_\delta) = M$ bins. Considering $m^*(K_{\delta,g}) + p$, since all items in K_δ have items of size at least δB, we conclude $\delta n' \leq m^*(K_\delta)$, here n' is the number of items in the instance $K_{\delta.}$(all items size is at least δB)

So, $p \leq \frac{(r-1)^2}{2} n' + 1 = (r-1)\, \delta n' + 1 \leq (r-1)\, m^*(K_\delta) + 1$

From step 4

$m^*(K_{\delta,g}) + p \leq m^*(K_\delta) + p$

Substituting p

$m^*(K_{\delta,g}) + p \leq m^*(K_\delta) + (r - 1)\, m^*(K_\delta) + 1 = r\, m^*(K_\delta) + 1$

Finally, by replacing $r = (1+2\delta)$ and M with $r\, m^*(K_\delta) + 1$ in the final expression we get

Max $(rm^*(K_\delta) + 1, rm^*(K)+1)$

So given a packing of K_δ with 'M ' bins we can find a solution for K in polynomial time whose measure is at most Max $(rm^*(K_\delta) + 1, rm^*(K)+1)$.

Asymptotic PTAS for bin packing: (algorithm in brief)

```
Input: Instance $K$ of bin packing and $1 < r < 2$

Begin

    $r = 1+2\delta$;   $p= \frac{(r-1)^2}{2} n$; $B$ is the size of the bin;

    Eliminate small items from the instance $K$ whose size $< \delta B$, Let the resulting
    instance be $K_\delta$ and $n$ be the number of items in $K_\delta$

    Partition $K_\delta$ instance into $m+1$ groups where $m=\frac{n}{p}$ , remove the last group and
    combine the rest to form the instance $K_{\delta,g}$ .

    Find the optimal solution for $K_{\delta,g}$ and let the result be $m^*( K_{\delta,g})$

    Insert first $p$ items into the $p$ new bins and calculate $m^*( K_\delta )$.

    Using First Fit algorithm, reinsert small items to the instance $K_\delta$, calculate and
    return the result Max $(rm^*(K_\delta ) + 1 , rm^*(K)+1)$

End
```

2.2 ONLINE ALGORITHMS FOR BIN PACKING:

In the online version of bin packing algorithms items are packed in the bin as per the input sequence. The current item is packed in the bin before the next item arrives and once an item is packed in a particular bin it cannot be moved. Now let us discuss about different online algorithms for bin packing algorithms.

2.2.1 Next Fit Algorithm:

This algorithm is one of the most basic online algorithms. Before we explain the algorithm let us know about a few parameters, b_j which represents the j^{th} bin where $j_{=-}$ 1,2,3…,n and a_i represents i^{th} item of the input sequence where $i_{=}$1,2,3…,n. In this algorithm, initially all bins are empty and we begin with bin $j = 1$ and item $i = 1$. So if bin b_1 has enough space for item a_1 to fit then we assign item a_1 to bin b_1, otherwise bin b_1 is closed and a new bin b_{j+1} (i.e. b_2) is opened to fill item a_i. A closed bin is never opened again for further allocation of items. In this manner we repeat the process till the input sequence ends.

Theorem: Given an instance x of ONLINE BIN PACKING, the algorithm Next Fit returns a result with value $m_{NF}(x)$ such that $m_{NF}(x)/m_{OPT}(x) \leq 2$ where $m_{OPT}(x)$ denotes the optimal solution for an instance x.

Proof: This proof was taken from a paper written by D.S.Johnson [1]. Firstly, 'A' denotes the sum of all the item sizes in the bin. In the next fit algorithm, only one bin (last used bin) is kept open and when an item doesn`t fit, that bin is closed and a new bin is opened. So at any given time only one bin is open and once the bin is closed it cannot be opened. Since the sum of the items of any two consecutive bins is always greater than

1, the number of bins used by Next fit algorithm is less than 2*A*.This is because on an average the bins are more than half full. On the other hand, the optimal solution uses bins which are at least the total size of the items (*A*). So $m_{OPT}(x) \geq A$

Hence we have $m_{NF}(x)/m_{OPT}(x) \leq 2$

The following example clearly presents the algorithm and the given instance follows the bound stated in this theorem.

Example:

Consider there exists an instance of *4n* items and the order of the items is as follows: $I = \{\frac{1}{2}, \frac{1}{2n}, \frac{1}{2}, \frac{1}{2n}, \ldots, \frac{1}{2}, \frac{1}{2n}\}$ (each pair is repeated 2n times). Fig (a) represents the optimal solution where the 2*n* items of size $\frac{1}{2}$ are filled in *n* bins and the remaining 2*n* items of size $\frac{1}{2n}$ are filled in a single bin. Hence the $m_{OPT}(I) = n+1$. Fig (b) represents the approximate solution of Next Fit algorithm where 2 items of size $\frac{1}{2}$ and $\frac{1}{2n}$ are filled in each bin, thereby occupying 2*n* bins to fill the given instance. Hence $m_{NF}(I) = 2n$.

Algorithm

Consider an instance *x* containing items a_i where $i \in (1,2 \ldots, n)$ and no of bins $b \longleftarrow 1$

```
Begin
        For $i := 1$ to $n$ do
        If the item $a_i$ can fit in the opened bin then
        Insert $a_i$ into the bin
        Else
        Insert the item $a_i$ into a new bin
        $b := b + 1$
```

```
    Return $b$
End
```

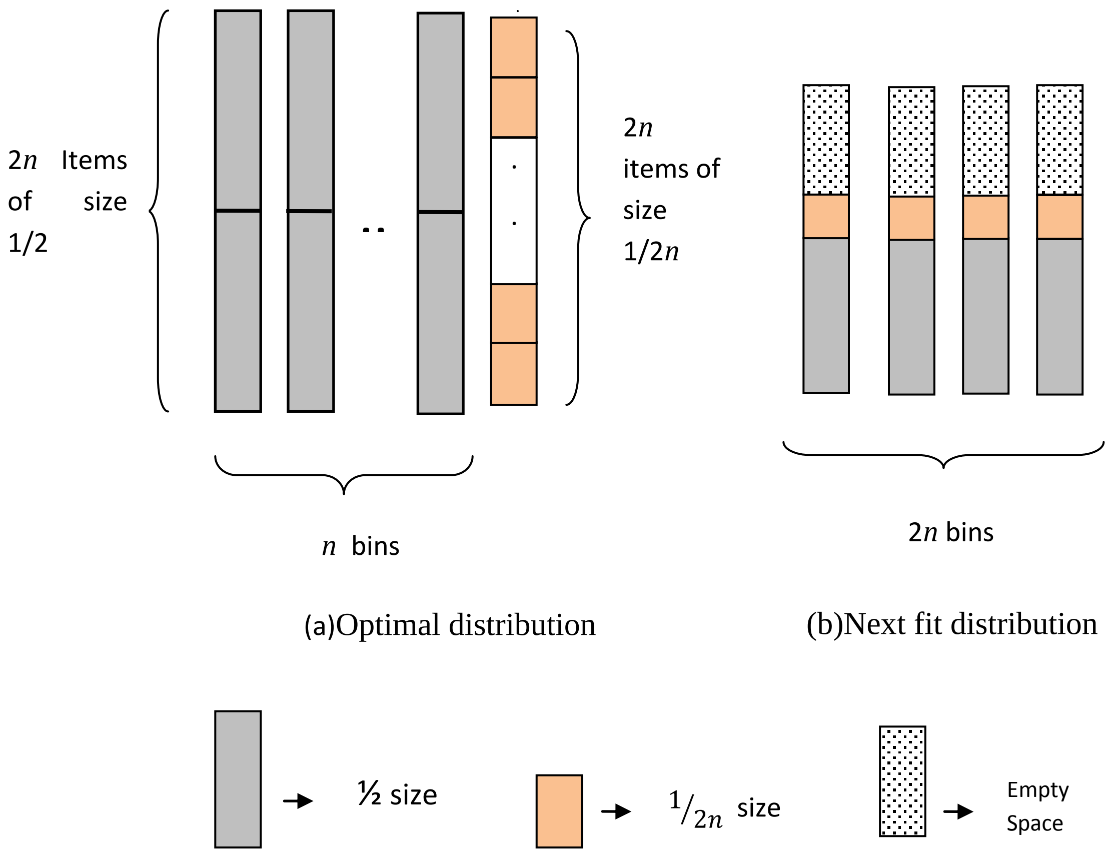


Fig 5: Example for Optimal bin distribution Vs. distribution using Next-fit algorithm

In fact, there is First Fit algorithm which has a better performance ratio than Next Fit when it comes to online algorithms for Bin Packing. Let us discuss about the first fit algorithm in the next section.

2.2.2 First Fit Algorithm:

Given an instance x of ONLINE BIN PACKING, the algorithm First Fit returns a result with value $m_{FF}(x)$ such that $m_{FF}(x) \leq 1.7\ m_{OPT}(x)+2$ where $m_{OPT}(x)$ denotes the optimal solution for an instance x. The numeric "2" represents the additive constant.

Algorithm

Consider bins b_j where $j \in (1,2 \dots , n)$

Consider an instance x containing items a_i where $i \in (1,2 \dots , n)$

```
    Begin
        For $i$ := 1 to $n$ do
        For $j$ := 1 to $n$ do
        if item $a_i$ can fit in the bin $b_j$
            then
                Insert $a_i$ into the bin
                Break; //exit for $j$ loop
        //continue for $i$ loop
    End
```

We know that the first fit algorithm is one of the online algorithms for the bin packing problem. The only difference between first fit decreasing (FFD) algorithm and first fit algorithm is the sorting step where the items are arranged in non-increasing order. Since first fit is an online algorithm, the sorting step is skipped and the distribution follows.

