An-Najah National University

Faculty of Graduate Studies

Algorithms of Optimization Techniques for Bin Packing Problem: A Comparative Study

By

Yasmin EL Karmi

Supervisor

Dr. Baker Abdulhaq

This Thesis is submitted in Partial Fulfillment of the Requirements for the Degree of Master of Advanced Computing, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine.

2021

Algorithms of Optimization Techniques for Bin Packing Problem:

A Comparative Study

By

Yasmin EL Karmi

This Thesis was Defended Successfully on 28/09/2021 and approved by:

Defense Committee Members	**Signature**
1. Dr. Baker Abdulhaq / Supervisor	……
2. Dr. Amjad Rattrout / External Examiner	……
3. Dr. Amjad Hawash / Internal Examiner	……

Dr Amjad RATTROUT

Dedication

Much gratitude to my mother, my father, my husband, my daughter, my siblings, my mother-in-law and father-in-law for support, help and love.

Acknowledgement

At first, I would like to thank the wonderful doctor, my supervisor Dr. Baker Abdulhaq for his great support and patience. Also all my instructors in Advanced Computing faculty for the great knowledge I gain.

الاقرار

أنا الموقعة أدناه، مقدّمة الرسالة التي تحمل العنوان:

Algorithms of Optimization Techniques for Bin Packing Problem: A Comparative Study

أقر بأن ما اشتملت عليه هذه الأطروحة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد. وأن هذه الرسالة كاملة، أو أي جزء منها لم يقدم من قبل لنيل أي درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

Declaration

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

اسم الطالبة: **Student's Name: Yasmin ELKarmi**

التوقيع: **Signature:**

التاريخ: **Date: 2021/09/28**

List of Contents

List of Tables

List of Figures

List of Abbreviations

FF	First Fit
BF	Best Fit
BPP	Bin packing Problem
FFD	First Fit Decreasing
BFD	Best Fit Decreasing
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
SA	Harmony Search
TS	Tabu Search
ACO	Ant Colony Optimization
HS	Harmony Search
ABC	Artificial Bee Colony
FA	Firefly Algorithm
CS	Cuckoo Search
A1	Zehmakan’s Approximation Algorithm 1
A2	Zehmakan’s Approximation Algorithm 2
GGA	Grouping Genetic Algorithm

Algorithms of Optimization Techniques for Bin Packing Problem: A Comparative Study

By
Yasmin EL Karmi
Supervisor
Dr. Baker Abdulhaq

Abstract

One of the most critical optimization problems called Bin Packing Problem (BPP) attracts researchers attention because it is an NP-Complete problem means the solution can not be found in polynomial time. It has many applications such as storage and filling container.

BPP aims to pick several items with different weights and pack them in a minimum number of bins without exceeding the bin's capacity. One dimension BPP (1D-BPP) is one of its variations. Researchers have developed and proposed many algorithms to find an optimal solution or near-optimal solution.

This research aims to make a comparison between six algorithms to solve one-dimensional BPP. Two heuristic algorithms proposed by Zehmakan [?] are approximation algorithms; one of them has an approximation ratio of 3/2, called A1 and A2. Those algorithms promise to perform more efficient and much better than other algorithms.

Two classical approximation algorithms First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) and two meta-heuristic algorithm namely Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) with specific parameters have been compared.

In this work, several data sets have been used with the known optimal solution. They vary between random and arranged. Also, they vary in size. Some groups are small such as 9, 20 items, and medium such as 50, 100, 120 items and large such as 250, 500, 1000 items.

Moreover, the sets vary in difficulty between easy and medium. So the number of bins used and running time have been compared to consider these algorithms' performance.

According to the number of bins used, A2 has performed better than A1 by comparing heuristic algorithms. However, it took much more running time than A1, especially in large data sets. Nevertheless, classical heuristics (BFD FFD) outperform both A1 and A2 in easy datasets, while in hard datasets A2 outperform the classical heuristics.

By comparing meta-heuristic algorithms according to the number of bins used, in small data sets, PSO has performed better than GA but in large sets it's almost the same. Also, PSO takes double running time than GA. PSO and GA have close results by the number of bins comparison and running time comparisons in other data sets. PSO is slightly better than GA when both the heuristics and the meta-heuristics are compared. Heuristic performs more efficient according to the number of bins and running time

Chapter 1

Introduction

1.1 Research Background

Bin Packing Problem (BPP) is one of the challenging problems nowadays due to the wide range of applications in science and computing[19]. BPP has many variations, one dimensional BPP (1D-BPP), two-dimensional BPP (2D-BPP), multidimensional BPP, packing by weight… and so on. [7]

In this thesis, we consider 1D-BPP, which aims to find the minimum number of bins with a bin capacity (C > 0) to pack items of different sizes.

Researchers proposed many algorithms to solve BPP. There are two types of optimization algorithms: heuristic algorithms and approximation algorithms, designed for a specific problem, these algorithms are efficient and strightforward and leads to optimal or near-optimal solution such as classical approximation algorithm First Fit Decreasing (FFD) and Best Fit Decreasing, and Zehmakan's approximation algorithm 1 and 2 (A1 and A2)[19] which depends on classifying the items, each class belong to a specific range, these algorithms show a promising algorithm since the approximation ratio, a ratio between algorithm solution and optimal solution for A1 is 3/2 and A2 is based on the effective algorithm FFD [19].

On the other hand, a nature-inspired algorithms that are part of metaheuristic algorithms take a place in BPP researches. Unlike heuristic

algorithms, these algorithms are class-independent algorithms that can be used to solve several optimization problems. It's not designed for a specific problem such as Genetic Algorithms (GA) [12] and Particle Swarm Optimization (PSO) [13] which are used in this work.

These algorithms have parameters that need to be set to enhance their performance. Usually, it is not easy to find these parameters. researchers are still active to tune parameters depends on the problem.

GA inspired by the natural-selection in evolution theory where the best individuals are selected to produce new offsprings with a slight chance for a weak individual to survive. GA has several operators includes selection, crossover, and mutation, can be set in various ways.

In this thesis, operators are set as in Falkenauer (1996). For implementing BPP in GA, several gene representation schemes such as bin-based representation and group-based representation are used in this research.

PSO [13] inspired by the movement of birds flocking and fish schooling where each particle is trying to find the optimal solution by updated its position and sharing information with other particles. For implementing PSO for BPP, the continuous algorithm (PSO) will be discretized to fit BPP. In this research, we are using Binary Particle Swarm Optimization (BPSO) proposed by khanesar (2007) [14].

These heuristic and meta-heuristic algorithms are used in this research to compare them based on the number of bins used and running time to help answer the critical question about BPP: What is the best algorithm to reach

the optimal or at least near-optimal solution for BPP?

1.2 Research objectives

This study aims to find the best algorithm for solving BPP. The study aims to find the best algorithm that leads to the nearest optimal solution by minimizing the number of bins used to pack the items that leads to reduce the storage space.

The objective is to compare based on number of bins and running time between six algorithms. The following objectives were presented to achieve this goal:

- between recently proposed algorithms (zehmakan approximation algorithm "A1 and A2").
- between meta-heuristic algorithms (GA and PSO).
- between heuristic and meta-heuristic algorithms.
- with classic heuristic algorithm (First Fit) (FF) and Best Fit (BF) algorithms).

1.3 Research hypotheses

In this work, we test several hypothesis as the following:

- The classical approximation algorithm (BFD, FFD) performs better than other approximation algorithms such as Zehmakan's approximation algorithms.

- According to researchers' work for solving BPP, the meta-heuristic algorithms' performance such as PSO and GA and tuning parameters and population size.
- The performance of the algorithms based on the number of bins and running time.

1.4 Thesis structure

This thesis contains five chapters as follows:

Chapter two is a theoretical background consists of three parts. The first is about an introduction to optimization problems, mathematical formulation and types of problem. The second one is about Bin Packing problem definition and mathematical formulation. The last part is about optimization algorithm techniques A1, A2, GA, and PSO.

Chapter three is related to the methodology, it explains how each algorithm is used to solve the Bin Packing Problem and tuning parameter for metaheuristic algorithm.

Chapter four discuss the experiments and results of the performance of the algorithms and analysis of the data.

Chapter five is the conclusion of all the work. Also, it considers researches and promising projects for the future in this field.

Chapter 2

Theoretical Background

2.1 Optimization Problems

The optimization problem is significant problems in mathematics and computer-like grouping problem. So what is the definition of an optimization problem? Why are they important? Furthermore, what are the types of these problems?

2.1.1 Definition

The term optimization means making something reaching the best phase, so optimization algorithms are about finding the optimal values for a set of feasible solutions. This kind of problems form most daily life problems such as finding the smallest path to work, the best way to save money and arranging books in boxes or even more complicated issues such has Scheduling log cutting in forests, airline reservation and GPS [1], [18].

2.1.2 Mathematical Formulation

The following mathematical formula. describes the optimization problem:

$$Minimize/Maximize \tag{2.1}$$

$$\text{Subject to } x \ \epsilon \ \Omega \tag{2.2}$$

Where f is the objective or cost function for the set of the decision variables x which are a subset of the constraints set [5]. As mentioned

before optimization problem is how to find the best solution, the ″best″ of the values of vector x depending on the objective function whether it's maximization or minimization of values of the feasible set.

2.1.2.1 Types of optimization problem

Many algorithms are proposed to solve optimization problems, to choose the suitable algorithm to reach at least a near-optimal solution, you have to know what type of optimization problem is.

1. Continuous optimization vs discrete optimization.

In discrete optimization, feasible solutions must belong to a discrete set. Unlike continuous optimization, a possible solution is derived from real numbers domain [23].

2. Unconstrained optimization vs constraint optimization.

It depends on the constraints whether they exist on the objective function while optimizing it or not.

3. One objective or many objectives.

Most of the optimization problems have one objective function mentioned before. However, some cases have many objective functions, such as in engineering problems.

This thesis deals with one of the widely known problems: discrete, constrained, and one objective problem called Bin Packing Problem (BPP).

2.2 Bin Packing Problem (BPP)

2.2.1 What is BPP?

Bin Packing Problem is one of the most important and widely used optimization problems because it is an NP-hard combinatorial problem. In addition to computing and science's essential applications, especially in storage, resource allocation and scheduling such as filling containers, machine scheduling, and technology mapping. Thus, many variations have been proposed and developed such as two-dimensional BPP, Multi-dimensional BPP, packing by cost...etc. [19].

Therefore many researchers were interested in finding the best algorithm to solve it. Books, articles, working papers were published about this problem as shown in figure 2.1. Many algorithms have been proposed from the thirties until now such as First Fit, Best Fit, Genetic Algorithm, and Simulated Annealing, scientists keep looking for the better, faster and more accurate algorithm. [6]

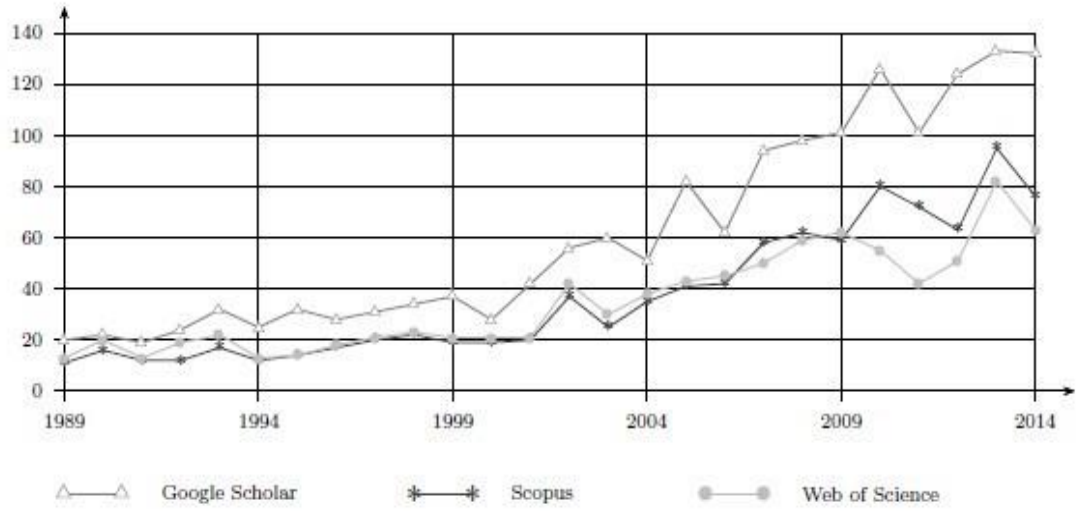

Figure 2.1: number of articles published about BPP through the years [6].

