Study of Bin Packing algorithms

Nitish Rawat
M110248CS
Guided by: Dr. Priya Chandran

December 6, 2013

Keywords: Task scheduling, Bin packing, Linear program, Resource pooling

Abstract

Task scheduling is a computational problem and a very challenging issue in multi-processing. Task scheduling can be abstracted as the famous "Bin Packing" problem which is **NP**-Hard. Several approximation solutions for bin packing exist in literature, some of which are based on quadratic programming and randomized rounding of fractional solutions.

In this work, those algorithms are studied. Also to gain more insight into implementation issue. It was attempted to implement linear programming using R.

Resource scheduling in computer cluster known as resource pooling is solved through bin packing. Resource pooling is the primary motivation of the work.

1 Introduction

Most natural optimization problems, including those arising in important application areas are **NP**-hard. "**NP**-hard optimization problems exhibit a rich set of possibilities, all the way from allowing approximality to any required degree, to essentially not allowing approximality at all" [1]. An algorithm that returns near-optimal solutions is called an approximation algorithm. Task scheduling is reduced to Bin packing. Bin Packing is a mathematical problem. Given a partition problem instance with n items having sizes a_1,a_2,...,a_n , we make an instance of the bin-packing problem with the items of same size and bins of capacity $\sum_i a_i/2$. Thus, a solution for the bin-packing instance that uses exactly 2 bins gives a solution to the partition problem (the partition is the items in the same bin). So Bin-packing problem can be reduced to the partition problem (**NP**-complete) problem making it **NP**-hard [1].

Bin packing are very well studied problems in combinatorial optimization. In Bin packing, a list L of real numbers between 0 and 1, 1 inclusive, is given, which is to be assigned to unit capacity bins in such a way that no bin receives numbers totaling more than 1 and a minimum number of bins is used [2]. So Bin packing is an optimization problem. There has been effort of trying to implement Bin packing optimization problem using linear program. Linear program is a method for optimization in finding maximum/minimum value of linear function of variables satisfying lesser number of constraints. Constraints are limiting value ranges of variables.

Let the minimum bins be L^*. For any (heuristic) bin-packing algorithm S, let $S(L)$ denote the number of bins used for the input list L and $R_s(k)$ the maximum ratio $S(L)/L^*$ for any list with $L^* = k$. The asymptotic performance ratio of S, denoted by R_s^∞, is defined as $\overline{\lim}_{k\to\infty} R_s(k)$. The problem of finding an optimal packing can involve the solution of the **NP**-complete partition problem and hence is likely to be computationally intractable. For a long time the best polynomial approximation algorithm was the so-called First-Fit-Decreasing algorithm.

In 2012, there has been attempt of formulating bin packing using linear program. Bin packing has been found to be infeasible. We try to find why it fails?

Besides having theoretical importance the problem has applications in partitioning problem, load balancing, cutting stock, resource allocation and many more.

2 Problem Statement

Study of approximation algorithm for bin packing and the implementation of the simplex algorithm for linear program in R.

3 Motivation

In computer cluster there is scenario when multiple resources are allocated in computer cluster. Examples of resources include storage, processing, memory, network bandwidth, and virtual machines. These resources may be shared or pooled from computer clusters.

Resource pooling is the primary motivation of this work. Resource pooling in cluster of different functional units are required for huge cost savings.

There can be a high degree of re-usability and recyclability for low-performing processors with high speed communication buses. This can be implemented by such scheduler. Thus a lot of cost and wastage of resources can be managed efficiently.

4 Literature Survey

There are many types of task-to-processor assignment (Bin Packing) problems found in literature. Some of the Bin-Packing problems are mentioned below:

We call an algorithm on-line if the numbers in list L are available one at a time, and the algorithm has to assign each number before the next one becomes available. We call an algorithm off-line if all the numbers of list L can appear together.

4.1 Problem definition: Bin Packing Problem [2]

A finite list $L = \langle a_1, a_2, \ldots, a_n \rangle$ of real numbers in the range (0,1], and a sequence of unit-capacity bins, $BIN_1, BIN_2 \ldots$ extending from left to right are given. The problem is to find an assignment or packing of the numbers into the bins so that no bin has contents totaling more than one, and yet the number of bins used, i.e., nonempty, is minimized. For a given list L this minimum is denoted by L*.