Ullman and Garey, Graham, and Ullman introduced the study of bin packing analysis in their respective papers. They show that the competitive ratio for the first-fit and best-fit satisfy:$R_{FF=}\frac{17}{10}$ and $R_{BF}\geq \frac{17}{10}$.The following proof was given by D.S.Johnson, Demers, Ullamn, Garey and Graham**[9]**.

2.2.2.1 Proof for the Upper bound:

Now for any given instance *I* let us prove $m_{FF}(I) \leq 1.7\ m_{OPT}(I)+2$, where $m_{FF}(I)$ gives the number of bins filled using first fit algorithm and $m_{OPT}(I)$ gives the optimal distribution for the given input sequence.

Before we begin with the proof let us define the following weight function:

$$w(it_i) = \begin{cases} \frac{6}{5}it & \text{if } 0 \leq it \leq \frac{1}{6} \\ \frac{9}{5}it - \frac{1}{10} & \text{if } \frac{1}{6} \leq it \leq \frac{1}{3} \\ \frac{6}{5}it + \frac{1}{10} & \text{if } \frac{1}{3} \leq it \leq \frac{1}{2} \\ 1 & \text{if } \frac{1}{2} < it \end{cases}$$

For any given instance $I = it_1, it2, \dots, it_n$, where it_i denotes the i^{th} item in the given input sequence. Let us define certain functions and variables which will be used in the proof.

$w(it)$	-	Weight function for item *it*. It gives us the weight for the respective *it* value.
$W(I) = \sum_{i=1}^{n} w(it_i)$	-	Weight of an instance *I* calculated by the summation of all the weights of the items in the instance *I*.
$W(B) = \sum_{j=1}^{t} w(it_{i_j})$	-	Weight of the bin *B* where $\{it_{i_j} \mid j = 1,2,\dots,t\}$ be the items assigned to bin *B*.

For the proof of the upper bound $m_{FF}(I) \leq \frac{17}{10} m_{OPT}(I)+2$,we need to prove the following 2 lemmas. | j= 1,2,…,t} be the items assigned to bin B.

Lemma 1:

For any bin B filled with items, we have $W(B) \leq \frac{17}{10}$.

Lemma 2:

If for any given instance I, first fit algorithm uses k ($m_{FF}(I)= k$) number bins to fill the items then

$\sum_{j=1}^{k} W(B_i) \geq m_{FF}(I) - 2$ which gives $m_{FF}(I) \leq W(I)+2$.

Now proving the above given 2 lemmas would prove the upper bound for the first fit algorithm.

Proof of lemma 1:

Now for the given instance, if every item has size $\leq \frac{1}{2}$ then

$$\frac{w(it_i)}{it_i} \leq \frac{3}{2} < \frac{17}{10},$$

which easily completes the proof. But, in any given scenario we cannot assume that all the items in the instance has size$\leq \frac{1}{2}$ so we consider some $it_i > \frac{1}{2}$ and $it_{i_1}, it_{i_2},\ldots, it_{i_t}$ represents the rest of the items that packed in the same bin as it_i. Since $it_i > \frac{1}{2}$, the other items in the same bin ($it_{i_1}, it_{i_2},\ldots, it_{i_t}$) occupy size $< \frac{1}{2}$ i.e. $\sum_j^t it_{i_j} < \frac{1}{2}$.

And therefore we need to show that $\sum_j^t w(it_{i_j}) < \frac{7}{10}$.

Without loss of generality, let us assume $c \leq \frac{1}{3}$, so even if there are items(like it_{i_j}) in the instance existing with size $>\frac{1}{3}$, we can infer it_{i_j} as two items i.e.

$it_{i_j}^1 = \frac{1}{3}$ and $it_{i_j}^2 = it_{i_j} - \frac{1}{3} < \frac{1}{6}$.

As the weight function *w* is linear and contains same slope for item size $it < \frac{1}{6}$ and $\frac{1}{3} \leq it < \frac{1}{2}$

$w(it_{i_j}) = w(\, it_{i_j}^1) + w(it_{i_j}^2)$.

Similarly we can assume that in any first fit distribution of instance there is at most one item $it_{i_j} < \frac{1}{6}$. So even if there are more items with size$<\frac{1}{6}$ for instance if two items $it_{i_1}, it_{i_2} < \frac{1}{6}$ then we can combine them into a single item it_{i_c} such that $it_{i_c} < \frac{1}{3}$.The weight function for the item it_{i_c} is as follows

$w(it_{i_c}) \geq w(it_{i_1}) + w(it_{i_2})$.

In the beginning of the proof we noticed that we can easily deduce the proof if all the items in the instance are of size$<\frac{1}{2}$. Since we need to prove the lemma for any instances we assumed that in a given bin $it_{i} > \frac{1}{2}$ and the other items in the same bin

$(it_{i_1}, it_{i_2}, \ldots., it_{i_t})$ occupy size $< \frac{1}{2}$ i.e. $\sum_j^t it_{i_j} < \frac{1}{2}$ and therefore prove $\sum_j^t w(it_{i_j}) < \frac{7}{10}$.

Using these considerations let us analyze the following cases to prove the lemma

- Case 1 :If t=1 we have two sub cases:

  i) If $it_{i_1} < \frac{1}{6}$, then $w(it_{i_1}) < \frac{1}{5} < \frac{7}{10}$.

ii) If $\frac{1}{6} \leq it_{i_1} \leq \frac{1}{3}$, then $w(it_{i_1}) < \frac{3}{5} - \frac{1}{10} < \frac{7}{10}$

If we have any items it_{i_1} in the interval $[\frac{1}{3}, \frac{1}{2})$ then using the procedure mentioned above we split it to two items with size$\leq \frac{1}{3}$.

- Case 2 :If t=2 we have two subcases

  i) If $it_{i_1} < \frac{1}{6} \leq it_{i_2} \leq \frac{1}{3}$ then $w(it_{i_1}) + w(it_{i_2}) < \frac{1}{5} + \frac{3}{5} - \frac{1}{10} = \frac{7}{10}$.

  ii) If $\frac{1}{6} \leq it_{i_1}, it_{i_2} \leq \frac{1}{3}$ then $w(it_{i_1}) + w(it_{i_2}) < \frac{9}{5}(it_{i_1} + it_{i_2}) - \frac{2}{10} < \frac{7}{10}$, since $it_{i_1} + it_{i_2} < \frac{1}{2}$.

- Case 3 :If t=3 we have two subcases

  i) If $it_{i_1} < \frac{1}{6} \leq it_{i_2}, it_{i_3} \leq \frac{1}{3}$, we have $w(it_{i_1}) + w(it_{i_2}) + w(it_{i_3}) \leq \frac{6}{5} it_{i_1} + \frac{9}{5}(it_{i_3} + it_{i_2}) - \frac{2}{10}$,Since $it_{i_1} + it_{i_2} + it_{i_3} < \frac{1}{2}$ which gives $it_{i_2} + it_{i_3} < \frac{1}{2} - it_{i_1}$. Substituting this in the above inequality gives $\frac{6}{5} it_{i_1} + \frac{9}{5}(it_{i_3} + it_{i_2}) - \frac{2}{10} < \frac{7}{10} - \frac{3}{5} it_{i_1} < \frac{7}{10}$.

  ii) If $it_{i_1} \geq \frac{1}{6}$ and $it_{i_2}, it_{i_3} \leq \frac{1}{3}$, we have $w(it_{i_1}) + w(it_{i_2}) + w(it_{i_3}) < \frac{6}{5}(it_{i_1} + it_{i_2} + it_{i_3}) - \frac{3}{10} < \frac{7}{10}$, since $it_{i_1} + it_{i_2} + it_{i_3} < \frac{1}{2}$.

- Case 4: If $t > 3$ which is not possible since there is at most one item $it_{i_j} < \frac{1}{6}$ and the other items are greater than $\frac{1}{6}$.

  Hence in this way we prove this lemma.

Before we begin with the proof of lemma 2 we need to know about coarseness.

Coarseness of a Bin:

For any bin B_i, we define coarseness as α such that there exists a bin B_j where $j<i$ and size of the bin B_j, $s(B_j)= 1- \alpha$, and for other bins B_k ,$k< i$, $s(B_k) \geq 1- \alpha$. The coarseness of the first bin is 0. In this definition $s(B_j)$, $s(B_k)$ denotes the size of the bin B_j, B_k which implies the amount of space occupied by the items in those bins.

A bin B with coarseness α contains items packed whose size is greater than α otherwise those items are packed in one of the previous bins. Coarseness α for a bin B can be considered as the maximum available space in the any of previous bins of B where the bins are filled using first fit algorithm.

Lemma 2:

If for any given instance I, first fit algorithm uses k ($m_{FF}(I)= k$) number of bins to fill the items then $\sum_{j=1}^{k} W(B_i) \geq m_{FF}(I) - 2$.

Proof:

Now if a bin B contains an item *it* whose size $> \frac{1}{2}$ then $w(B) \geq 1$, so our emphasis will be on those bins containing items of size $\leq \frac{1}{2}$ and as a result $w(B) < 1$. Let $B_1, B_2, \ldots, B_z$ be list of non empty bins filled in this particular order. Now coming to coarseness all bins have coarseness $\alpha < \frac{1}{2}$, if a bin B has coarseness $\alpha \geq \frac{1}{2}$ then it implies that in bin B there is only one item with size$> \frac{1}{2}$.