BPP aims to pack several items with different weights in the minimum number of bins, the capacity of these bins must not exceed, this problem is a particular case of Cutting Stock which is about finding the minimum material with a known size that needed to be cut into specific parts of different sizes. Also, Knapsack Problem is a particular case of BPP that aims to find the maximum value of items packed in a bin (bag) with known capacity where capacity must not exceed, unlike BPP, in knapsack problem not all items have to be packed. Each item has a specific weight and value. In general, these problems, which called packing optimization problems, change according to the number of bins and items. [6]

2.2.2 Mathematical Formulation

The Bin-Packing Problem (BPP) can be formulated as the follows: Each item with specific weight will be packed to one bin without exceeding the capacity (2.4) of the bin and minimum of bins. The mathematical formulation of the problem is

$$\text{minimize} z = \sum_{i=1}^{n} y_i \tag{2.3}$$

$$\text{subject to} \sum_{j=1}^{n} w_j x_{ij} \leq c\ y_i \quad , i \in N = \{1,2,\dots,n\}, \tag{2.4}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j \in N, \tag{2.5}$$

$$y_i = 0\ or\ 1, \qquad i \in N, \tag{2.6}$$

$$x_{ij} = 0\ or\ 1, \qquad i \in N\ ,\ j \in N, \tag{2.7}$$

Where

Z=Number of bins.

n=Number of items.

w_i=weight of item j.

c =capacity of each bin.

$$y_i = \begin{cases} 1 & if\ bin\ i\ is\ used; \\ 0 & otherwise, \end{cases} \quad (2.8)$$

$$x_{ij} = \begin{cases} 1 & if\ item\ j\ is\ assigned\ to\ bin\ i; \\ 0 & otherwise \end{cases} \quad (2.9)$$

$$0 < w_j \leq c \quad for \quad j \in N \quad (2.10)$$

[16]

This problem has two constrains, in (equation 2.4) items must not exceed the bin's capacity, and in (equation 2.5) no item could be placed in two bins and all items have to be packed.

2.3 Optimization Algorithms

To solve any optimization problem, we have to determine the objective. Then the optimization algorithm will find the feasible solutions for the problem.

Picking the suitable optimization algorithm for the problem depends on "the nature of the algorithm, the desired quality of solutions, type of problem, the available computing resource, time limit, the availability of the algorithm implementation, and the experts of the decision-makers".

[31]

picking the suitable optimization algorithm for the problem depends on algorithm's nature and type of problem as mentioned in (2.1.3). In addition to time limit, complexity, the experts of decision-makers and how easy and available the algorithm to implement. [31]

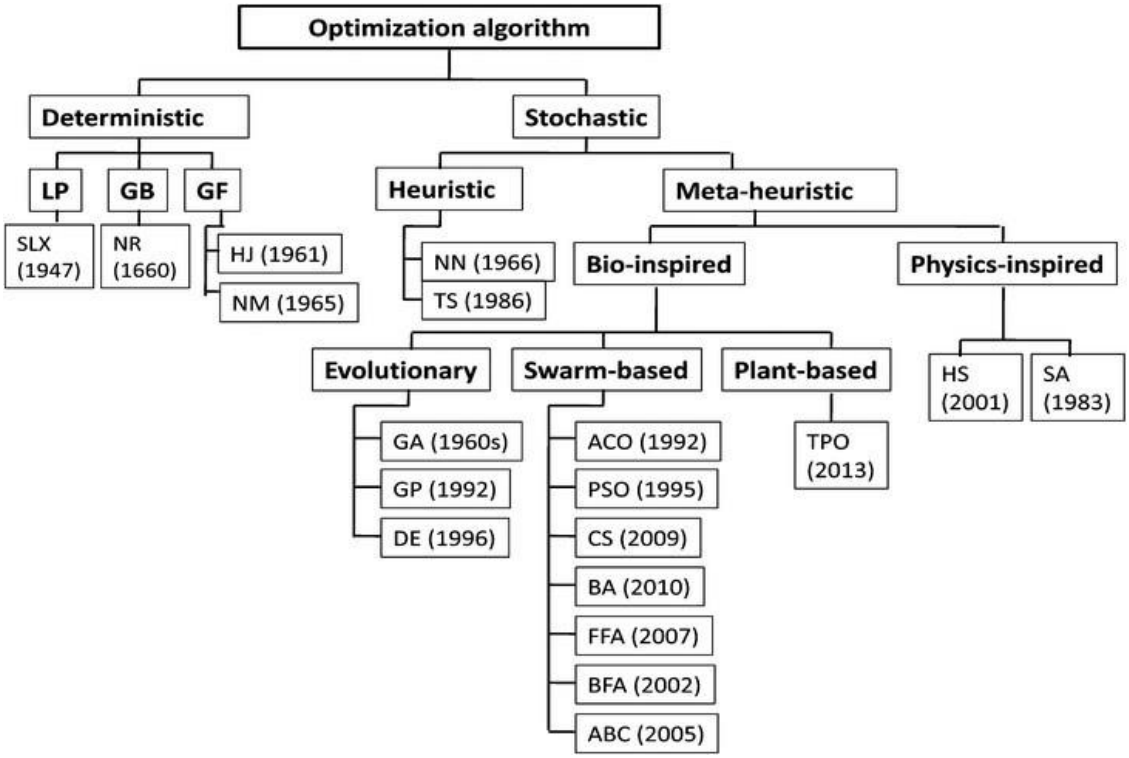

Figure 2.2: Optimization Algorithm Classification[11]

So optimization algorithms can be classified into two classes: Deterministic algorithms which are the algorithms that do not have any randomness in them so running them several times with the same initial point will give the same final results such as zehmakan approximation 1 (A1) and Hill climbing. Unlike Stochastic algorithms that have randomness, leadsing to multiple final results when running the program,

whether it starts with the same initial value or not such as Genetic Algorithm and PSO. [31]

2.3.1 Heuristic and Approximation Algorithms

Heuristic algorithms are class-dependent algorithms that can solve only a specific problem by choosing the next step to find a solution based on the collected information. These algorithms can lead to optimal or near-optimal solutions [30]

Heuristics can be classified into two classes: first one is local based heuristics, which can find the solution for all items a whole iteratively like in [24]) who starts with a subset of items and pack them into bins, evaluates the maximum free space and starts unallocated them until reaching to 0 unallocated items in 2D BPP. These algorithms are known to be fast and effective. The other classification is a construction based that deals with each item and packed it until all items are packed. Suppose this algorithm has a guarantee on the solution (α), it will be called approximation algorithm like in First Fit (FF) which takes an item and place it in the first bin without exceeding the capacity until all items are packed. Also as in Best Fit (BF) which takes an item and places it in the bin with the minimum free space that fits it, and many more algorithms [24]

As mentioned before, heuristic algorithms are effective and fast since it can find a suitable solution in a polynomial-time where the running time is guaranteed but there’s no guarantee to find the optimal solution. In the

other hand, approximation algorithms guaranteed to reach a near optimal solution in polynomial time.

The guarantee on the performance of these algorithm is also called the approximation factor or approximation ratio. If approximation algorithms' objective function is minimizing then the approximation ratio should be at least one, otherwise, in maximization function, the approximation ratio should be at most one [19].

This ratio can be measured to be the ratio of the measured function and the measure of the optimal solution as the following equation:

$$\alpha(A) = min_{I \epsilon z} \frac{m_I(A(I))}{m_I(Opt(I))} \quad (2.11)$$

Where,

z : optimization problem

A: optimization algorithm

I : items of the problem

Sol(I) is the set of feasible solutions to I

m_I : Sol(I) → R is the measure function associated with I, and

Opt(I) ⊆ Sol(I) is the feasible solutions with optimal measure (be it minimum or maximum)

Suppose the optimization problem is a minimization problem. In that case, the solution of approximation algorithm A(I) is trying to reach the optimal solution OPT(I). Therefore, A(I) is bigger than OPT(I), then α is at least one.

Furthermore, suppose it is a maximization problem. In that case, the

approximation algorithm A(I) is less than OPT(I), then the approximation ratio is less than 1.

The approximation algorithms are mathematically useful because this performance ratio gives an idea of how it will perform if the items perform well or not[25]

2.3.2 Meta-heuristic Algorithms

Meta-heuristic algorithms are independent algorithms that can solve many problems, not specific for one problem and create its design. These algorithms are developed from heuristics since it selects heuristic algorithms and modifies them with a degree of randomness that leads to a better solution without any previous knowledge about it. Also, it protects the solution from getting trapped in local optima[32] [31].

meta-heuristic algorithms have been proposed in recent years. These algorithms are classified into nature-inspired algortihms called bio-inspired algorithms inspired by the nature of animals and birds. and non-nature-inspired algorithms.[3]

Bio-inspired algorithms like ant colony algorithms mimic the ant routine in finding food by marking its path with Pheromone if food is found. it will come back from the same direction and mark it again with Pheromone to concentrate it more to lead other ants to the food source. Such as in the Genetic algorithm, Bee Algorithm (BA), Bat Algorithm (BA) and Particle Swarm optimization. In comparison, Tabu Search (TS) is the non-inspired

algorithm.[31]

classification for meta-heuristic algorithms, the population-based algorithms that depend on multiple agents, each agent produces a solution such as individuals in the genetic algorithm and particles in PSO. Unlike Trajectory based algorithms which use only one agent in each iteration in the search space to find the optimal solution such as Tabu Search (TS) and simulated annealing [31] [3].

Meta-heuristics have parameters that need tuning; most of the parameters are related to each other, making it challenging. The tunning process can be done through the execution such as Adaptive PSO (APSO) [35]. or before the implementation such as in GA.[9]

In this work, for solving BPP, we compare Genetic algorithm (GA) with Particle Swarm optimization (PSO).

2.3.2.1 Genetic Algorithm (GA)

Genetic algorithms proposed by J. Holland [12] based on the Darwinian principle, are population-based algorithms that use multiple individuals. These individuals consist of chromosomes, and each chromosome consist of genes and genes are composed of 0s or 1s see figure 2.3 [17].

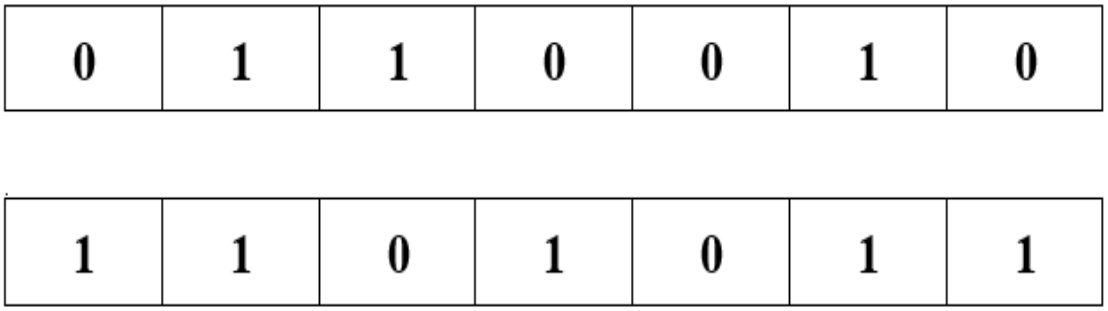

Figure 2.3: Chromosome and gene

Chromosomes are the basic stones in GA, by initializing population consist of chromosomes and evaluating its fitness. Then select two individuals for mating. After the selection, chromosomes will crossover by swapping genes from them and producing a new offspring. After that, chromosome needs to be mutated by a small random probability that changes the chromosome to maintain diversity. This leads to producing a new solution in each generation. The fitness function evaluates the solution's performance; the higher, the better (Yang, 2014). General steps of GA as follows:

1. initialize population.

2. while stopping criteria not satisfied.