4.2 Vector Bin Packing Problem (VBP)[3]

Given a set of n rational vectors $p_1, \ldots, p_n$ from $[0, 1]^d$, find a partition of the set into sets $A_1, \ldots, A_m$ such that the sums of each dimension of all vectors in the same partition is ≤ 1. The objective is to minimize m, the size of the partition.

4.3 Existing Algorithms in Bin packing

Some of the various algorithms proposed in the literature are found upon. It is predicted that the quality of solution obtained will be much better than using a single fixed algorithm. Implicitly, the success of this idea depends on the hypothesis that different algorithms "favor" different regions in the input space. A comprehensive set of inputs for different algorithms are being studied in [4] by Chetan Rao in recent past.

4.3.1 Next-Fit [4]

While there are items remaining:

Step 1. Check the open bins if the item can be placed. If yes, place it and proceed to the next item.

Step 2. Else, Close the oldest bin, open a new one and go back to step 1.

This is an on-line algorithm.

4.3.2 First-Fit

In this algorithm the change is that the bin is never closed. The bins are open and checked if item can be placed in it.

While there are items remaining:

Step 1. Check the open bins if the item can be placed. If yes, place it and proceed to the next item.

Step 2. Else, open a new bin and go back to step 1.

It is to be noted that increasing items may induce worst-case performance. This is an on-line algorithm. First-Fit algorithm has an asymptotic performance ratio of 17/10.

4.3.3 First-Fit Decreasing

As noted the worst case performance instance of First-Fit. The algorithm First-Fit decreasing avoids this and improves the performance.

Step 1. Sort the items in decreasing order.

Step 2. Check the open bins if items can be placed. If yes, place it and proceed to the next item.

Step 3. Else, Open a new bin and go back to step 2.

First-Fit decreasing algorithm has an asymptotic performance ratio of 11/9 [5] (i.e. 1.222). This is off-line algorithm.

4.4 Asymptotic Polynomial Time Approximation Scheme

Fernandez and Leukar [2] gave the first asymptotic polynomial-time approximation algorithm. They said Bin Packing can be solved within a factor of (1 + ϵ) in linear time by putting forward an elimination of small pieces and rounding technique that allowed them to reduce the problem of packing large items to finding an optimum packing of just a constant number of items. ϵ is a small number.

4.5 Linear Program[3]

Bin Packing can be formulated as linear program. Linear programs are optimization method which finds maximum value of linear function subject to less number of contraints than number of variables.

Let the pieces be numbered $p_1, ..., p_n$. The formulation has 0-1 variables x_{ij}. The **binary variable** x_{ij} indicates if piece p_i is assigned to BIN_j and the binary variable y_j indicates whether BIN_j is in use or not. Our objective is to minimize the number of bins used. The linear program is formulated as follows -

$$\text{minimize} : \sum_{j=1}^{m} y_j \tag{1}$$

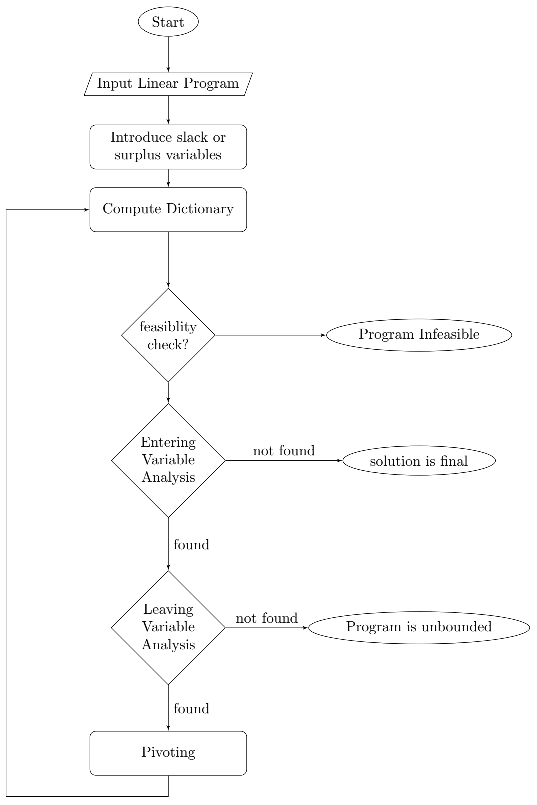
such that

$$\begin{aligned}
\sum_{j=1}^{m} x_{ij} &= 1 && 1 \leq i \leq n && (2)\\
\sum_{i=1}^{n} p_i.x_{ij} &\leq 1 && 1 \leq j \leq m && (3)\\
y_i &\geq x_{ij} && 1 \leq i \leq n, 1 \leq j \leq m && (4)\\
x_{ij} &\in \{0,1\} && 1 \leq i \leq n, 1 \leq j \leq m && (5)
\end{aligned}$$