Now to prove this lemma we make use of the following claims which also have to be proved. The two claims are as follows

Claim 1:

For a bin *B* with coarseness $\alpha < \frac{1}{2}$ and containing items $it_1 \geq \ldots \geq it_k$, if $\sum_{i=1}^{k} it_i \geq 1 - \alpha$ then $\sum_{i=1}^{k} w(it_i) \geq 1$.

Claim 2:

If a bin *B* is packed with items $it_1 \geq it_2 \geq \ldots \geq it_k$ and $\sum_{i=1}^{k} w(it_i) = 1 - \beta$, $\beta > 0$ then either

(i) $k = 1$ and $it_1 \leq \frac{1}{2}$.

OR

(ii) $\sum_{i=1}^{k} it_i \leq 1 - \alpha - \frac{5}{9}\beta$.

Using the above two claims we will prove the bound for the first fit algorithm and the two claims will be proven later.

Let $W(B_i) = 1 - \beta_i$, $\beta_i > 0$ where $W(B_i)$ represents the weight of the bin B_i and let $\alpha_i < \frac{1}{2}$ represents the coarseness of a bin B_i. Let it_1^i, it_2^i,… to denote the items packed in B_i.

Let us assume a variable $l > 1$, now using the definition of coarseness and claim 2, for $1 < i \leq l$.

$\sum_j it_j^{i-1} \geq 1 - \alpha_i \xrightarrow{\qquad}$ (1) is derived using the definition of coarseness.

$\sum_j it_j^{i-1} \geq 1 - \alpha_{i-1} - \frac{5}{9}\beta_{i-1} \xrightarrow{\qquad}$ (2) is derived from claim 2.

Combining (1) and (2) we get

$\alpha_i \geq 1 - \sum_j it_j^{i-1} \geq 1 - \alpha_{i-1} - \frac{5}{9}\beta_{i-1} \xrightarrow{\hspace{3em}}$ (3)

Considering (3) we get

$\alpha_i \geq \alpha_{i-1} + \frac{5}{9}\beta_{i-1}$

$\frac{9}{5}(\alpha_i - \alpha_{i-1}) \geq \beta_{i-1}$

Applying summation, we get

$\sum_{i=1}^{l} \beta_i \leq \frac{9}{5} \sum_{i=2}^{l} \alpha_\text{i} - \alpha_{\text{i}-1} = \frac{9}{5}(\alpha_l - \alpha_1) \leq \frac{9}{5} . \frac{1}{2} < 1.$

Here if we consider case(i) of claim 2 then $\alpha \geq \frac{1}{2}$ but we assumed $\alpha < \frac{1}{2}$ and hence the previous bin size is filled with 1- $\alpha > \frac{1}{2}$, $W(B) \geq 1$ but we know that $W(B)$=1- β which is a contradiction. So case(i) of claim2 does not hold. Since β_l cannot exceed 1, we have $\sum_{i=1}^{l} \beta_i \leq 2$. Now let m be the number of the bins other than $B_1, \dots, B_l$ used by first fit so that $m+l= m_{FF}(I)$.

$W(I) \geq m + \sum_{i=1}^{l} W(B_i) = m + \sum_{i=1}^{l} 1 - \sum_{i=1}^{l} \beta_i = m + l - \sum_{i=1}^{l} \beta_i \geq m_{FF}(I) - 2.$

So we get, $m_{FF}(I) \leq W(I) + 2$.Hence proved.

Combining this with lemma 1 we get

$m_{FF}(I) \leq \frac{17}{10} m_{OPT}(I) + 2.$

Now we are left dealing with the proving claim1 and claim 2.

Claim 1:

For a bin B with coarseness $\alpha < \frac{1}{2}$ and containing items $it_1 \geq \ldots \geq it_k$, if $\sum_{i=1}^{k} it_i \geq 1 - \alpha$ then $\sum_{i=1}^{k} w(it_i) \geq 1$.

Proof:

In an given instance for $it_1 > \frac{1}{2}$, the result is immediate since $w(it_1) = 1$.So we therefore assume that $it_1 \leq \frac{1}{2}$.If $k \geq 2$ then by the definition of coarseness, we have $it_1 \geq it_2 \geq \alpha$ where α is the coarseness of the bin. Now we need to consider cases based on the different values of α.

Case 1: If $\alpha \leq \frac{1}{6}$, then $\sum_{i=1}^{k} it_i \geq 1 - \alpha \geq \frac{5}{6}$. Now for all items in the range $[0, \frac{1}{2}]$ the slope of weight function w is greater than $\frac{6}{5}$, i.e. $\frac{w(it)}{it} \geq \frac{6}{5}$ for $0 \leq it \leq \frac{1}{2}$ where it represents the item whose size is in the range $[0, \frac{1}{2}]$. Since $\frac{w(it)}{it} \geq \frac{6}{5}$ for $0 \leq it \leq \frac{1}{2}$, we have $\sum_{i=1}^{k} w(it_i) \geq \frac{6}{5} . \frac{5}{6} \geq 1$.

Case 2: If $\frac{1}{6} \leq \alpha \leq \frac{1}{3}$, we get three subcases based on the value of k

(i) $k=1$. This case is not possible since $it_1 \leq \frac{1}{2}$, we have $\sum_{i=1}^{k} it_i \geq 1 - \alpha$ which gives $1 - \alpha \leq \frac{1}{2}$,this is a contradiction as we have $\alpha \leq \frac{1}{3}$.

(ii) $k=2$.If both $it_1 \geq it_2 \geq \frac{1}{3}$, then $\sum_{i=1}^{k} it_i \geq 2(\frac{6}{5} . \frac{1}{3} + \frac{1}{10}) = 1$. Now if both the items it_1, it_2 are less than $\frac{1}{3}$, then $it_1 + it_2 < \frac{2}{3} < 1 - \alpha$, contradicts our hypothesis. If $it_1 \geq \frac{1}{3}$ and $it_2 < \frac{1}{3}$, as $it_1 \geq it_2 > \alpha$ and $\alpha \leq \frac{1}{3}$, we get

$w(it_1) + w(it_2) = (\frac{6}{5}\, it_1 + \frac{1}{10}) + (\frac{9}{5}\, it_2 - \frac{1}{10}) = \frac{6}{5}\,(it_1 + it_2) + \frac{3}{5}\, it_2$, Since

$it_1 + it_2 \geq 1 - \alpha$ and $it_2 > \alpha$ we get $w(it_1) + w(it_2) = \frac{6}{5}\,(it_1 + it_2) + \frac{3}{5}\, it_2$

$\geq \frac{6}{5}\,(1 - \alpha) + \frac{3}{5}\,\alpha \geq 1$,since $\alpha \leq \frac{1}{3}$.

(iii) $k \geq 3$. The working procedure for this case is similar to the

previous step. So if both $it_1 \geq it_2 \geq \frac{1}{3}$ then we have the claim. If $it_1 \geq$

$\frac{1}{3}$ and $it_2 < \frac{1}{3}$, as $it_1 \geq it_2 > \alpha$ and $\alpha \leq \frac{1}{3}$, we get

$\sum_{i=1}^{k} w(it_i) = w(it_1) + w(it_2) + \sum_{i=3}^{k} w(it_i) \geq (\frac{6}{5}\, it_1 + \frac{1}{10}) + (\frac{9}{5}\, it_2 -$

$\frac{1}{10}) + \frac{6}{5}\sum_{i=3}^{k} it_i = \frac{6}{5}\sum_{i=1}^{k} it_i + \frac{3}{5}\, it_2$,

Since $\sum_{i=1}^{k} it_i \geq 1 - \alpha$ and $it_2 > \alpha$ we get

$\sum_{i=1}^{k} w(it_i) \geq \frac{6}{5}\sum_{i=1}^{k} it_i + \frac{3}{5}\, it_2 \geq \frac{6}{5}\,(1 - \alpha) + \frac{3}{5}\,\alpha \geq 1$, since $\alpha \leq \frac{1}{3}$.

Considering if the first two items are $< \frac{1}{3}$,i.e. $\frac{1}{3} > it_1 \geq it_2 > \alpha$ then

$\sum_{i=1}^{k} w(it_i) = w(it_1) + w(it_2) + \sum_{i=3}^{k} w(it_i) \geq (\frac{9}{5}\, it_1 - \frac{1}{10}) + (\frac{9}{5}\, it_2 -$

$\frac{1}{10}) + \frac{6}{5}\sum_{i=3}^{k} it_i = \frac{6}{5}\sum_{i=1}^{k} it_i + \frac{3}{5}\, it_2 + \frac{3}{5}\, it_2 - \frac{1}{5}$.

Since $\sum_{i=1}^{k} it_i \geq 1 - \alpha$ and $it_1 \geq it_2 > \alpha$ we get

$\sum_{i=1}^{k} w(it_i) \geq \frac{6}{5}\sum_{i=1}^{k} it_i + \frac{3}{5}\, it_2 + \frac{3}{5}\, it_2 - \frac{1}{5} \geq \frac{6}{5}\,(1 - \alpha) + \frac{3}{5}\,\alpha + \frac{3}{5}\,\alpha -$

$\frac{1}{5} = \frac{6}{5}\,(1 - \alpha) + \frac{6}{5}\alpha - \frac{1}{5} = \frac{6}{5} - \frac{6}{5}\,\alpha + \frac{6}{5}\alpha - \frac{1}{5} = 1$.Now let's move on to the next

case.