   - select parents for mating .
   - crossover paretns to produce new offsprings.
   - mutate the new offsprings.
   - evaluate the fitness of new offsprings.
   - population = new population.

theses are shown in figure 2.4 [17]

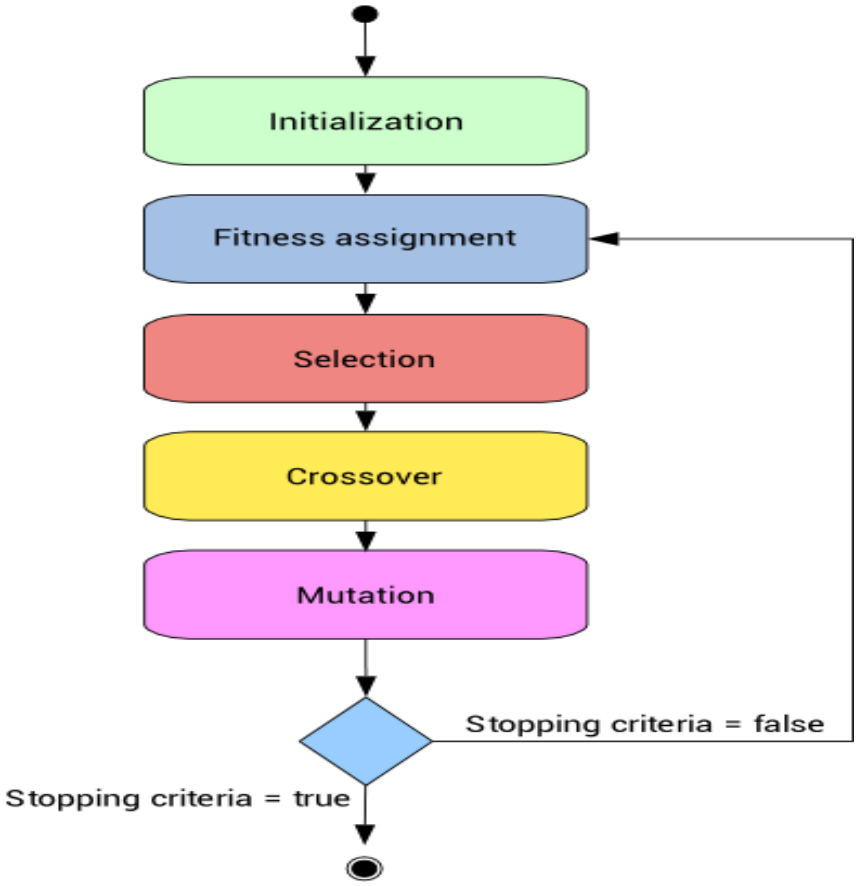

Figure 2.4: Genetic Algorithm flowchart [37]

2.3.2.2 GA operaters

1. Selection Selection is process of choosing the best individuals to form a new population as shown in figure 2.5

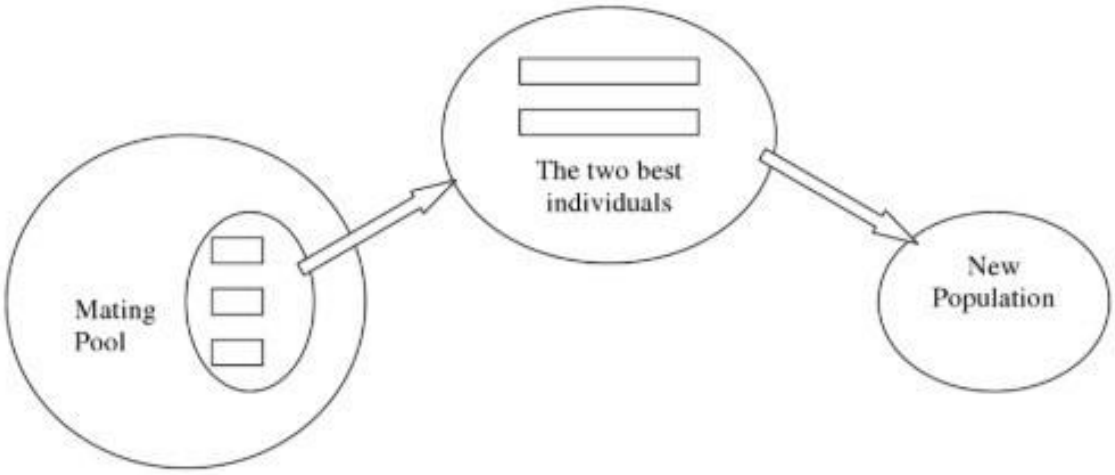

Figure 2.5: Selection process

To achieve this step, many methods were proposed such as:

(a) Roulette Wheel Selection: which also called fitness proportionate proposed by [12] because it depends on evaluation of the fitness of all chromosomes and finds their proportion by measuring the ratio of chromosome fitness and sum of all chromosomes fitness as shown in figure 2.6. chromosome 1 is the fittest, and chromosome 4 is the worst. chromosome 1 will have a bigger chance when the wheel rolled.[27]

(b) Tournament Selection: proposed by [10]. the number of chromosomes is selected randomly to choose chromosomes for the mating pool. A number of the selected chromosomes called the tournament size. The fittest chromosome will be chosen for mating pool[4].

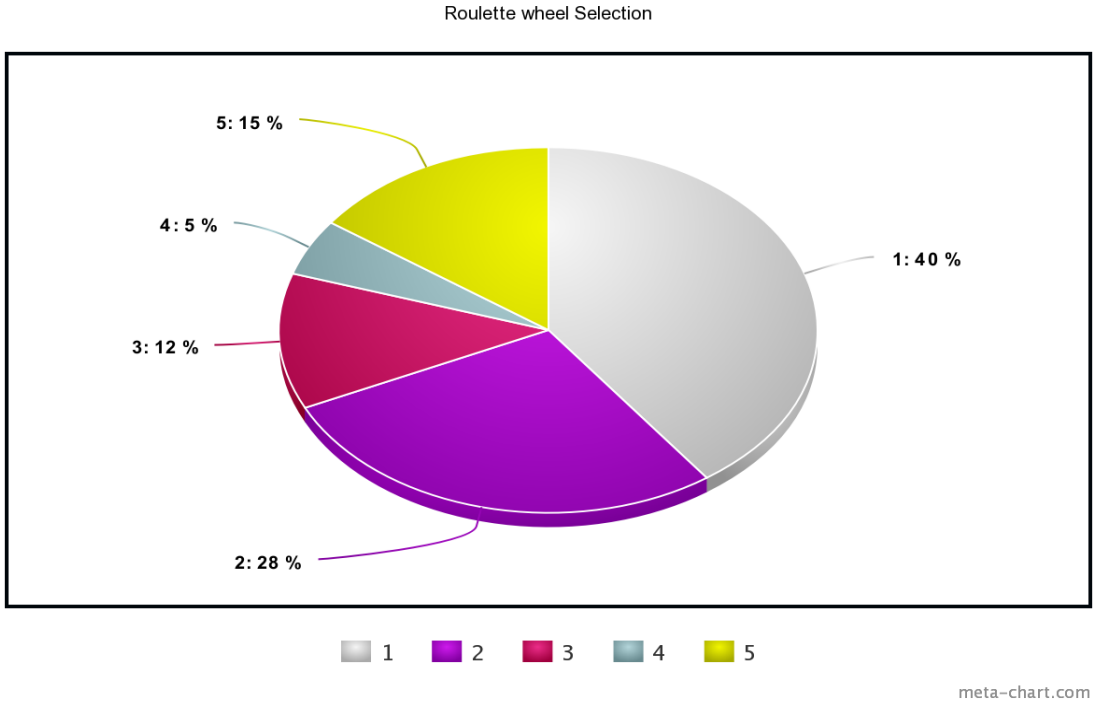

Figure 2.6: Selection Methods

2. Crossover Crossover is the process of swapping the selected parents from the previous step in some parts of it with a high random probability to produce a new offsprings by one of the crossover methods such as One Point Crossover, N-points Crossover, Uniform Crossover. as shown in figure .2.7[26] [31]

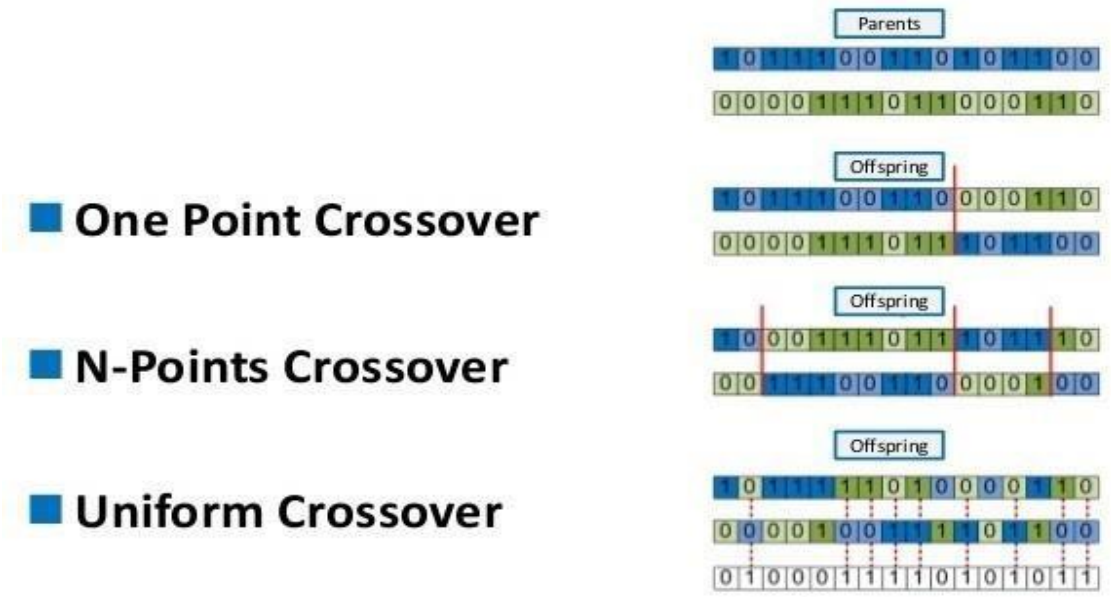

Figure 2.7: Cross over operators[21]

In figure 2.7 chromosomes in one point crossover methos are split into two string by a random point. One string in chromosome 1 is swapped with one string in chromosome 2, Both strings are on the same side, producing a new offsprings. this method was proposed by Holland [12] [26].

Another method is similar to one point crossover is N-point crossover. Instead of choosing one random point to split chromosome, at least two points are chosen randomly such as in figure 2.8 3 random points split the chromosome, the second and the forth chromosomes are swapped[15].

Syswerda proposed a method with no split points. however, it deals with each gene separately [28] in figure 2.8, each gene randomly chooses one of the parent's genes. each gene inherits some of one of the parent's genes. [26]

3. Mutation is the process of changing a gene in the chromosomes by swapping the gene with another gene in the same chromosome or put it in a random place with a low random probability as shown in figure 2.8, unlike crossover, mutation maintains diversity because crossover operator may lead to similar chromosomes, so there is no diversity to expand the search space and explore it [27] [4].

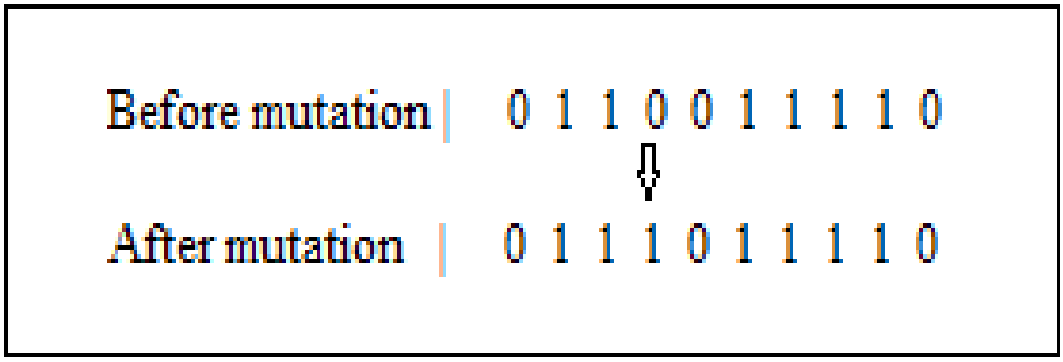

Figure 2.8: Mutation Method

2.3.2.3 Particle Swarm Optimization (PSO)

PSO was developed by Kennedy and Eberhart in 1995 [13] it mimics the behavior of fish and bird schooling [31]. "PSO maintaining strong abilities of convergence and global search" [33], it had many variations such as Adaptive Particle Swarm Optimization (APSO) [35], Particle Evolutionary Swarm Optimization (PESO) [34]…etc. figure 2.9 shows the movement of PSO.

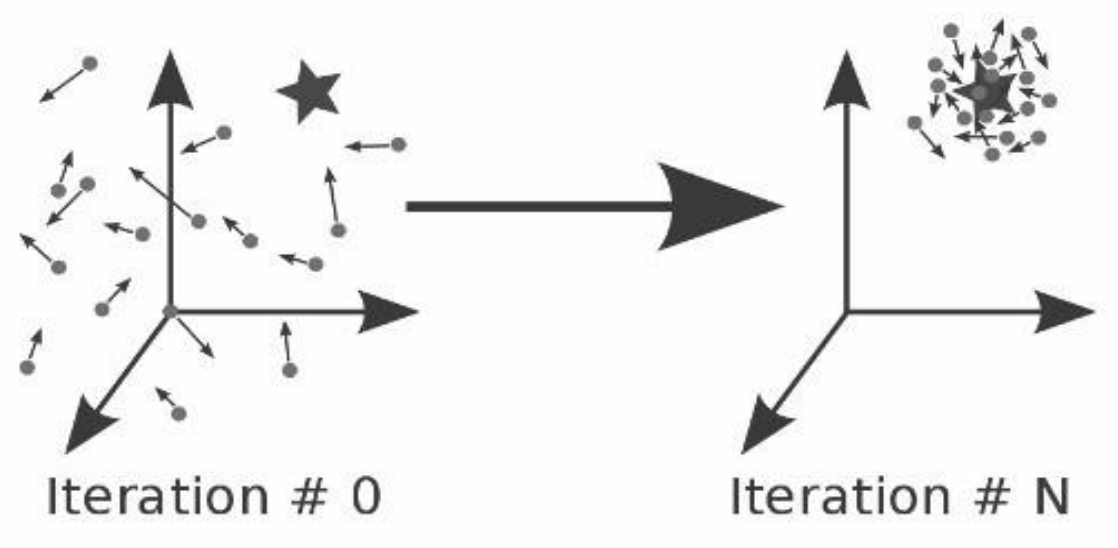

Figure 2.9: PSO movement [29]

As mentioned before, optimization algorithms are classified into stochastic

and deterministic algorithms. PSO saves the best position and global best position. It has a randomness behavior, so it is a stochastic algorithm; the particle's movement is measured by updating the particle's velocity and particle's position as follows: [31]

$$v_i^{t+1} = w * v_i^t + c_1\, rand_1^t(pBest_i^t - x_i^t) + c_2\, rand_2^t(nBest^t - x_i^t)] \quad (2.12)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2.13)$$

x_i : position vector

v_i : velocity vector

pbest : best position

nbest : global best position

$rand_1^t, rand_2^t$: random numbers between [0,1]

c_1 self-cognition.

c_2 social influence.