The constraints of the problem are as follows -

- Constraint (2) states that every vector is packed in a bin.
- Constraint (3) ensures that the packed vectors do not exceed the bin capacity.
- Constraint (4) tells whether a bin is used or not.
- Constraint (5) ensures that a vector is either packed entirely in a bin or not.

4.6 Design of simplex Method



4.7 Algorithm for feasibility check

```
Data: Linear program dictionary
Result: Yes or No
//$b$ is R.H.S. of basis variables;
while $(b_i) \in b$ do
    if $(b_i) < 0$ then
        return No;
        break;
    end
    return Yes;
end
```

Algorithm 1: Algorithm for feasibility check

4.8 Algorithm for entering variable analysis

```
Data: Linear program dictionary
Result: Entering variable index
//$C$ is objective function coefficient;
while $(C_i) \in C$ do
    if $(C_i) > 0$ then
        Entering_Variable_Index = i;
        break;
    end
end
```

Algorithm 2: Entering Variable Analysis

4.9 Algorithm for leaving variable analysis

```
Data: Linear program dictionary
Result: Leaving variable index
//$b$ is RHS of basic variables;
//$A$ is coefficient of constraints;
for $i \in b_i$ do
    if $A_{i,entering\_variable} < 0$ then
        $bound_i = b_i/(-A_{i,entering\_variable})$
    end
end
Leaving_Variable_Index = which.min($bound$);
```

Algorithm 3: Leaving Variable Analysis

4.10 Pivoting

5 Work Done

Two scripts in R language were written. They are simplex.R and Bin_Packing.R

```
**Data**: Linear program dictionary ($D'$)
**Result**: Linear program dictionary ($D''$)
//$D', D''$ is dictionary of linear program;
$pivotElement = A_{leaveVariable,enteringVariable}$;
// Invert the row of pivotElement;
$A_{leaveVariable,} = A_{leaveVariable,}/(-pivotElement)$;
$A_{leaveVariable,enteringVariable} = 1/(pivotElement)$;
$b_{leaveVariable} = b_{leaveVariable}/(-pivotElement)$;
//replace the leaving Variable in constraints;
for $b_i \in b \wedge i \neq leaveVariable$ do
    $temp = A_{i,enteringVariable}$;
    $A_{j,} = A_{j,} + temp * A_{leaving_Variable,}$;
    $A_{j,entering_Variable} = temp/pivotElement$;
    $b_j = b_j + temp * b_{leaving_Variable}$;
end
```

Algorithm 4: Pivoting

6 Program Specification of Bin Packing

6.1 Input:

List (L) of p_i, pieces, in the range (0,1] for $i = 1..n$.
Example:

- 0.1,0.8
- 0.2,0.5,0.4,0.7,0.1,0.3,0.8
- 0.1,0.5,0.1

6.2 Output:

Dictionary (D) of Linear Program for the list (L). The dictionary is a data-structure for linear program. Dictionary is a representation of solution of linear program. Dictionary (D) contains the following data:

- m = number of basic variables
- n = number of non-basic variables
- B = indices of basic variables (basis set)
- N = indices of non-basic variables (independent set)
- b = RHS of constraints
- A = matrix of coefficients of rules(constraints)
- z0 = solution of the linear program
- C = co-efficients of objective function

6.3 Sample Input:

```
0.1
0.8
```

6.4 Respective Output:

```
10 6 # m n
7 8 9 10 11 12 13 14 15 16 # B
1 2 3 4 5 6 # N
1 1 -1 -1 1 1 0 0 0 0 # b
0 0 -1 -1 0 0 # A
0 0 0 0 -1 -1
0 0 1 1 0 0
0 0 0 0 1 1
0 0 0.1 0 0.8 0
0 0 0 0.1 0 0.8
1 0 -1 0 0 0
1 0 0 -1 0 0
0 1 0 0 -1 0
0 1 0 0 0 -1
0 -1 -1 0 0 0 0 # z0 C
```