*Case 3:*If $\frac{1}{3} < \alpha < \frac{1}{2}$, we get two subcases to deal with i.e. when k=1,2.If k=1 then $it_1 \geq 1 - \alpha > \frac{1}{2}$. *Since* $it_1 \geq \frac{1}{2}$, we have $w(it_1) = 1$. Now for $k \geq 2$, we need to do case analysis similar to the previous subcases in case2.

In this manner we can prove the claim 1.

Proof of Claim 2:

Claim 2:

If a bin B is packed with items $it_1 \geq it_2 \geq \ldots \geq it_k$ and $\sum_{i=1}^{k} w(it_i) = 1 - \beta$, $\beta > 0$ then either

(i) $k = 1$ and $it_1 \leq \frac{1}{2}$.

or

(ii) $\sum_{i=1}^{k} it_i \leq 1 - \alpha - \frac{5}{9}\beta$.

Proof:

If k=1and $it_1 > \frac{1}{2}$,then it is not possible that $\beta > 0$ since $w(it_1) = 1$.Now if $k \geq 2$ then by the definition of the coarseness of the bin we have $it_1 \geq it_2 \geq \alpha$. Let $\sum_{i=1}^{k} it_i = 1 - \alpha - \gamma$. Then we construct a bin packed with items $it_3, it_4, \ldots it_k$ and two items δ_1, δ_2 respectively. Now let $\delta_i \geq it_i$ and $\delta_1 + \delta_2 = it_1 + it_2 + \gamma$ and both $\delta_1, \delta_2 < \frac{1}{2}$.Using claim1 for this bin we get

$$\sum_{i=3}^{k} w(it_i) + w(\delta_1) + w(\delta_2) \geq 1. \qquad \longrightarrow (1)$$

For the items in the range $[0, \frac{1}{2})$ the slope of weight function w is $\leq \frac{9}{5}$, so we get

$$w(\delta_1) + w(\delta_2) \leq w(it_1) + w(it_2) + \frac{9}{5}\gamma. \qquad \longrightarrow (2)$$

Now substituting (2) in (1) we get

$\sum_{i=3}^{k} w(it_i) + w(it_1) + w(it_2) + \frac{9}{5}\gamma \geq 1.$

$\sum_{i=3}^{k} w(it_i) + w(it_1) + w(it_2) \geq 1- \frac{9}{5}\gamma.$

But we know that $\sum_{i=1}^{k} w(it_i)$= 1- β, substituting this in the above inequality gives

$\beta \geq 1- \frac{9}{5}\gamma$ which gives $\gamma \geq \frac{5}{9}\beta$.

Hence the claim holds.

2.2.3 Best Fit Algorithm:

Best fit is another online algorithm that packs the input item according to the following rule: while trying to pack item a_i, the best fit algorithm assigns the item to the bin whose empty space is minimum. If the item a_i is unable to fit in any of the opened bins then a new bin is opened to pack that item a_i.

Algorithm

Consider bins b_j where $j \in (1,2 \dots , n)$

Consider an instance x containing items a_i where $i \in (1,2 \dots , n)$ and no of bins $b \longleftarrow 1$

```
Begin
     for $i$ := 1 to $n$ do
          Sort bins $b_j$ in decreasing order such that the bin with minimum space
          available is placed first. Let the sorted sequence be { $B_1, B_2 \dots , B_n$}
     for $k$:=1 to $n$ do
```

```
        if the item $a_i$ can fit in the bin $B_k$ then
                Insert $a_i$ into the bin
                break; // exit for $k$ loop
                //continue for $i$ loop
End
```

Though First Fit and Best Fit are better than Next Fit, the worst case performance is the same for all the three algorithms. So there is a need for better approximation algorithm. A better approximation algorithm is obtained by observing that the worst performance for First Fit (and best fit) seems to occur when smaller items appear before larger items in a given instance.

2.2.4 Harmonic Algorithm:

Under online algorithms for bin packing problem, we have another algorithm **[8]** based on non-uniform partitioning of interval (0, 1] into M sub-intervals. Consider an instance $L = \{it_1, it_2, it_3 \ldots , it_n\}$, where $0 < s(it_i) \leq 1$, $s(it_i)$ denotes the size of item it_i in the given instance L. In this algorithm, the interval (0 , 1] is partitioned into harmonic sub intervals $I^M = \{ (0,\frac{1}{M}] , (\frac{1}{M}, \frac{1}{M-1}] , \ldots (\frac{1}{2} , 1] \}$ where M is a positive integer. Now each item it_i is classified and put in one of these sub intervals based on their size. An item it_i is called I_k item, if the item size is in the interval $I_k = (\frac{1}{k+1}, \frac{1}{k}]$, $k>1$.If the item size is in the interval $I_M = (0,\frac{1}{M}]$,then the item is called I_M item. In this manner all the items in the instance or the sequence are classified. So the I_k filled bin (bin with all I_k items) packs exactly k items irrespective of the actual sizes of the items. Using this background, we discuss about Algorithm Harmonic.

This algorithm opens an active bin for each type i.e., one bin of I_1 type items, one bin of I_2 type items and so on. Hence a total of M bins are active at any given time (since M sub intervals).When an item it_i belonging to sub-interval I_k (I_k item) arrives, it is packed in the corresponding active bin, if that is bin filled and has no enough space to pack item I_k then it is closed and a new bin is open for that sub-interval items. This harmonic algorithm is independent of the arriving order of the items. A disadvantage with this algorithm is when items of size> $\frac{1}{2}$ are packed then one bin per item is used resulting in wasting a lot of free space in each single bin. Now based on the harmonic algorithm we have *k* – binary algorithm which works on the lines of harmonic algorithm.

2.2.5 *k* – Binary Algorithm:

The Algorithm *k* – binary partitions the interval (0,1] into sub intervals in the following given manner (0,1] = $\bigcup_{k=1}^{M} I_k$ where

$$I_k = \begin{cases} \left(\frac{1}{2^k}, \frac{1}{2^{k-1}}\right] & \text{for } 1 \leq k < M \\ \left(0, \frac{1}{2^{M-1}}\right] & \text{for } k = M \end{cases}$$

M represents the number of partitions. For example, if M=3 then the interval (0,1] is partitioned into three intervals $(0, \frac{1}{4}]$, $(\frac{1}{4}, \frac{1}{2}]$, $(\frac{1}{2}, 1]$.

Algorithm:

```
Begin
    Partition interval (0,1] into $M$ sub-intervals $I_k$, $k = 1,2,\ldots,M$.
$B_{count} = M$ ; Open $M$ new bins, one for each sub-interval.
for k= 1 to $M$
        $b_k = 0$,  $b_k \leq 1$ and $1 < k \leq M$, $b_k$ – bin size for sub interval $I_k$
end for
for $i = 1$ to $N$ do
if  $0 < it_i < \frac{1}{2^M}$ then $it_i$ is an $I_M$ item.
        if  $b_M + s(it_i) > 1$ then
                $it_i$ does not fit in the $I_M$ bin.
                $B_{count} = B_{count} + 1$;
                $b_M = 0$;
        end if
    Pack $it_i$ item in the opened $I_M$ bin , $b_M = b_M + s(it_i)$
Else if $\exists k \frac{1}{2^{k-1}} < it_i \leq \frac{1}{2^k}$, $1 \leq k \leq M$ then $it_i$ is a $I_k$ piece
        if  $b_k + s(it_i) > 1$ then
                $it_i$ does not fit in the $I_k$ bin.
                $B_{count} = B_{count} + 1$;
                $b_k = 0$;
        end if
      Pack $it_i$ item in the opened $I_k$ bin , $b_k = b_k + s(it_i)$
    End if
    End for
    for $k$ = 1 to $M$ do
        if $b_k$= 0 then
            $B_{count} = B_{count} - 1$;
        End if
    End for
    Output $B_{count}$
End
```

2.3 Summary and Results of standard bin packing:

We know that bin packing is one of the classic and well-studied problems in the field of computer science. Since bin packing belongs to the class of Np hard problems, it is really difficult to come up with a polynomial time algorithm which solves the problem to give an optimal solution. So as a result, approximation algorithms are presented to find the closest possible solution to the optimal. One of the most basic and a simple online algorithm is next fit with a competitive ratio of 2. The proof for this ratio is simple and is given in the above sections. Now to study the problem we make use of competitive ratio which can also be termed as performance ratio, approximation ratio or worst case ratio. The competitive ratio is given different names in textbooks and papers (like performance ratio, worst case ratio, approximation ratio). But general practice followed to avoid confusion is that competitive ratio is used to analyze online algorithms and worst-case ratio, approximation ratio is used for offline algorithms. This is only a general practice implemented by some of the researchers and publishers to avoid confusion. In this section we make use of two different competitive ratios to study about the results or the bounds. The two types of competitive ratios are absolute competitive ratio and asymptotic competitive ratio. The asymptotic competitive ratio is used to represent the asymptotic cases and is defined in the 2D bin packing section. Now let's define the absolute competitive ratio. For a given instance I, let $m_A(I)$ be the number of bins used by the online algorithm A ($m_A(I)$ can also be termed as the cost of the algorithm A) and let $m_{OPT}(I)$ be the number of bins used by the optimal solution then the absolute competitive ratio R_A for the online algorithm can be given as $R_A = sup_I\{\frac{m_A(I)}{m_{OPT}(I)}\}$.