Since PSO mimic the movement of birds, suppose each particle is a bird searching for a place to land see figure 2.10, each particle has it is own current position $x_i(t)$ and current velocity $v_i(t)$, to update the bird's velocity.

The particle depends on the local and global search, then the self-cognition factor (c_1) and the social influence factor (c_2) need to be balanced. If the cognition factor is larger than the social factor, then the local search ability will be more significant. If the social facor is bigger than the cognition factor, then the global search ability will be more significant.[22] [35]

The following figure 2.10 shows the particle $x_i(t)$ updated its position to $x_i(t+1)$ depending on the local best position and global best position.

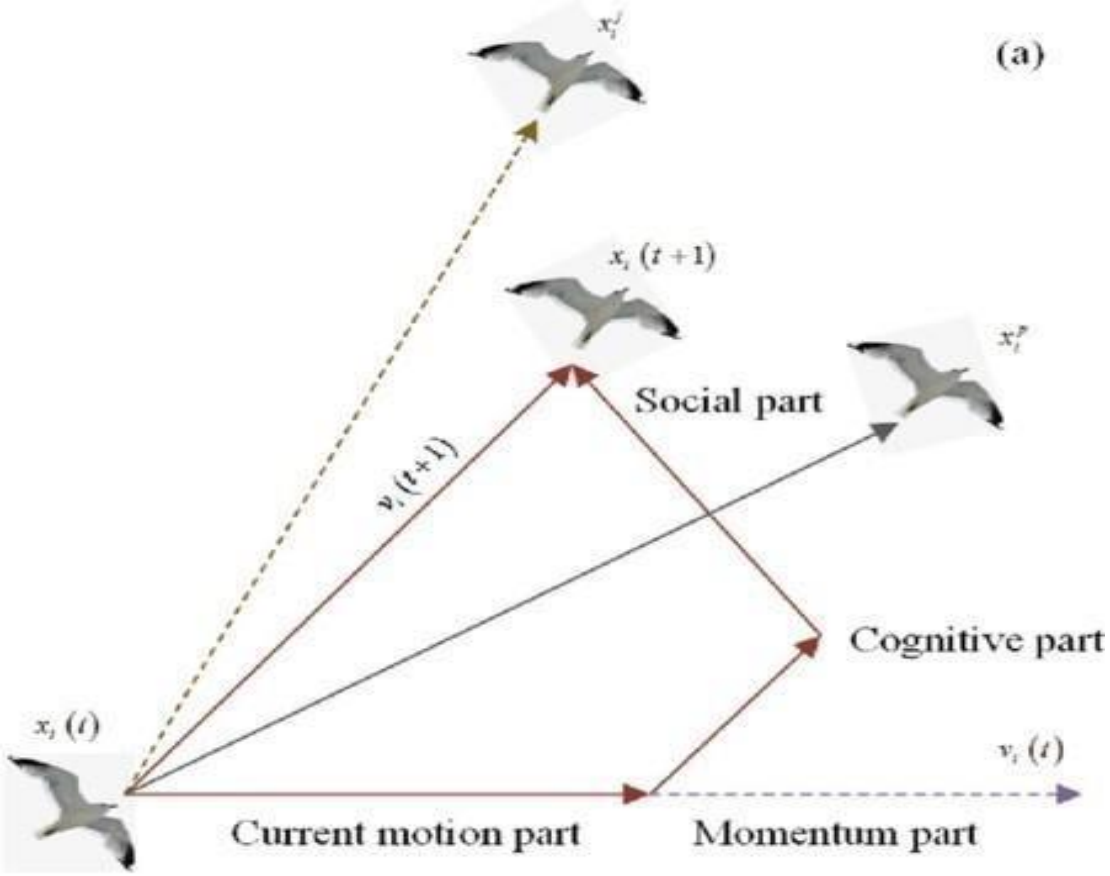

Figure 2.10: Particle Movement [20]

Chapter 3

Methodology

3.1 Introduction

The focus of this study is on finding the most near-optimal algorithm of one dimensional BPP. One of the approaches is the heuristic algorithms, also called approximation algorithms, when the solution guarantees on how much the solution is near the optimal solution. Also, there are meta-heuristic algorithms that mimic the best features of nature.

This study is a quantitative and experimental research type. Four optimization algorithms were experimented and compared to two of the most efficient approximation algorithms on several benchmark problems with an already known optimal solution. This comparison is based on the number of bins used and running time.

3.2 Optimization Algorithms

This work is showing a comparison between six optimization algorithms. On one hand, four of them are approximation algorithms, the well-known Best Fit Decreasing (BFD), First Fit Decreasing (FFD) and Zehmakan approximation algorithm (A1 and A2). On the other hand, two meta-heuristic algorithms, Genetic Algorithm, and Particle Swarm Optimization. These two algorithms have parameters that need to be specified.

A1 and A2 have recently proposed algorithms, these algorithms are chosen because the approximation ratio is 3/2 that means the approximation ratio

is close to optimal solution, which is the best ratio that has been reached so far. also they are efficient and one of them is based on First fit algorithm and simple [19].

These algorithms have been tested in eight sets of instances of BPP from OR Library. They been compared to Guochuan's algorithms and Berghammer's algorithm [36] [2] and have shown much better and more effective and efficient results than them. Also, it showed interesting results by comparing it with FFD, and they are parameter-less.

Other algorithms were chosen, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The meta-heuristic algorithms are suitable for BPP because it's one of the most effective, practical and successful approaches. It has very essential concepts: intensification and diversification, and its simplicity and flexibility.

To test the efficiency of these algorithms in bin packing problem, it has been tested on some of the benchmark problems and compared with each other and with FFD, BFD.

3.2.1 First Fit Decreasing (FFD)

One of the classical approximation algorithm for BPP, based on arranging the items in the decreasing order, then place the items one by one on the first bin that fits the item without exceeding the capacity. Figure 3.1 shows an example on BFD, seven items with bin capacity 10.

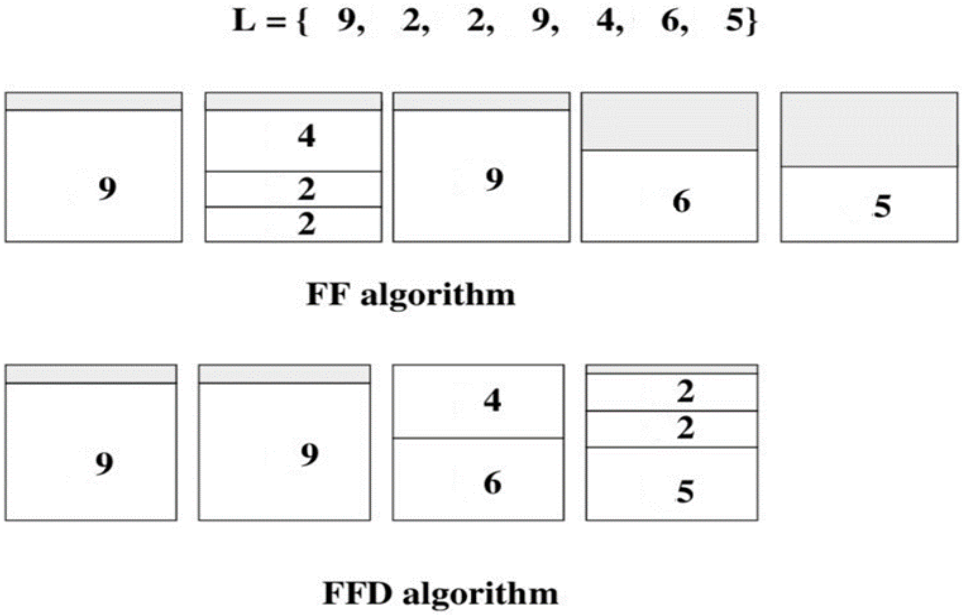

Figure 3.1: FF vs FFD example

3.2.2 Best Fit Decreasing (BFD)

Another classical approximation algorithm which also based on arranging the items in the decreasing order. Unlike FFD, BFD places the items in the best bin fits without exceeding the capacity and with less free space. Figure 3.2 shows example on BFD and FF that was mentioned before, 7 items with bin capacity 10.

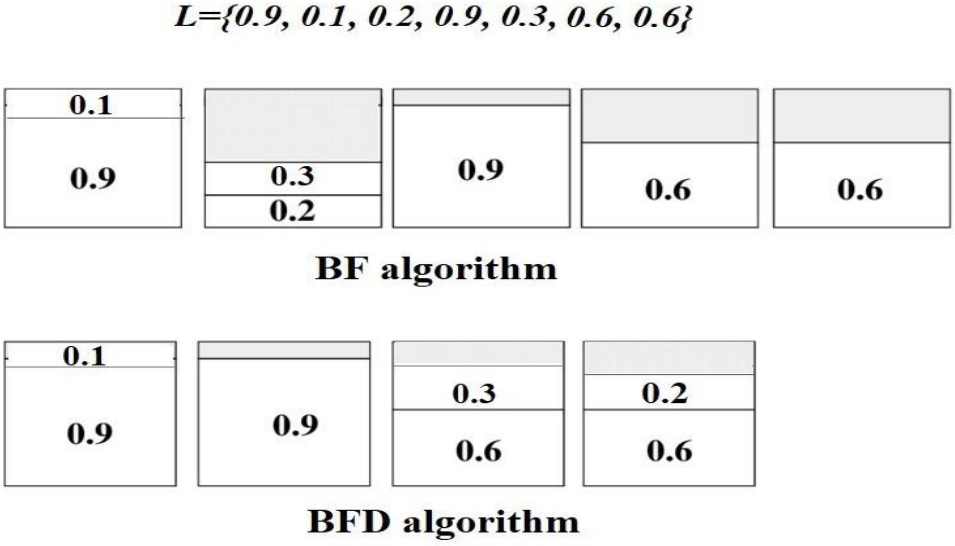

Figure 3.2: BF vs BFD example

3.2.3 Zehmakan's approximation algorithm 1 (A1)

A1 is based on ranking and classifying technique, which aims to create output bins, items will be classified into 4 ranges (S, M1, M2, L) to fill at least 2/3 of output bins as follows:

S = (0 - 1/3)

M1= (1/3 - 1.5/3)

M2= (1.5/3 - 2/3)

L= (2/3 - 1)

start with L item by placing them directly and separately into bins. Each item in M2 will be matched with the biggest item in M1 without exceeding the bin's capacity. If any item left in M1, then it will be matched with each other leading to fill at least 2/3 of bin. If M2 is not empty, then M2 will match with at least one item from S, and the last step is matching the remaining S items with each other.

Most of the output bins are 2/3 full, therefore, this algorithm has approved that its approximation ratio is 3/2 as follows :

Theorem 1: If all the output bins of the proposed algorithm A1 are at least 2/3 full, then the approximation ratio is at least 3/2.

Proof:

1. L items fill at least 2/3 of bin capacity.

2. M2 has a space of (1/3 - 1.5/3) that matches with M1 items with a range (1/3 - 1.5/3) to fill at least 2/3 of bin capacity.

3. the remaining items in M1 will match with each other and fill at least 2/3 of bin capacity.

4. the remaining items in M2 will be matched with some S items with range (0 - 1/3) without exceeding bin capacity. so we claim it fills more than 2/3 of bin capacity, as a result there are two cases :

Case 1: if all S items are matched and there still M2 items. The space of M2 items (1/3 - 1.5/3) it cannot be matched with L or itself, so the remaining items will be placed in separated bins.

Case 2: if all M2 items are matched and there still S items. the remaining S items are matched with each other, leading to fill 2/3 of bin's capacity.
then all the output bins are at least 2/3 full.

Suppose A* is the number of output bins in optimal solution where all the bins are filled, and A is the number of output bins found by A1 Algorithm. then the bin I can be discribed as follows:

$$(\mathrm{A}^{*}) > I > \frac{2}{3}A \tag{3.1}$$

then the approximation ratio as mentioned in 2.11

$$\alpha(A) = \frac{A}{A^{*}} < \frac{3}{2} \tag{3.2}$$

we proposed a modification on Algorithm A1. where L items will be matched with S items.