7 Program Specification of Linear Program

7.1 Input:

Dictionary (D) of linear program
Example:

-
  ```
  10 6 # m n
  7 8 9 10 11 12 13 14 15 16 # B
  1 2 3 4 5 6 # N
  1 1 -1 -1 1 1 0 0 0 0 # b
  0 0 -1 -1 0 0 # A
  0 0 0 0 -1 -1
  ```

```
0 0 1 1 0 0
0 0 0 0 1 1
0 0 0.1 0 0.8 0
0 0 0 0.1 0 0.8
1 0 -1 0 0 0
1 0 0 -1 0 0
0 1 0 0 -1 0
0 1 0 0 0 -1
0 -1 -1 0 0 0 0 # z0 C
```

-
  ```
   3 4
  1 3 6
  2 4 5 7
  1 3 0
  0 0 -1 -2
  1 -1 0 -1
  -1 0 -2 0
  1 -1 2 3 1
  ```

7.2 Output:

Solution of the linear program represented by dictionary

Table 1: Exit Status

Status	Meaning
1	Problem is Infeasible
2	Problem is unbounded
3	solution is found

7.3 Sample Input:

Dictionary of linear Program:

```
3 4
1 3 6
2 4 5 7
1 3 0
0 0 -1 -2
1 -1 0 -1
-1 0 -2 0
1 -1 2 3 1
```

7.4 Respective Output:

```
šolution of dictionary¨
[1] 7
```

7.5 Program in human readable format

$$\begin{array}{r|lllll}
x_1 & 1 & & & -1x_5 & -2x_7 \\
x_3 & 3 & +1x_2 & -1x_4 & & -1x_7 \\
x_6 & 0 & -1x_2 & & -2x_5 & \\
\hline
z & 1 & -1x_2 & +2x_4 & +3x_5 & +1x_7
\end{array}$$

x_4 enters and x_3 leaves

$$\begin{array}{r|lllll}
x_1 & 1 & & & -1x_5 & -2x_7 \\
x_4 & 3 & +1x_2 & -1x_3 & & -1x_7 \\
x_6 & 0 & -1x_2 & & -2x_5 & \\
\hline
z & 7 & +1x_2 & -2x_3 & +3x_5 & -1x_7
\end{array}$$

7.6 Input of Bin Packing

```
10 6
7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6
1 1 -1 -1 1 1 0 0 0 0
0 0 -1 -1 0 0
0 0 0 0 -1 -1
0 0 1 1 0 0
0 0 0 0 1 1
0 0 0.1 0 0.8 0
0 0 0 0.1 0 0.8
1 0 -1 0 0 0
1 0 0 -1 0 0
0 1 0 0 -1 0
0 1 0 0 0 -1
0 -1 -1 0 0 0 0
```

7.7 Output

```
 Problem is Infeasible
```

7.8 Conclusion

Study of approximation algorithm of bin packing and implementation of simplex algorithm for linear program in R is done.

8 Future Work

There are algorithms of bin packing in non-linear programming. Next-fit, first-fit, first-fit decreasing and APTAS algorithms of bin packing are studied. Linear programming is implemented for bin packing. Wolfe's algorithm, LP relaxation of bin packing by Gilmore and Gomory, fast algorithms by D. Johnson, randomized algorithm for d-dimensional vector packing by Chekuri and Khanna exist in literature. Bin packing can be approached by mixed constraint non-linear programming.

The bin packing algorithm in literature may be extended to solve the instance of **task scheduling on cluster of heterogenous computer**. Heterogenous computer are machines which have different available resources. Heterogenous cluster may cotain computing resources/devices like mobile computers, laptop computers, and desktop conmputers. A **circuit** may be fabricated for task scheduling. Such machine may be concatinated with heterogenous cluster/LAN/WAN to utilize full computing resources of networked devices.

References

[1] Vazirani and Vijay V. *Approximation algorithms.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[2] W. Fernandez de la Vega and G. Lueker. Bin packing can be solved within 1 + epsilon in linear time. *Combinatorica*, 1:349–355, 1981. 10.1007/BF02579456.

[3] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '99, pages 185–194, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[4] Chetan Rao. Technical report, University of Wisconsin-Madison, 2012.

[5] Andrew Chi-Chih Yao. New algorithms for bin packing. *J. ACM*, 27(2):207–227, April 1980.