It was Johnson who extensively studied, analyzed this problem and presented his PhD dissertation on this problem in 1973.He showed that next fit has a competitive ratio of 2 and he [9] also showed that first fit has a asymptotic competitive ratio (performance) ratio of $\frac{17}{10}$.The proof for the ratio is complicated when compared to the proof of next fit algorithm (performance ratio =2) and it is explained in the above chapters. After this result of first fit, a question was raised by Johnson, if there exists a polynomial time online algorithm better than the first fit (i.e. performance ratio < $\frac{17}{10}$). This was resolved by Yao [30] who presented *refined first fit* with an asymptotic performance ratio of $\frac{5}{3} =$

1.66. In the same paper Yao also showed that unless P=Np, it is computationally intractable to come up with an online algorithm whose ratio $< \frac{3}{2}$. Lee and Lee [6] presented a *refined harmonic* algorithm which had a better ratio of 1.63597. This harmonic algorithm was further improved by Ramanan, Brown, Lee and Lee [31] in 1989 developed *modified harmonic* and *modified harmonic 2* whose asymptotic performance ratios were 1.61562 and 1.61217. After this result, Seiden [22] in 2002 developed an algorithm called *super harmonic* algorithm achieving an asymptotic performance ratio of 1.58889 which is by far the best known upper bound in online bin packing algorithms. This means that for online bin packing, there is no other algorithm itdeveloped whose asymptotic competitive ratio < 1.5889.

Coming to results based on absolute competitive ratio, Simchi-Levi [35] showed that first fit and best fit has an absolute competitive ratio no more than 1.75 and first fit decreasing and best fit decreasing has an absolute competitive ratio of 1.5.These algorithms takes $O(n\log n)$ time. G.Zhang [37] came up with a constant space online algorithm which runs in linear time. Furthermore, he [37] also proved that the algorithm returns a result whose absolute competitive ratio is 1.75. He also presented a constant space offline algorithm which runs in linear time and showed that the absolute approximation ratio is 1.5.Now exploring the lower bounds, Zhang[11] gave a lower bound of $\frac{5}{3}$.This result was given by Zhang which is mentioned in the paper by Epstein[11]. This result in Epstein`s paper is referenced from Zhang through private communication. So the $\frac{5}{3}$ is the best known lower bound result for the absolute competitive ratios of online bin packing problem.

Now let us discuss about the lower bound results for the asymptotic competitive ratio of the online bin packing problem. As we all know Yao [30] also showed that unless P=Np, it is computationally intractable to come up with an online algorithm whose performance ratio $< \frac{3}{2}$. This statement shows that 1.5 is the lower bound for the bin packing problem. This bound of 1.5 was further improved to 1.536 by Liang [33] and Brown [32] individually. This bound was further improved to 1.54014 by Van Vliet [12] in 1992. This 1.54 bound is by far the best known lower bound of asymptotic approximation ratio for the online bin packing problem.

Now coming to the results of the offline version of bin packing, D.S.Johnson [1], in this doctoral thesis, showed $m_{FFD}(x) \leq \frac{11}{9} m^*(x) + 4$, he proved that the performance ratio (approximation ratio) for *FFD* cannot get better than $\frac{11}{9}$. This results seemed to work considerably well for higher values of $m_{OPT}(x)$ ($m_{OPT}(x) > 10$). After this result, a lot of work and research was done to find the closest asymptotic additive constant (like 4) which is required to find better approximations for smaller instances. In this process, after the D.S.Johnson, B.S.Baker[2] proved that additive constant can be reduced to 3. Later in 1991,Yue Minyi [3] further reduced additive constant to1 i.e. $m_{FFD}(x) \leq \frac{11}{9} m^*(x) + 1$. Later in 1997, L. Rongheng, M. Yue [4] furthur tried to reduce the additive constant to $\frac{7}{9}$ but they did not prove the statement but gave a draft about it. They also conjectured that the tight additive constant can be $\frac{5}{9}$ (which proves to be an incorrect result). Finally in November 2011 Gyorgy Dósa, Rongheng Li, Xin Han and Zsolt Tuza [5] claimed that the lower bound for the additive constant is $\frac{6}{9}$. This bound is the most recent and the best known result for the first fit decreasing algorithm.

We also know that the offline bin packing problem admits asymptotic polynomial time approximation scheme (PTAS$^{\infty}$). This algorithm was given by Fernandez la Vega and Lueker [19] whose asymptotic approximation ratio is 1+ ε where $\varepsilon > 0$.This can be elaborated as for a given instance I and for any $\varepsilon>0$, ,we have an asymptotic PTAS (PTAS$^{\infty}$) that runs in polynomial time and returns a result of at most $(1+\varepsilon)\ m_{OPT}(I)+ 1$ bins. If the items are sorted, the running time of this algorithm is $O(n) + f(\frac{1}{\varepsilon})$. Kamarkar and Karp [34] presented asymptotic fully polynomial time approximation scheme (AFPTAS) and showed that bin packing problem has an AFPTAS. So for a given instance I and ε, the ratio can be given as $(1+ \varepsilon)\ m_{OPT}(I)+ O(\frac{1}{\varepsilon^2})$. The running time of this algorithm polynomially depends on $\frac{1}{\varepsilon}$. Since asymptotic FPTAS returns a better result than PTAS$^{\infty}$, it is considered to be the best known upper bound of asymptotic approximation ratio for offline bin packing problem. Since we have an algorithm which returns an asymptotic approximation ratio of $(1+ \varepsilon)$, $\varepsilon>0$, it is given that the lower bound for the asymptotic approximation ratio of offline bin packing is 1.

Now let us deal with the results based on absolute competitive (approximation) ratio for offline bin packing problem. Simchi-Levi [34] proved that first fit decreasing and best fit decreasing has an absolute competitive ratio of $\frac{3}{2}$. This result is the best known upper bound of absolute competitive (approximation) ratio for offline bin packing problem. As for the lower bound of the absolute approximation ratio, Garey and Johnson [35] gave a bound of $\frac{3}{2}$ which is considered to be the best known bound.

	Asymptotic Competitive(approximation) Ratio	Absolute Competitive(approximation) ratio
Online bin packing	1.540[12]	$\frac{5}{3}$ [11]
Offline bin packing	1	$\frac{3}{2}$ [35]

Table 2: Best known Lower bounds for standard bin packing problem.

	Asymptotic Competitive(approximation) Ratio	Absolute Competitive(approximation) ratio
Online bin packing	1.589 [16]	$\frac{7}{4}$ [36]
Offline bin packing	$FPTAS$[33]	$\frac{3}{2}$ [34]

Table 3: Best known Upper bounds for standard bin packing problem.

CHAPTER 3

VARIANTS OF BIN PACKING PROBLEM

3.1Bin packing with Rejection:

3.1.1 Introduction to the Problem:

Bin packing with rejection is considered to be a special case of classical bin packing. The bin packing problem with rejection was presented and studied by Dósa and Y.He [10]. In bin packing problem, we have the input sequence of items whose size is in the range (0, 1] and these items needs to be filled in unit sized bins. We have continuous supply of unit sized bins and the sum of the items packed in a particular bin should not exceed its bin capacity. Our goal is to minimize the number of bins used and each item is packed in one bin. So when the item is ready to be packed in the bin, there is a possibility that the item might be refused or get rejected to be packed in that bin. This is where we need to consider bin packing with rejection.

So in a real time scenario or in many applications there are situations where the items are refused or rejected to be packed in a bin. When such items are refused or rejected we have a cost associated with the item termed as ‘rejection cost’. We need to understand that the rejection cost is associated with an item but not with the bins. To understand the concept of rejection cost let us consider the following examples. Let us consider an application where bins are disks and items are the files which needs to be saved on the disks. Now if a file is rejected to be saved on the disk, its rejection cost would be the cost of transferring it and saving the file on the alternative media. Similarly in another application where bins are storage spaces, rejection cost is paid to the disappointed customer whose items cannot be stored. Thus we came to know that in bin packing with rejection each item is associated with a rejection cost.

For a given input instance of items I, each item $it_i \in I$ contains size and rejection cost which is denoted by $s(it_i)$ and $r(it_i)$ respectively. The bin packing with rejection has online and offline versions of bin packing. In an online bin packing with rejection each item it_i belonging to the instance I(containing n items) is represented as $(s(it_1), r(it_1))$,$(s(it_2), r(it_2))$,… $(s(it_n), r(it_n))$. The items are arrived one after the other. Upon arrival they must be either assigned or rejected. Once an action is made it cannot be revoked.

In the bin packing problem, our goal is to minimize the number of bins used for packing but in bin packing with rejection our goal is to minimize the sum of the following two entities.

i. Sum of rejection cost of the rejected items.
ii. Number of bins used for packing the items.

So our goal is to minimize this sum. The rejection costs are larger than 1.

The offline version of the problem is dealt in a different way which is related to caching. Dósa and Y.He[10] suggested an application for the offline version of the problem where items are files which needs to be used on the local system. A file is used exactly once at a later time. One way to deal with this is to download the file to the local system and save it on the local server. So when the file is needed, the time taken to retrieve the file is quick but it occupies space in the local server. In this option the incurred cost is the cost of the local servers. The other option would be downloading the file directly from the external server when there is a requirement but the retrieval time is more when compared to the previous option. In the second option the rejection cost is associated with the cost of transferring the file from the external server. An algorithm is needed to generate minimum cost results or outputs using the two options available.