Theorem 2: If the modified Algorithm's output bins are 2/3 full, then the approximation ratio is 3/2.

Proof:

1. L has a empty space of (0/3 - 1/3) that matches with S items with range (1/3 - 1.5/3) to fill more than 2/3 of bin capacity.

2. M2 has a empty space of (1/3 - 1.5/3) that matches with M1 items with range (1/3 - 1.5/3) to fill at least 2/3 of bin capacity.

3. if any M1 items have remained, they will be matched with each other and fill at least 2/3 of bin capacity.

4. if any M2 items and S items have remained, will be matched togather to fill at least 2/3 of bin capacity.

Then all the output bins are at least 2/3 full.

As proved before, if all the bins are 2/3 full, then the approximation ratio is 3/2.

Algorithm 3.1 shows the pseudocode and the steps of A1

Algorithm 3.1:
Zehmakan's Approximation Algorithm 1 (A1)

Read inputs & classify them into S, M_1, M_2, and L
Sort M_1 & M_2
***For** (any item a in M_2)*
  ***If** (a can be matched with at least an item in M_1)*
    Match a with the biggest possible item in M_1;
    Eliminate them & Bin-counter ++;
  ***Else** continue;*
Bin-counter + = $\frac{|M_1|}{2}$;
***For** (any item a in M_2)*
  Do
    Choose an item b in S;
    $a = a + b$ & eliminate b & $c = b$;
  ***While** ($a \leq 1$)*
  Eliminate a & put c in S & Bin-counter ++;
***While** (S is not empty)*
  Choose an item a in S;
  ***While** ($a \leq 1$)*
    Choose an item b in S & $a = a + b$ & $c = b$;
  Eliminate a & Bin-counter++ & put c in S
End

3.2.4 Zehmakan approximation algorithm 2 (A2)

A2 is the new version of FFD. It considers 10 classes of bins $B_i = B_0, B_1, B_2, \ldots, B_9$, and 10 ranges $R_i = R_0, R_1, R_2, \ldots, R_9$ each range contains items that fill $(i * 0.1, (i + 1) * 0.1)$ of bin's capacity.

Although this algorithm is parameter-less, it has a scale parameter that presents the number of classes of bins and ranges.

Since A2 is based on BFD, matching items start with the biggest range

(R_9) until reaching the smallest one (R_0). all items in any range should be matched and placed in the suitable set of bins (B_i) that contains only bins with free capacity $(i * 0.1, (i + 1) * 0.1)$.

Each item will go through matching process to choose a suitable bin, start with B_0 until B_9, pick random bin from B_i, Check if there is a space for item. If there is a space, place it and move the bin to suitable set of bins. Else, search for another bin to place it. If there is no bin suitable for it, open a new bin.

The pseudocode of algorithm A2 and steps are shown in Algorithm 3.2

Algorithm 3.2:
Zehmakan's Approximation Algorithm 2 (A2)

***Consider** 10 sets of bins $B_i \; \forall \; 0 \leq i \leq 9$. A bin is in the set B_i if the bin's free space between (i*0.1) and ((i+1) 0.1). (At first all sets are empty;*
***Consider** 10 ranges $\boldsymbol{R_i = (i * 0.1, (i + 1) * 0.1)} \; \forall \; 0 \leq i \leq 9$.*
***Read** items from input.*
***Put** the items into corresponding ranges*
***For** $(i = 9; \; i > -1; \; i - -)$*
    ***While** (R_i is not empty)*
        ***Choose** an item a in R_i randomly;*
        ***For**$(j = 0; \; j < 10; \; j + +)$*
            ***Choose** a random bin in $\boldsymbol{B_j}$;*
            ***If** (the bin has enough space for a)*
                ***Put** a in the bin & change that to the appropriate $\boldsymbol{B_i}$;*
                ***Eliminate** a from $\boldsymbol{R_i}$;*
                ***Break**;*
        ***Put** the item a into an empty bin and put the bin into corresponding $\boldsymbol{B_i}$;*
        ***Eliminate** a from $\boldsymbol{R_i}$*

3.2.5 Genetic Algorithm (GA)

Many researchers proposed several genetic algorithm mechanisms to solve BPP in this work. We use [8], the Grouping Genetic Algorithm for Bin Packing Problem.

3.2.5.1 Gene Representation

There are many representation schemes for bin packing problems such as object-based representation, bin-used representation, Mohamadi's representation scheme [17]. The gene representation used in this work proposed by [8] which called group-based representation. Unlike most of the representation schemes that are item-oriented, group-based representations are group-oriented. The cost function depends on the bins groups where each bin represents a group of items. The items will be presented as 1, 2, 3... N, the groups will be labeled as A, B, C… as in the following example:

my caption 1

Items 1 and 3 are placed in group (bin) A, item 2 placed in group B, items 4 and 5 placed in group C, and item 6 placed in group D, so the number of bins is 4 in solution. The group part presented, for example, CABD represents the bins used, which are 4 bins, to use it in the genetic operations, so the item part need to identify the items as follows: A= 1, 3, B= 2, C= 4, 5, D= 6 Or 1, 3 2 4, 5 6

3.2.5.2 Cost Function for BPP

The usual cost function used is the minimum number of bins required to pack all items without exceeding the bin's capacity. We used another

function based on maximizing the following

$$f_{BPP} = \frac{\sum_i (\frac{F_i}{C})^k}{N} \quad , \quad i = 1 \ldots \ N \qquad (3.3)$$

Where,

N: number of bins,

F_i: the sum of item sizes in bin i,

C: bin capacity,

k: concentration on the most filled bin to the least filled bin, k>1.

3.2.5.3 Selection

After testing several tournament sizes in this work, the selection method used is tournament selection of size two. In algorithm 3.3, in figure 3.3 two tournaments are selected randomly from the population and evaluated the fitness value, the fittest chromosome will be chosen for the mating pool.

Poplation	Tournament	choosen individual
BCADE	ACDB	ACDB
ACDB	ABCD	
ABCDE		
ABDC		
ABCDEF		

Figure 3.3: Example of tournament selection of size 2

Algorithm 3.3: Tournament Selection

```
1: P ← population
2: t ← tournament size, t ≥ 1

3: Best ← individual picked at random from P with replacement
4: for i from 2 to t do
5:     Next ← individual picked at random from P with replacement
6:     if Fitness(Next) > Fitness(Best) then
7:         Best ← Next
8: return Best
```

3.2.5.4 Crossover

The chosen crossover method is one point crossover because it performs better than other methods in most of the scheduling problems.

First, the selected parents are copied to start the crossover process as in figure 3.4.

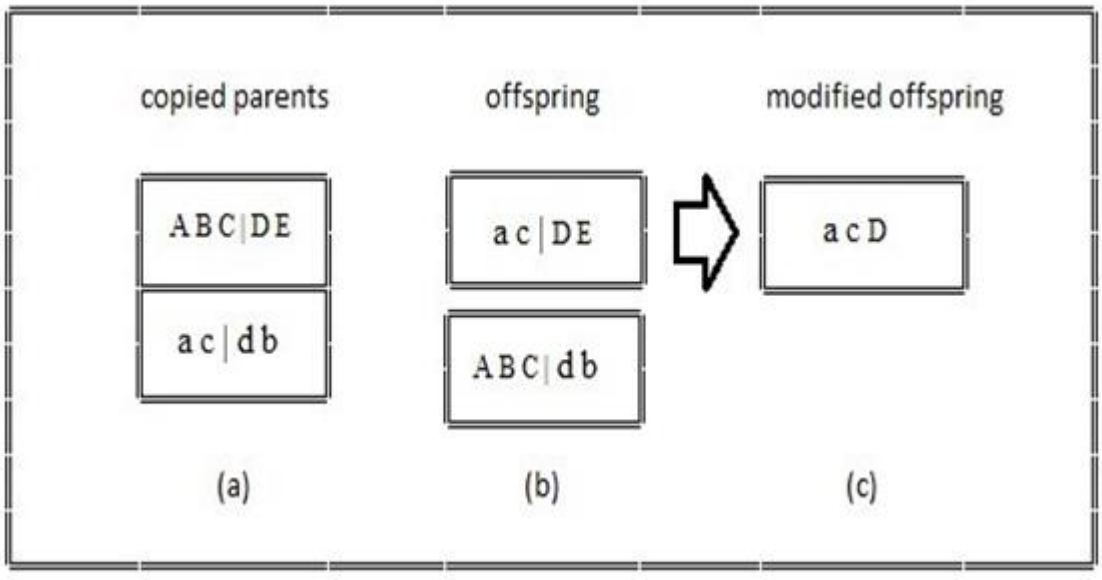

Figure 3.4: example of crossover

1. The crossover point is chosen randomly as in (A)

2. The first part of the first parent is replaced by the second part of the second parent

3. The first part of the second parent is replaced by the second part of the first parent producing offspring as in (B).

Since in BPP no items can be placed in two bins. Suppose E contains an item already placed in A then bin E is removed as in (C).

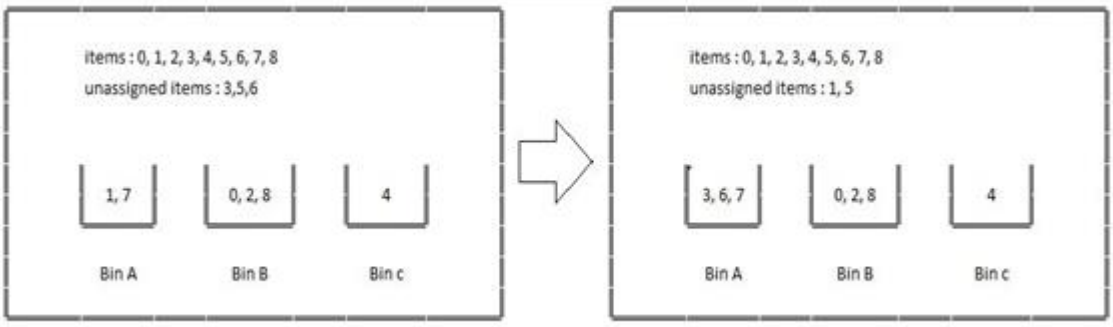

Figure 3.5: example of crossover

Suppose the removed bin E contains other items 3, 5, 6. These unassigned items need to be placed.[16], up to three of the assigned items in each bin can be placed by one or two of the assigned items as in figure 3.5. Item 1 is replaced by items 3 and 6, so item 1 will be unassigned and items 3 and 6 will be assigned items. Repeat this for all bins. If any items were still unassigned, then use FFD to assign them.

3.2.5.5 Mutation

Mutation is very simple; choose a random bin, then mutate it as in figure 3.6. The bin c is selected to be mutate to be removed, and the bin's items

will be unassigned. The unassigned items placed in bin c will be placed as in the crossover by using FFD to assign them.

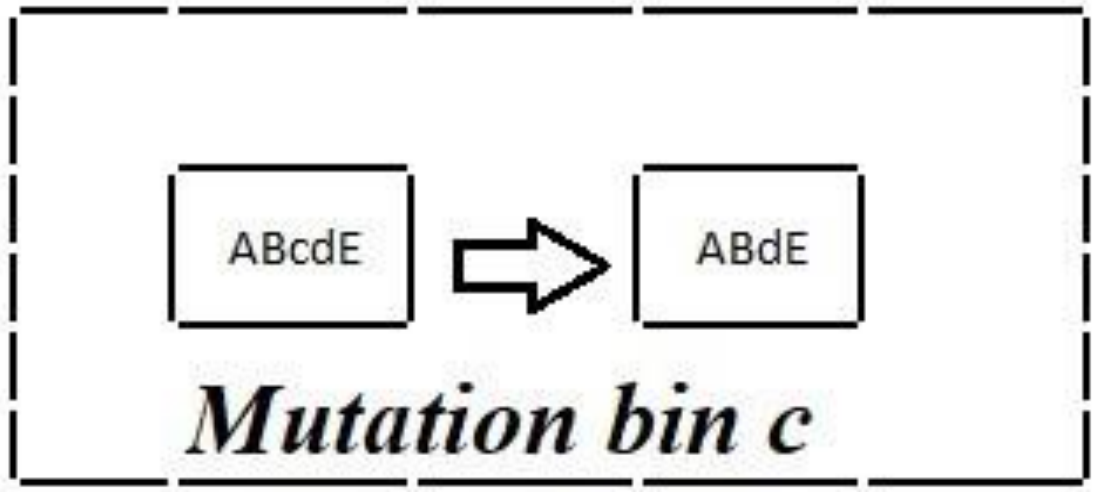

Figure 3.6: Mutation example

3.2.6 Particle Swarm Optimization (PSO)

Particle swarm optimization is one of the best meta-heuristic algorithms, but it's designed for continuous problems, so many researchers, proposed techniques to solve discrete problems such as this work problem, BPP.