3.1.2 Results:

Dósa and Y.He studied four variants of bin packing with rejection in their paper titled “bin packing problems with rejection penalties and their dual problems” [10]. These variants are offline and online bin packing with respect to the absolute and the asymptotic measures. For offline version of bin packing with rejection Dósa and Y.He present an algorithm with absolute approximation ratio of 2 and asymptotic approximation ratio of $\frac{3}{2}$. Furthermore, it is stated that unless P=N_p, we cannot have an algorithm with absolute approximation ratio less than $\frac{3}{2}$.

The absolute approximation ratio and asymptotic approximation ratio was further improved from 2 and $\frac{3}{2}$ to $\frac{3}{2}$ and $(1+\varepsilon)$ approximation by Leah Epstein [11] using the previous results from [10].

Now coming to online version of bin packing with rejection Dósa and Y.He present an algorithm with absolute competitive ratio of 2.618 while the lower bound is 2.343.They present an algorithm with asymptotic competitive ratio of $(1.75+ \varepsilon)$ while the lower bound which is 1.5401 was due to Van Vliet [12]. We get the best asymptotic competitive ratio for the bounded space algorithms (where only a constant number of bins are open bins) and it is shown that the ratio is the same for standard bin packing problem in [11]. For instance Epstein adapted Harmonic algorithm of Lee and Lee [6] and shown in [11] that they have the same asymptotic competitive ratio of 1.69103 as standard bin packing. Epstein came up with an improved unbounded space algorithm which is a modification of modified harmonic algorithm gives asymptotic competitive ratio of approximately 1.61562.

3.2 2D Bin Packing

3.2.1 Introduction to the Problem

The classical bin packing problem has been a one of the oldest and well-studied problems in field of computer science. In this section we will be discussing about the two dimensional bin packing problem and its results. It is observed that the 2D bin packing problem is a generalization of the classic bin packing problem. In 2D bin packing, each item is associated with two parameters width and height. So, in this problem each item it_i is a rectangle of width $w_i \leq 1$ and height $h_i \leq 1$ where $1 \leq i \leq n$, here n denotes the number of items in the instance.

For a given instance I containing n items, each item it_i is a rectangle of width $w_i \leq 1$ and height $h_i \leq 1$ needs to be packed into unit sized square bins, our goal is minimize the number of bins used for packing. In this problem, the items should be packed in the bins with no overlapping. The items are packed such that its sides are parallel to the edges of the bin and rotation of the items is not allowed. So in this way we can explain 2D bin packing problem. In short, 2D bin packing can be explained as a procedure where the given sets of 2D rectangles (items) are packed into unit square bins such that the number of packed bins is minimal. Since 2D bin packing problem is a generalization of 1D bin packing problem, it is considered to be an Np-hard problem too.

The potential use of 2D bin packing in many real time and industrial applications is a motivating factor in studying this problem. This 2D bin packing is used in applications like packing items in warehouses and trucks, cutting stock problems (cutting rectangles from sheets of a given size) and many more. In the cutting stock problems, we have glass/metal rectangular sheets of fixed or standard size. But the requirement from the customer could be rectangular sheets with arbitrary sizes which are less than the original standard sheet, now the sheets need to be cut in such a way that the numbers of standard sheets used is minimum. Especially for these kinds of applications, it is necessary and important to study the online version of 2D bin packing and its algorithms.

So in the online bin packing, the items are packed as per the input sequence and each item is assigned to the bin without the knowledge about the remaining items. To study and evaluate the performance of the online bin packing, the commonly used ratio is asymptotic competitive ratio. This ratio helps us in assessing the performance of the algorithm so let us understand the ratio and its importance. For a given instance I of online bin packing, let be the number of bins used by the online algorithm A ($m_A(I)$ can also be termed as the cost of the algorithm A) and let $m_{OPT}(I)$ be the number of bins used by the optimal solution then the asymptotic competitive ratio for the online algorithm can be given as

$$R_A^\infty = \lim_{m\to\infty} \sup_I \{\frac{m_A(I)}{m_{OPT}(I)} \mid m_{OPT}(I) = m\}.$$

After defining the asymptotic competitive ratio, we need to know about the variants in the online bin packing problem. In [14], the online bin packing problem is said to have two types of models based on the space constraint. They are

1. Bounded space model.
2. Unbounded space model.

Now that we came to know that there are two variants in the online bin packing, it is necessary that we need to know the difference between the two.

Unbounded Space model:

In the unbounded space model, there is no limit on the number of active or open bins available for packing items whereas in the bounded space model, we have a constant number of bins which are opened at any point of time available for packing items.

Bounded Space Model:

So in the bounded space model, if an item is ready to be packed and if none of the active or opened bins have enough space to that item then one of the bins is closed and a new bin is opened. So in this way in the space bounded model, the number of active or opened bins is remained constant. This model is practical and seems more realistic where it can be implemented in many real time applications.

Now in 1-space bounded multi-dimensional bin packing problem, there is only one active or opened bin at any given time and since it's a multidimensional bin packing problem we deal with *d* dimensional hyperbox(items) which needs to be filled in *d* dimensional hypercube (bins) with unit size where $d \geq 2$. We know that any two dimensions *i* and *j* define a plane P_{ij}. 90° rotation of the item in any plane P_{ij} is allowed, otherwise, the competitive ratio is unbounded. This result is taken from [15]. Keeping all these constraints in mind we pack the *d* dimensional hyberboxes in the *d* dimensional unit sized cubes trying to minimize the number of bins used. So if the item cannot be packed in the active bin then the active bin is closed and a new bin is opened. Once a bin is closed, it cannot be opened again.

Now let us explain this using an example. In the given figure below, for an instance I we have 3 items it_1, it_2, it_3 which arrive to be packed in the given order. Since we are dealing with 1-space 2D online bin packing problem, we need to keep in mind that only one bin is allowed to be in active state (opened bin) at any given time. Now after packing it_1, we need to pack it_2 which can be packed in two different ways. One way is to pack the item directly with no rotation and the other way is to pack the item after 90° rotation. In the non-optimal configuration the second item is packed with no rotation and then when the third item arrives, there is no enough space to accommodate the third item, so a new bin is opened. But in the optimal configuration the second is rotated 90° and then packed. As a result there is enough space for the third item which is also packed in the same bin. So the below figure explains the packing procedure in 1 space 2D online bin packing problem.

It was Zhang,Y.L. Chin, Hing-Fung Ting, Xin Han and Zhuo Chang [14] who studied this variant of the bin packing problem closely and gave an online algorithm with a competitive ratio 4^d which was the first study on 1-space bounded d dimensional bin packing problem.

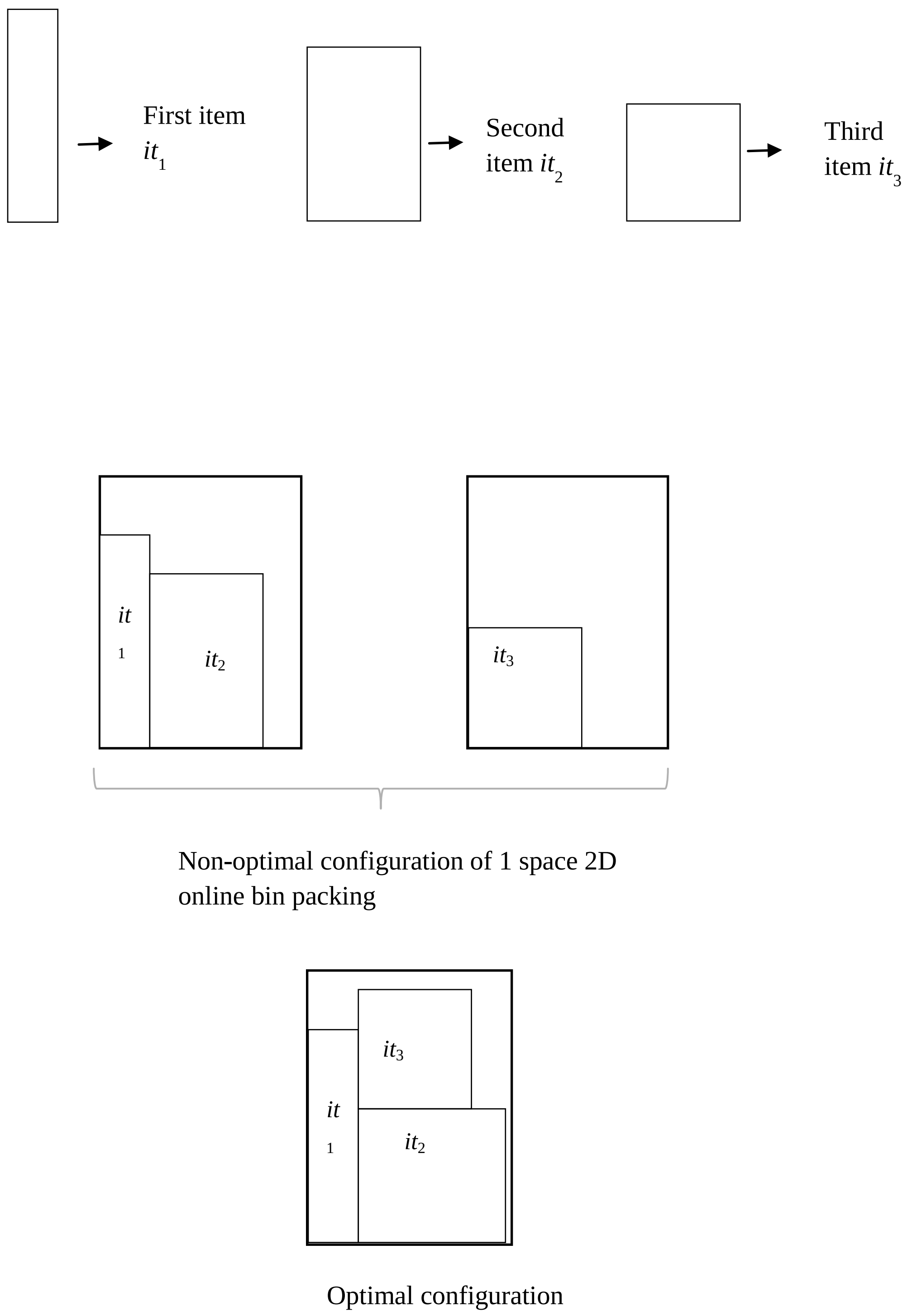


Fig 6: Example for optimal and Non-optimal configuration of 1 space 2D online bin packing

3.2.2 Results:

Firstly discussing about the results of 2D Online bin packing problem, lets present the lower bounds of the problem. It was Galambos [18] who provided a lower bound of 1.6 for the 2D online bin packing problem. This lower bound was improved to 1.808 by Galambos & Van Vliet [21] and to 1.857 by Van Vliet[22] and the finally to 1.907 by Blitz [23] in the year 1996.

Now coming to the Asymptotic competitive ratio which was defined in the above section, Coppersmith and Raghavan [20] came up with the first online algorithm with asymptotic competitive ratio of 3.25. Csirik[25] in 1993 improved this ratio to 3.0625. Csirik and Van Vliet[24] presented an algorithm for all *d* dimensions and particularly for 2D online bin packing problem they gave an asymptotic competitive ratio of 2.8596. Relying on the techniques of the improved harmonic algorithm, Han [26] in 2001 improved the ratio to 2.7834. In 2003, Seiden and Van Stee[17] further improved this bound to 2.66013. They presented an algorithm regarded as $H \otimes C$ where H is the harmonic algorithm [6] and C is considered to be an instance of the improved harmonic algorithm. After Seiden and Van Stee`s 2.66013 bound, the improved bound for this problem became an open question, a lot of work was going on to improve the bound. One deliberate idea to improve the bound was to use an instance of super harmonic algorithm instead of improved harmonic algorithm which was used by Seiden and Van Stee. Nevertheless, Seiden and Van Stee[17] also stated that the previous analysis framework doesn`t work to improve the bound further. Finally in 2011, Han, Francis, Zhang and Yong [13] presented an improved and a better result. They gave a bound of 2.5545 for the 2D online bin packing problem which remains to be the most recent result in this domain.

Since it was known that the previous analysis framework cannot be extended to the super harmonic algorithm, they came up with a new analysis framework which is useful for analyzing online 2D and multidimensional bin packing problems. They also gave a new weighting function which was considered to be much simpler than ones given in Seiden[16] paper. So the new weighting functions in combination with the new framework helped them in designing the algorithm H $\otimes$ SH+ where H is the harmonic algorithm and SH+ is the super harmonic algorithm which gave an upper bound of 2.5545.

Now we discuss about the offline version of the 2D bin packing problem. It was Chung [27] in 1982, who gave an approximation algorithm with an asymptotic performance ratio of 2.125. This bound was improved to 1.69103 by Caprara [28] in 2002. Finally in 2009 Bansal [29] further improved this bound by presenting a randomized algorithm with asymptotic performance ratio of at most 1.525.He also showed that the two-dimensional bin packing problem does not admit an asymptotic polynomial-time approximation scheme.

3.3 2D Strip packing problem and its results:

In strip packing problem a given set of rectangular input items with width and height bounded by 1 is packed into a vertical strip of fixed width 1 and infinite height. The goal is to minimize the height of the strip which packs the given input of rectangles. While packing no two rectangles should overlap with each other and the sides of the rectangles are parallel to the strip sides. Rotations are not allowed. Several industrial applications and real life applications like cutting and packing use variants or extensions of this problem and this motivates us to study many 2D bin packing problems(variants of the problem).

We have online and offline version to this problem. If we know the all the rectangles before we pack the items then it is regarded as an offline version whereas in the online version the packing is done as per the input sequence and the packing decision is done before the next rectangle arrives. Once a rectangle is packed it cannot be moved. Strip packing is Np-hard and the lower bound of 1.5401[12] is valid for online strip packing.

Results:

Let us now discuss the results of the strip packing problem, for the offline version of the strip packing Coffman [44] presented algorithms next fit decreasing height (NFDH) and first fit decreasing (FFDH) height which returned asymptotic approximation ratios of 2 and 1.7 respectively. Golan [45] improved this ratio to $\frac{4}{3}$. This result was further improved when Baker [46] came up with an asymptotic approximation ratio of $\frac{5}{4}$.Another important result for offline strip packing is asymptotic fully polynomial time approximation scheme (AFPTAS) by Kenyon and Remila [47]. After this AFPTAS from Kenyon and Remila, in 2005 Jansen and Stee [48] presented an AFPTAS which included the case where rotations of 90° are allowed. They developed this algorithm using linear programming and random techniques. The additive constant of this result was improved from $O(\frac{1}{\varepsilon^2})$ to 1 by Jansen & Solis-Oba [49] for the cost of a higher running time.

Now coming to the absolute approximation ratio Schiermeyer [50] in 1994 and Steinberg [51] in 1997 presented algorithms which returned an absolute approximation ratio of 2. This remained to be the best upper bound of absolute approximation ratio for more than a decade. This upper bound of 2 established for more than a decade was broken when Rolf Harren and Rob van Stee[52] presented an algorithm which returned an absolute approximation ratio of 1.9396 .

Now coming to the online version of strip packing, Baker and Schwarz [53] introduced an online strip packing algorithm called shelf algorithm. In this algorithm items are packed left to right in rectangular strips or rows forming levels called shelves. The first shelf is placed at the bottom of the bin/strip and the consequent shelves are produced by a horizontal line passing through the top of the tallest item in the shelf below. This kind of packing items in rectangular strip is shelf packing. The shelf packing introduced by Baker and Schwarz [53] was an elegant idea to implement standard bin packing algorithms to online strip packing. In this way next fit and first fit algorithms were employed to obtain asymptotic competitive ratios of 2 and 1.7 respectively. Similarly this idea was extended to harmonic shelf algorithm by Csirik and Woeginger [54] to obtain an asymptotic competitive ratio of 1.6910. In 2007 Han [55] further improved this bound to 1.5888. He formulated a relation between strip packing and one dimensional algorithm and thus showed online strip packing admits an algorithm with asymptotic competitive ratio of 1.5888. A lower bound of 2 for the absolute competitive ratio of online strip packing was given by Brown [56] in 1982.

Chapter 4

CONCLUSION AND FUTURE WORK

4.1 Recent Papers or Developments

It is noted that bin packing problems is one of the classic and challenging problems in the field of computer science. This problem has been studied over 30 years and to this date work has been going on to find new improvised approaches and better results. However it seems most of the easy results has been attained and after analyzing the proofs of these results it's been realized that getting these results was not easy. However bin packing problem has been fruitful in developing methods and served as a proving ground for techniques for approximation schemes and has helped in developing methods for other problems like Scheduling, Resource allocation and many more. Bin packing to this date has ever new applications and especially the variants of bin packing are important to information technology. This problem has many potential applications in different real world industries (like transportation, logistics, Information Technology etc.). Some of the applications are scheduling television programming, cutting stock problem, cloud computing, truck loading problem and many more. As we know there are different variants of the problem and similarly several approaches for tackling them, a lot of papers have been published relating to this problem.

In this thesis, we list a few papers published recently which marks the most updated work going on in this field. Some of the papers are solving the two-dimensional bin-packing problem with variable bin sizes by greedy randomized adaptive search procedures and variable neighborhood search by Andreas M. Chwatal and Sandro Pirkwieser [37] in 2011. In 2011, Friedrich Eisenbrand, Domotor Palvolgyi and Thomas

Rothvo[38] presented a paper called Bin Packing via Discrepancy of Permutations. This paper was recently revised in February 2012. In 2012, Filipe Brandao and Joao Pedro Pedroso [39] presented a paper solving bin packing related problems using an arc flow formulation. Abdesslem Layeb and Sara Chenche [40] came up with a paper titled a novel GRASP Algorithm for Solving the Bin Packing Problem which was published on April 2012.In 2012, Guido Perboli, Roberto Tadei, Mauro M. Baldi [41] published a paper the stochastic generalized bin packing problem and again in the same year along with Crainic T.G they presented another paper branch-and-price and beam search algorithms for the generalized bin packing problem[42]. In February 2012, Gyorgy Dosa and Leah Epstein [43] presented a paper called generalized selfish bin packing. The information of these recent papers indicate the amount of work and research going into this field and it emphasis the importance of bin packing problem.

To share the recent progress in this field of bin packing, a fourth international workshop on bin packing and placement constraints BPPC'12 is being held on May 29th 2012 at Nantes, France. This workshop is associated to the Ninth International Conference on the Integration of Artificial Intelligence and Operations Research techniques in Constraint Programming CPAIOR 2012.