Khanesar (2007) proposed a novel binary PSO that implemented in this work. The velocity of particle is a probability of bit. It takes one or zero. If the position of a particle changes into 1, the velocity will denoted as v_{ij}^{0} and if the position of particle changes into 0, the velocity will denoated as v_{ij}^{1}, as in 3.4.

and the particle best position and global best position depend on a fixed variable (c_1, c_2) and on random variables (r_1, r_2) as shown in 3.5 and 3.2.6.

$$v_{ij}^{c} = \begin{cases} v_{ij}^{0} & if\ \ x_{ij} = 1 \\ v_{ij}^{1}, & if\ \ x_{ij} = 0 \end{cases} \tag{3.4}$$

and

$$P_{ibest}^{j} = \begin{cases} 1\ , \ then\ \ d_{ij,1}^{1} = c_1 r_1\ \ and\ \ \ d_{ij,1}^{0} = -c_1 r_1 \\ 0\ , then\ \ d_{ij,1}^{0} = c_1 r_1\ \ and\ \ \ d_{ij,1}^{1} = -c_1 r_1 \end{cases}$$

(3.5)

$$P_{gbest}^{j} = \begin{cases} 1\ \ , \ then\ \ d_{ij,2}^{1} = c_2 r_2\ \ and\ \ \ d_{ij,2}^{0} = -c_2 r_2 \\ 0\ \ , then\ \ d_{ij,2}^{0} = c_2 r_2\ \ and\ \ \ d_{ij,2}^{1} = -c_2 r_2 \end{cases} \tag{3.6}$$

unlike standard BPSO that updates the velocity as in standard PSO in equation 2.13, Novel BPSO updates the velocity as in equations 3.7 and 3.8

to update the velocity of jth bit in ith particle, consider the best position is one, so according to equation 3.5, the temporary value $d_{ij,1}^{1}$ will increase and $d_{ij,1}^{0}$ will decrease. as a result, the velocity of $v_{ij,1}^{1}$ increases and $v_{ij,1}^{0}$ decreases as in equation 3.4.

and if the best position is zero, then the velocity of $v_{ij,1}^{1}$ increases and $v_{ij,1}^{0}$ decreases as in equation 3.8.

$$v_{ij}^{1} = w v_{ij}^{1} + \ d_{ij,1}^{1} + \ d_{ij,2}^{1} \tag{3.7}$$

$$v_{ij}^{0} = w v_{ij}^{0} + \ d_{ij,1}^{0} + \ d_{ij,2}^{0} \tag{3.8}$$

To update particle's position, the velocity must be normalized as in equation 3.9, then the position is updated according to equation 3.10.

$$v_i^c j(t) = sig(v_{ij}(t)) = 1/1 + e^{-v_i j(t)} \quad (3.9)$$

$$r_{ij} = \begin{cases} 1, & if \ \ v_i j(t) = sig(v_{ij}(t+1)) \\ , & otherwise \end{cases} \quad (3.10)$$

$$x_{ij}(t+1) = \begin{cases} x'_{ij}(t) \ \ , \ if \ \ r_{ij} < v'_{ij} \\ x_{ij}(t) \quad , if \ \ r_{ij} > v'_{ij} \end{cases} \quad (3.11)$$

Where

P_{ibest} : best position of particle i

p_{gbest} : global best position

r_1, r_2: random variables

c_1, c_2: fixed variables.

w: inertia weight

x: variable position.

d_{ij}^0, d_{ij}^1 : two temporary values.

The pseudocode of BPSO and steps is shown in algorithm 3.4 [14]

1. Initialize the swarm X_i , the position of particles is randomly initialized within the hypercube. Elements of X_i are randomly selected frombbinary values 0 and 1.

2. Compare the performance F of each particle, using its current position $X_i(t)$. Compare the performance of each individual to its best performance so far: If $F(X_i t) < F(ibest)$: $F(P_i best) = F(X_i(t) P_i best = X_i(t)$

3. Compare the performance of each particle to the global best particle: If $F(X_i(t) < F(P_g best)$.

4. Change the velocity of the particle, v_{ij}^1 and v_{ij}^0 according to 3.2.6 , 3.7.

5. Calculate the velocity of change of the bits, v_{ij}^c as in 3.4.

6. Generate the random variable $r_i j$ in the range: (0,1). Move each particle to a new position using equation 3.11.

7. Go to step 2, and repeat until convergence.

Chapter 4

Experiments and Results

4.1 Introduction

In this chapter, we will compare the performance of six algorithms, two of them are approximation algorithms proposed by Zehmakan (2015), two classical approximation algorithms (FFD BFD), and a widely used meta-heuristic algorithms (GA PSO) on Bin Packing Problem by comparing the number of bins and running time. Each algorithm runs 30 times.

In meta-heuristic algorithms, GA parameters and gene representation are set according to E.Falkenauer [8]. The used PSO binarization is proposed by Khanesar (2007). To assess the merit of all algorithms, in addition to the heuristic algorithms, we re-implemented them in the experiment. We use the t-test to analyze the results and draw them with MatLab. We used several benchmark problems as follows:

1. Type 1: BP1, BP2, BP3, and BP4 are benchmark problems presented by E. Falkenauer (1994) which are consisted of uniformly distributed items between 20 and 100 with bin capacity 150, And downloaded from BPPLIB.

2. Type 2: M1, M2, M3, M4, and M5 are easy benchmark problems considered by Mohamadi (2010) with a small number of items between 4 and 40.

3. Type 3: N1C1W1_ G, N1C2W1_ B, N2C3W1_ H, N3C2W1_ D, and N3C3W1_ E are data sets by A.Scholl and R.Klein(1997), where number of items are between 50 and 500, and capacity of the bin is varying between 50, 100, 150, the items are arranged in the decreasing order between 1 and 100.

4. Type 4: HARD0, HARD1.. . HARD9 also are data setes by A.Scholl and R.Klein(1997), where the number of items are between 20000 and 35000, and capacity of the bin is 100000.

5. Type 5: u120-1, u120-2,u120-3... u120-10 are benchmark problems from OR Library, which uniformaly distributed items are between 20 and 100.

4.2 Heuristic Algorithms Comparison

Zehmakan (2015) has proposed two algorithms (A1 and A2) to prove the efficiency of his algorithms. He tested them on type 1 of data sets, which mentioned previously, and compared them with Gouchuan's algorithm and Berghammer's algorithm [2]. Zehmakan's experiment is shown in figure 4.1

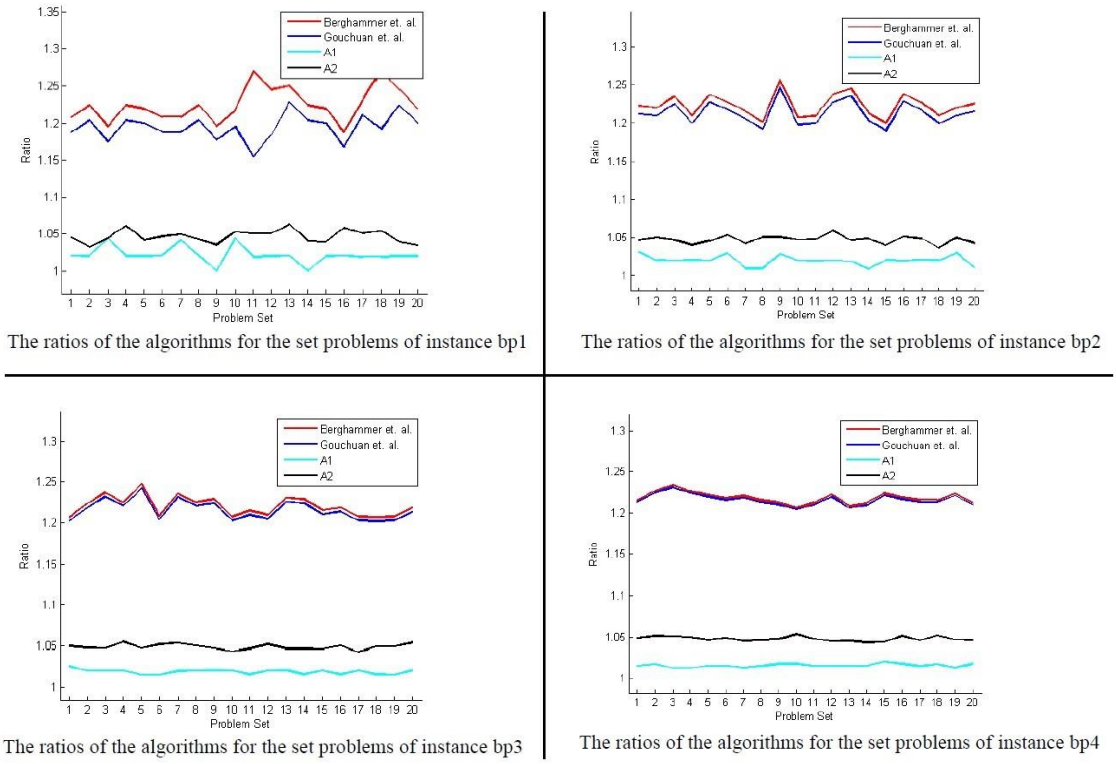

Figure 4.1: The ratio of the algorithms (A1, A2, Berghammer's algorithm, Gouchuan's algorithm) for the set problem of instances bp1, bp2, bp3 and bp4 [19]

The ratio of the algorithm is the proportion of the algorithm solution to optimal solution. As mentioned before, the approximation ratio of minimization problem should be at least one. Since the optimal solution is less than or equal to the algorithm solution.

In Zehmakan's (2015) experiment, figure 4.1 shows promising algorithms, A1 algorithm shows the best results among compared algorithms, while both A1 and A2 gave optimal or near-optimal solutions with a ratio at most 1.05 as shown figure 4.1. Unlike the other two algorithms [2], that gave a ratio at least 1.15 and never reach the optimality.

where A1 is more accurate and need less space than A2, but A2 shows better results in time complexity and the homogeneity of items.

These algorithms are re-implemented in this experience on the same benchmark problem type 1 data set. Due to stochastic features of A2, each problem set runs 30 times and the result is reported. The results are shown in figure 4.2.

Unlike Zehmakan's experiment A2 preforms better than the A1, also it is hard to reach optimality for A1 algorithm where the least ratio is 1.05, otherwise, A2 has a ratio at most 1.06 as it is described in Zehmakan's experiment. For time and space complexity. A1 shows better results than A2. But for the accuracy A2 is shown a very interseting results.

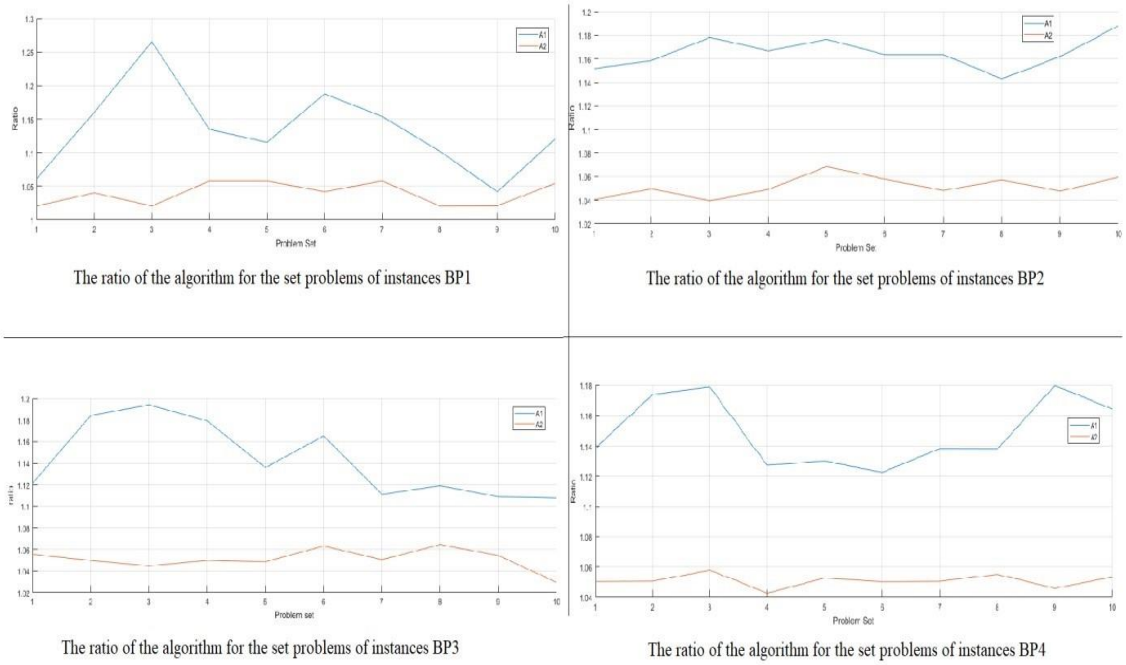

Figure 4.2: The ratio of the algorithm for the set problems of instances for bin packing

In this experiment, a modification was proposed for algorithm A1 to obtain a better performance, by matching the L items with S items to fill bins. But this modification fails, because it performed worse results. Both algorithms A1 and modified A1 are implemented and experimented on data set from OR Library as shown in the results in figure 4.3.

The problem sets vary from 10 to 1000, so as shown the bigger problem set, the better performance. But still has worse results although it shows an interesting results in type 4 sets. so the original algorithm will be considered in the comparison.

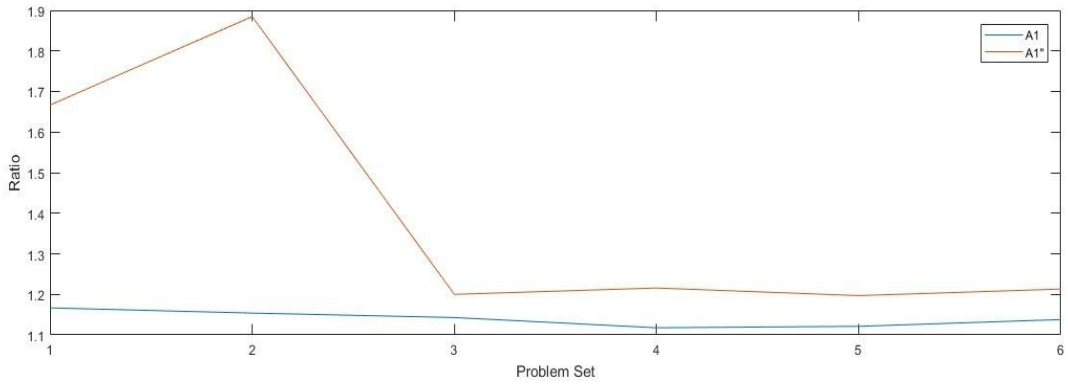

Figure 4.3: The Ratio of algorithms for the set problems for A1 modified A1

Due to randomness in A2, the following table 4.1 is statistics of a sample of the experimented problem sets run for 30 times. The bigger the problem set, the higher the spread of solution space while the standard deviation is getting bigger.

Table 4.1: Descriptive statistics of Zehmakan Approximation Algorithm 2

items	Optimal	Mean A2	Std. Deviation	Std. Error	95% CI Min	95% CI Max
120	50	50.1	0.32	0.101	49.87	50.33
250	99	103.9	0.57	0.18	103.49	104.31
500	198	208.5	0.34	0.108	207.73	209.27
1000	399	419.2	0.63	0.199	418.75	419.65
1000	395	414.2	0.96	0.304	413.54	414.25

The figure 4.4 shows the proposed algorithms A1 and A2 performance on several benchmark problems mentioned in section 4.1 (type 1, 2 and 3), the results of implementing A1 and A2 shown in table 5.2 in Appendix. Comparing them according to accuracy of results, A2 performs more efficiently than A1. The figure shows that the results are calculated on 15 problem sets arranged in an increasing order. In the problem sets with small number of items from 1 to 7, there's a difference in performance. But while the problem is getting bigger and more difficult, the performance of A1 becomes more efficient and close to the performance of A2.

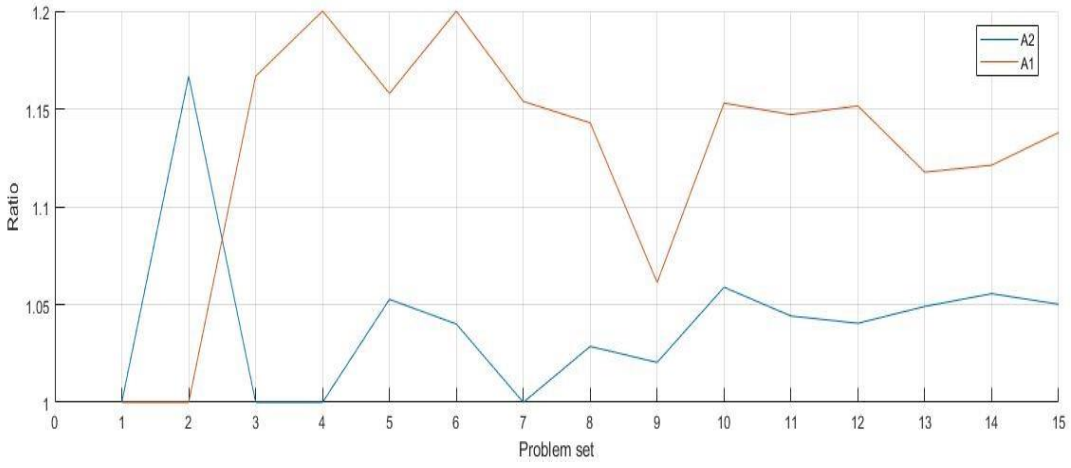

Figure 4.4: comparison between A1 and A2 by the number of bins

By comparing the classical approximation algorithms (FFD, BFD), both gave the same results (number of bins used) shown in table 5.2. So comparing them with A1 and A2 shows in figure 4.5, in small problem sets, the classical algorithms can reach the optimal solution, in larger problems the solution has a ratio of at most 1.01 which is very close to the optimal solution. Also, A2 is similar to FFD and BFD but with a ratio of at least 1.04 since A2 based on FFD. Otherwise, A1 gave the worst results with a ratio of at least 1.15.

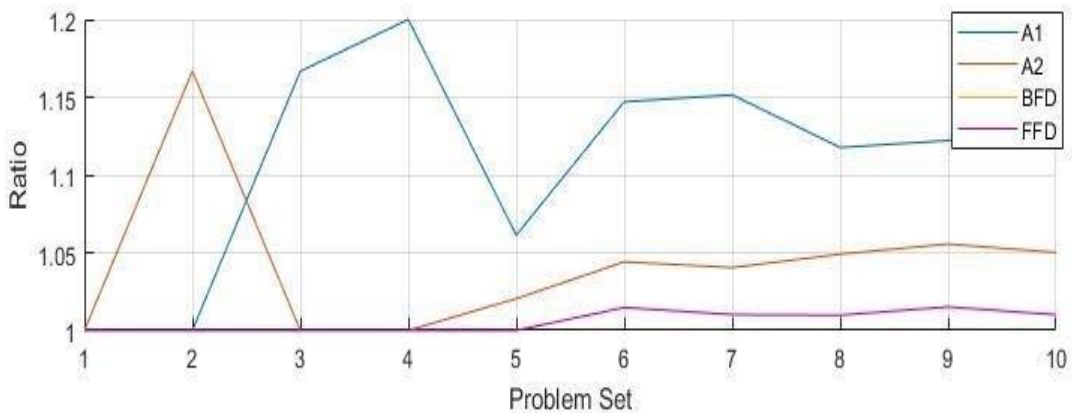

Figure 4.5: comparison between A1, A2, FFD and BFD by number of bins

So, table 5.3 shows a comparison between the four algorithms by running time. A1 is the fastest algorithm among the four algorithms with a best-case of O(1) while BFF, FFD shows a fine running time of O(n), and A2 shows a bad results O(nlogn) as shown in figure 4.6

A1 is not only the fastest algorithm but also it saves space because it requires all the inputs at the beginning of the algorithm and doesn't keep all inputs during the whole process(off-line algorithms).

But A2, FFD and BFD process the inputs one by one where it keeps all the inputs during processing (on-line algorithm) so they consume more space than A1.

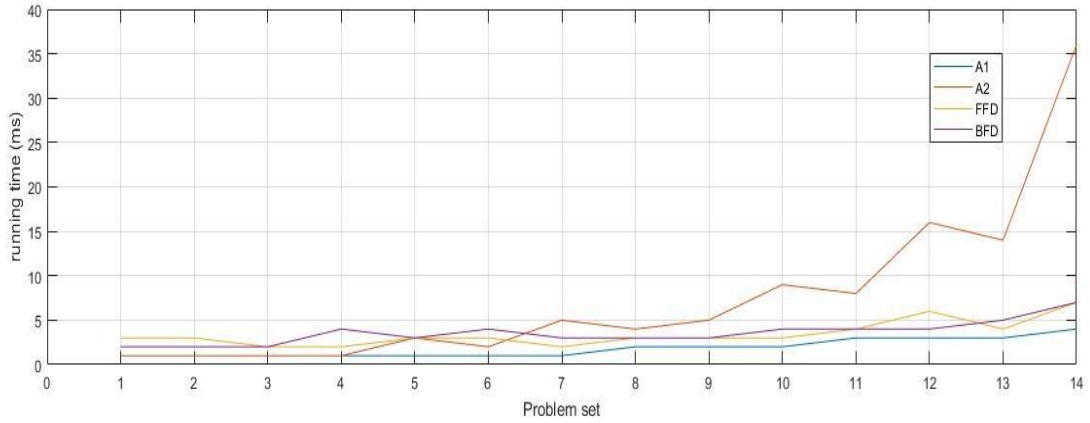

Figure 4.6: comparison between A1, A2, FFD and BFD by running time

4.3 Meta-heuristic algorithms comparison

Meta-heuristic algorithms are effective and efficient for bin packing problem, depending on choosing the right parameters. Both GA and PSO are re-implemented to solve BPP as described before in chapter three. In this section we will set the parameters and find results.

As mentioned before, the Grouping Genetic Algorithm is used in this work. The parameters are set as in Falkenauer (1997) with some modification set by experimenting different parameters as follows:

1. Mutation: 0.66

2. Mutation size: Mutation * Population size

3. Crossover probability: 0.4

4. Size of Offsprings: (crossover probability * Population size)/2

5. Number of iteration: 1000

6. Tournament selection round: 2

Population size have been tested for genetic algorithm for benchmark problem of 500 instances with optimal solution of 198 as shown in Table 4.2

Table 4.2: Sample of solutions for several population size for GA

Pop. size	Solutions
50	239 257 227 242 240
100	237 239 235 238 235
1000	235 237 232 237 241

Also for BPSO, as mentioned in Khanesar (2007), parameters are set same as the continuous PSO, the constriction coefficients c1 and c2 both are set to 2.05, which is useful to analyze the convergence. Also for PSO, population size has been tested on the same benchmark as shown in Table 4.3

Table 4.3: Sample of solutions for several population size for PSO

Pop. size	Solutions
50	239 232 227 242 230
100	237 240 235 238 236
1000	241 241 232 237 237

The larger the population size, the higher the time complexity. The population size 50 is the best for running time comparison. However, for the accuracy of the objective function results which is the minimum number of bins, the following statistics in Table 4.4 shows that the best

population size for both GA and PSO is 100 with Confidence interval between 0.92 and 7.08

Table 4.4: statistics on GA and PSO for different population size

Pop. size	GA mean (M1)	PSO mean (M2)	M1-M2	GA St. Dev.	PSO St. Dev	95% CI min	95% CI max
50	244.6	233.2	11.4	17.98	2.28	-11.03	33.83
100	236.8	232.8	4	2.39	1.39	0.92	7.08
200	236.4	236.6	-0.2	3.029	4.04	-5.57	5.17

To measure the efficiency of PSO and GA, both have been tested on the same benchmark problems that used in heuristic and the results are obtained in table 5.4 in appendix, figure 4.7 shows that PSO outperforms GA by comparing number of bins. At small to medium problem set but in large

sets ,problem sets with number of item of 250,500,1000, both algorithms give almost the same solution

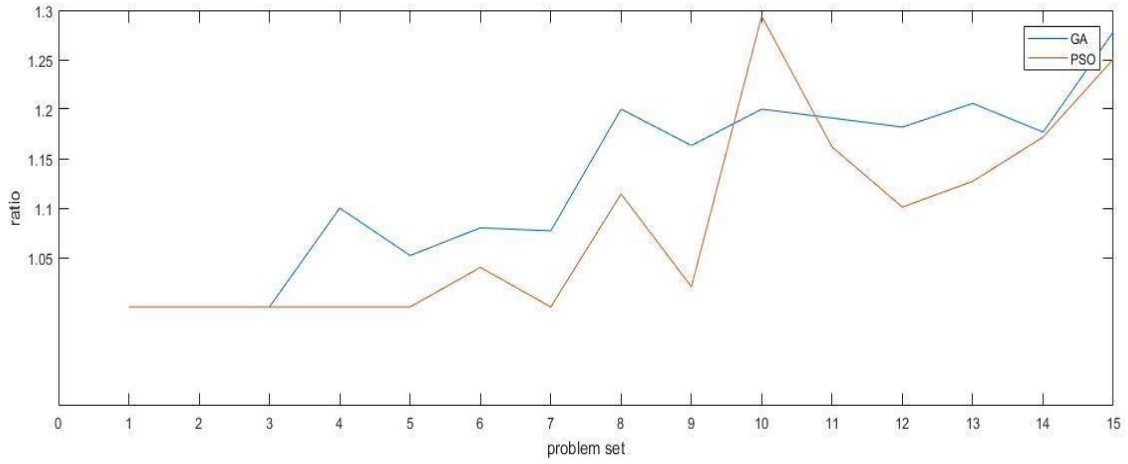

Figure 4.7: comparison between PSO and GA by number of bins

Comparing meta-heuristic to heuristics, it shows that heuristics performs much better than meta-heuristics by comparing both number of bins and

running time. since PSO outperformed GA in metaheuristics algorithms, and A2 outperformed A1 in heuristic algorithms. figure 4.8 shows a comparison between the performance of heuristic A2 and the performance of metaheuristic algorithm PSO.