BIBLIOGRAPHY:

1. Johnson, D.S.: Near-optimal bin-packing algorithms. Doctoral Thesis. MIT Press, Cambridge (1973)
2. Baker, B.S.: A new proof for the first-fit decreasing bin-packing algorithm. J. Algorithms,49-70(1985)
3. Yue, M.: A simple proof of the inequality $FFD(L) \leq 11/9 OPT(L) + 1, \forall L$, for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica 7(4), 321–331(1991)
4. Li, R., Yue, M.: The proof of $FFD(L) \leq 11/9 OPT(L) + 7/9$. Chinese Science Bulletin 42(15) (August 1997)
5. Gyorgy Dósa , Rongheng Li, Xin Han, Zsolt Tuza : Tight absolute bound for First Fit Decreasing bin-packing: $FFD(I) \leq 11/9\ OPT(I) + 6/9$.
6. C. Lee and D. Lee. A simple on-line bin-packing algorithm. *Journal of ACM*, 32:562572,1985.
7. Johnson,D.S., and Garey, M.R.(1985), A 71/60 theorem for bin packing, J.Complexity 1, 64-106.
8. Doina Bein, Wolfgang Bein, and Swathi Venigella. Cloud Storage and Online Bin Packing. http://www.egr.unlv.edu/~bein/pubs/bein_cloud_19.pdf.
9. D. S. Johnson, A.Demers, J. D. Ullman, M. R. Garey, and R. L. Graham(1974). Worst-case performance bounds for simple one-dimensional packing algorithms.SIAM J. Computing 3, 299-325.
10. G. Dósa and Y. He. Bin packing problems with rejection penalties and their dual problems. Information and Computation, 204(5):795_815, 2006.

11. Leah Epstein. Bin Packing with Rejection revisited. In Proc. of the 4th International Workshop on Approximation and Online Algorithms (WAOA 2006), pages 146–159, 2006.
12. A. Van Vliet. An improved lower bound for online bin packing algorithms. Information Processing Letters, 43(5):277_284, 1992.
13. Xin han, Guochuan Zhang, Yong Zhang, Francis y. L. Chin and Hing-Fung Ting. A new upper bound 2.5545 on 2d online bin packing. Journal ACM Transactions on Algorithms (TALG), Volume 7 Issue 4, September 2011.
14. Yong Zhang, Francis Y. L. Chin, Hing-Fung Ting, Xin Han, Zhuo Chang: Online Algorithm for 1-Space Bounded Multi-dimensional Bin Packing. FAW-AAIM '11,308-318.
15. Fujita, S.: On-Line Grid-Packing with a Single Active Grid. Information Processing Letters 85, 199–204 (2003).
16. Seiden, S. 2002. On the online bin packing. J. ACM 49, 640–671.
17. Seiden, S. and Van Stee, R. 2003. New bounds for multidimensional packing. Algorithmica 36, 261–293.
18. Galambos, G. 1991. A 1.6 lower-bound for the two- dimensional on-line rectangle bin-packing. Acta Cybern.
19. Fernandez de la Vega, W., and Lueker, G.S. (1981), Bin packing can be solved within 1+ε in linear time, Combinatorica 1, 349-355.
20. Coppersmith, D. and Raghavan, P. (1988), Multi-dimensional online bin packing : Algorithms and worst-case analysis.

21. Galambos, G. and Van Vliet, A. 1994. Lower bounds for 1-,2- and 3- dimensional on-line bin packing algorithms. Computing 52,3,281-297.

22. Van Vliet,A. 1995. Lower and upper bounds for online bin packing and scheduling heuristics. Ph.D. dissertation. Erasmus University.

23. Blitz, D., Van Vliet, A., and Woeginger, G. J. 1996. Lower bounds on the asymptotic worst-case ratio of online bin packing alorithms. Unpublished manuscript.

24. Csirik, J. and Van Vliet, A. 1993. An on-line algorithm for multidimensional bin packing. Oper. Res. Lett 13,149–158.

25. Csirik, J., Frenk, J. B. G., and Labbe, M. 1993. Two-dimensional rectangle packing: On-line methods and results. Disc. Appl. Math. 45, 3, 197–204.

26. Han, X., Fujita, S., and Guo, H. 2001. A two-dimensional harmonic algorithm with performance ratio 2.7834.IPSJ SIG Notes 93, 43–50.

27. Chung, F., Garey, M., and Johnson, D. 1982. On packing two-dimensional bins. SIAM J. Alg. Disc. Meth. 3, 1,66–76.

28. Caprara, A. 2002. Packing 2-dimensional bins in harmony. In Proceedings of the Symposium on Foundations of Computer Science (FOCS). 490–499.

29. Bansal, N., Caprara, A., and Sviridenko, M. 2009. A new approximation method for set covering problems with applications to multidimensional bin packing. *SIAM* J. Comput. 39, 4, 1256–1278.

30. RAMANAN, P., BROWN, D., LEE, C., AND LEE, D. 1989. On-line bin packing in linear time. J. Algor. 10, 305–326.

31. BROWN, D. J. A lower bound for on-line one-dimensional bin packing algorithms. Tech. Rep. No.R-864, Coordinated Sci. Lab., Univ. of Illinois, Urbana, Ill., 1979.

32. LIANG, F. M. A lower bound for on-line bin packing. Inf Proc. Lett. 10 (I 980), 76-79.

33. N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82, pages 312–320, 1982.

34. D. Simchi-Levi. New worst-case results for the bin-packing problem. Naval Res. Logist., 41(4):579–585, 1994.

35. M. R. Garey, D. S. Johnson, Computer and Intractability: A Guide to the theory of NP-Completeness, New York, Freeman, 1979.

36. G. C. Zhang, X. Q. Cai, C. K. Wong, Linear-time approximation algorithms for bin packing problem, Operations Research Letters, 26(1999) 217-222.

37. Andreas M. Chwatal, Sandro Pirkwieser: Solving the Two-Dimensional Bin-Packing Problem with Variable Bin Sizes by Greedy Randomized Adaptive Search Procedures and Variable Neighborhood Search. EUROCAST (1) 2011: 456-463.

38. F. Eisenbrand, D. Palvoelgyi and T. Rothvoss. Bin Packing via Discrepancy of Permutations. Symposium on Discrete Algorithms (SODA 2011), San Francisco, USA, January 22-25, 2011.

39. Filipe Brandao , Joao Pedro Pedroso. Solving Bin Packing and Related Problems Using an Arc Flow Formulation.

40. Abdesslem Layeb, Sara Chenche . A Novel GRASP Algorithm for Solving the Bin Packing Problem. International Journal of Information Engineering and Electronic Business (IJIEEB) Volume 4, Number 2, 8-14, April 2012.

41. G. Perboli, R. Tadei, and M. M. Baldi. "The stochastic generalized bin packing problem". Discrete Applied Mathematics, 2012.

42. Baldi M. M., Crainic T.G., Perboli G., Tadei R., Branch-and-price and beam search algorithms for the Generalized Bin Packing Problem, pp. 18, 2012.

43. Gyorgy Dosa, Leah Epstein. Generalized selfish bin packing, 2012.

44. Coffman EG, Garey MR, Johnson DS, Tarjan RE (1980) Performance bounds for level oriented two dimensional packing algorithms. SIAM J Comput 9:808–826.

45. Golan, Performance bounds for orthogonal, oriented two-dimensional packing algorithms, SIAM J. Comput. 10, 571-582, 1981.

46. B.S. Baker, D.J. Brown, and H.P. Katseff, A 5/4 algorithm for two-dimensional packing. J. Algorithms 2, 348-368, 1981.

47. C. Kenyon and E.R´emila, A near-optimal solution to a two-dimensional cutting stock problem, Mathematics of Operations Research 25, 645-656, 2000.

48. K. Jansen and R. van Stee. On strip packing with rotations. In STOC: Proc. 37th ACM Symposium on Theory of Computing, pages 755-761, 2005.

49. K. Jansen and R. Solis-Oba. New approximability results for 2-dimensional packing problems. In MFCS: Proc. 32nd International Symposium on Mathematical Foundations of Computer Science, pages 103-114, 2007.

50. Schiermeyer I (1994) Reverse-fit: a 2-optimal algorithm for packing rectangles. In:Proceedings of the second annual European symposium on algorithms (ESA), pp 290–299.

51. Steinberg A (1997) A strip-packing algorithm with absolute performance bound 2. SIAM J Comput 26(2):401–409.

52. Rolf Harren and Rob van Stee (2009):Improved Absolute Approximation Ratios for Two-Dimensional Packing Problems. Proceedings of the 12th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, LNCS 5687, pp. 177-189.

53. B.S. Baker and J.S. Schwarz, Shelf algorithms for two-dimensional packing problems,SIAM J. Comput. 12, 508-525, 1983.

54. J. Csirik and G.J. Woeginger, Shelf algorithm for on-line strip packing, Information Processing Letters 63, 171-175, 1997.

55. Ye D, Zhang G (2007) On-line scheduling of parallel jobs in a list. J Sched 10:407–413.

56. Brown DJ, Baker BS, Katseff HP (1982) Lower bounds for on-line two-dimensional packing algorithms. Acta Inform 18:207–225.

VITA

Graduate College
University of Nevada, Las Vegas

Yoga Jaideep Darapuneni

Home Address :

1555 E Rochelle Ave, #Apt 252,
Las Vegas, Nevada - 89119

Degrees:

Bachelor of Science in Computer Science, 2006

Jawaharlal Nehru University, Hyderabad

Thesis Title: A Survey of Recent and Classical Results in Bin Packing Problem.

Thesis Examination Committee:

Chair Person, Dr. Wolfgang Bein, Ph.D.

Committee Member, Dr. Ajoy K Datta, Ph.D.

Committee Member, Dr. Ju-Yeon Jo, Ph.D.

Committee Member, Dr. John Wang, Ph.D.