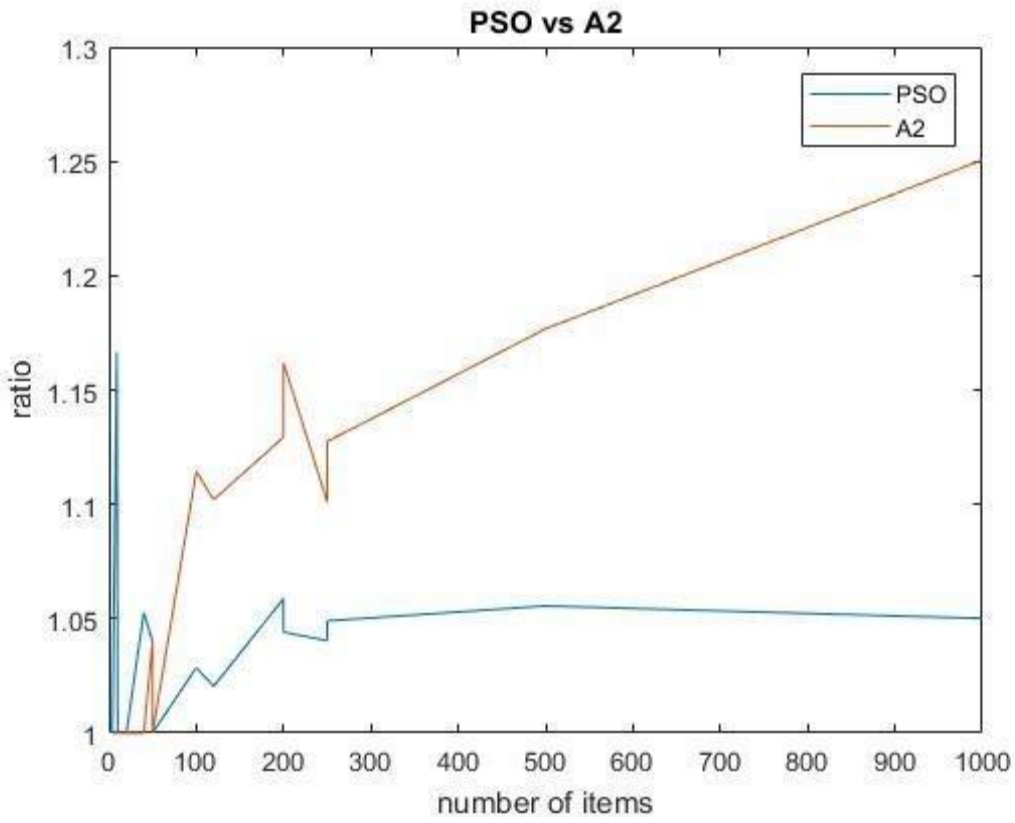

Figure 4.8: comparison between PSO and A2

Another experiment is set for a different benchmark problem type 5 on all algorithms where the implemented results are shown in table 5.6 in appendix. This comparison for medium size, easy problem sets, both BFF and FFD reaches the optimal solution among all the sets, GA and PSO have a close results but A2 outperform them, and A1 has the worse results comparing to all algorithms but still acceptable results for BPP shown in figure 4.9.

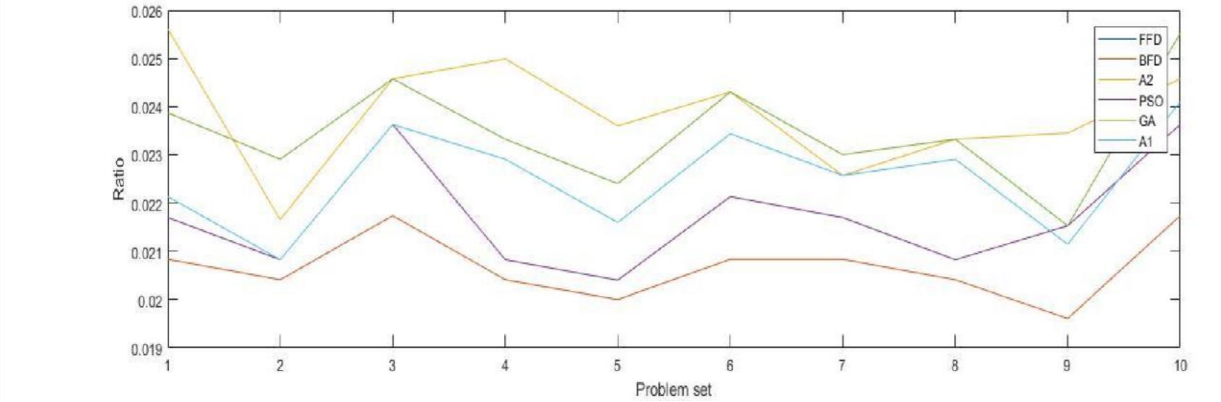

Figure 4.9: comparison between all algorithms by number of bins for easy datasets

By comparing running time for easy benchmark problems GA and PSO has a fine complexity of O(n) and A1, A1, BFD and FFD are showing a good complexity.

Also an experiment is done on hard problem set type 4 with 200 items and capacity of 100000 shown in Figure 4.10. A2 outperforms all the algorithms with interesting results very close to optimal solution, which the results are shown in appendix table 5.8, where all algorithms reach acceptable results except A1 is the worse with solutions far from optimal solution because the distribution of items are focused on S and M1 ranges, according to A2 algorithm M1 range is matching with M2 and itself, and S range is matching with M2 and itself, then M1 won’ t match with S leading to a big increase on number of the used bins.

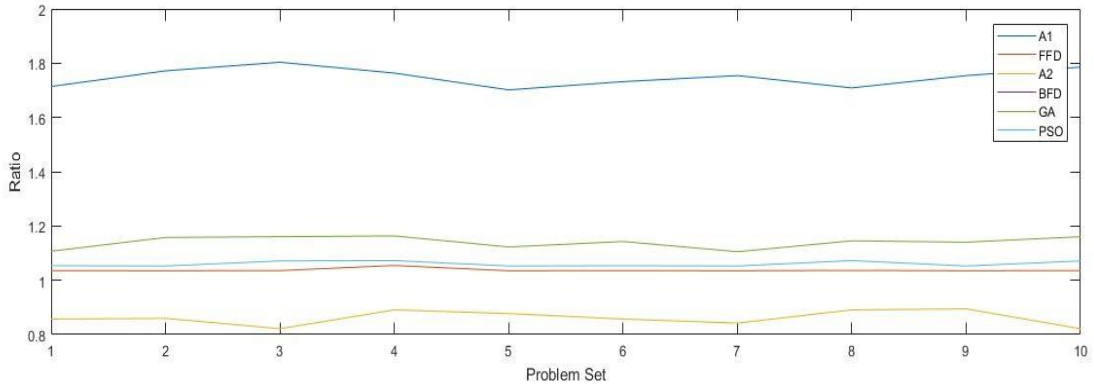

Figure 4.10: comparison between all algorithms by number of bins for hard datasets

By comparing the running time for hard data sets, the following figure 4.11 shows that both FFD and BFD have the best complexity time among all algorithms. Therefore A2 has a fine complexity time better than GA and PSO. A1 has the worst results due to the distribution of the items.

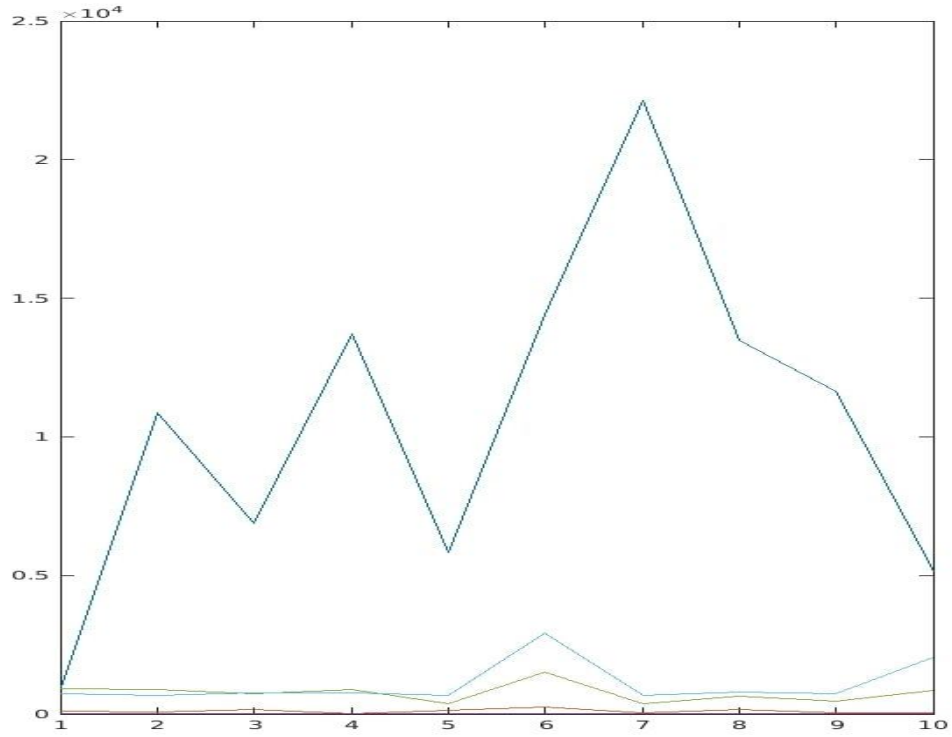

Figure 4.11: comparison between all algorithms by the running time for hard datasets

Chapter 5

Conclusion

In this thesis, we searched for optimal or near-optimal solution for Bin Packing Problem which aims to find the minimum number of bins used to pack items. This problem is very important in computing and science. Therefore, many researchers have proposed algorithms to reach the optimal solution as mentioned in the previous chapter.

Six algorithms were chosen to make a comparison based on the number of the used bins and running time to test the performance of these algorithms (A1, A2, BFD, FFD, GA and PSO). The question of this thesis is what is the best algorithm for BPP is worked around as follows:

1. Comparing heuristic algorithms by the number of bins used, A2 outperforms A1, but classical heuristics FFD and BFD outperform both of them, since they reach the optimal solution in all small to medium items size benchmark problem.
2. Comparing heuristic algorithm by the running time, all algorithms take milliseconds to reach a solution but A2 has a bad complexity time.
3. The experiments showed similarities between A1 and FFD by comparing number of bins.
4. Comparing metaheuristic algorithms by number of bins leads to two different results, on the first experiment PSO outperform GA while on the second one PSO and GA has a close results, PSO is slightly better than GA.

5. Comparing metaheuristic algorithm by running time, in some cases PSO took double GA running time in most cases.

6. Comparing the six algorithms shows that all the algorithm are efficient and effective, although meta-heuristic algorithms are good competitor for heuristic algorithm. In easy datasets classical heuristics outperform all algorithms, but in hard datasets A2 outperforms all algorithms with a small difference between all of them. according to number of bins.

This thesis highlighted two approaches for the same problem after comparing and analyzing the results of the experiments were done, the researchers who are interested in BPP will be able to choose the suitable approach for their work.

this thesis can be used as a base for promising researches in the future, for example, the heuristic and met-heuristic algorithms could be combined together to reach for better algorithms for BPP.

References

[1] Niclas Andreasson, Anton Evgrafov, and Michael Patriksson. An introduction to optimization: Foundations and fundamental algorithms. Chalmers University of Technology Press: Gothenburg, Sweden, 1:1–205, 2005.

[2] Rudolf Berghammer and Florian Reuter. A linear approximation algorithm for bin packing with absolute approximation factor 32. Science of Computer Programming, 48(1):67‑80, 2003.

[3] Omid Bozorg-Haddad, Mohammad Solgi, Hugo A Loi, et al. Meta-heuristic and evolutionary algorithms for engineering optimization. JohnWiley & Sons, 2017.

[4] Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms, part ii: hybrid genetic search strategies. Computers & Industrial Engineering, 36(2): 343‑364, 1999.

[5] Edwin KP Chong and Stanislaw H Zak. An introduction to optimization. John Wiley & Sons, 2004.

[6] Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. European Journal of Operational Research, 255(1):1‑20,53, 2016.

[7] Ugur Eliiyi and Deniz TUSEL ELIIYI. Applications of bin packing models through the supply chain. International Journal of Business and Management Studies, 1(1):11‑19, 2009.

[8] Emanuel Falkenauer. A hybrid grouping genetic algorithm for bin packing. Journal of heuristics, 2(1):5‑30, 1996.

[9] M Fallahi, S Amiri, and M Yaghini. A parameter tuning methodology for metaheuristics based on design of experiments. International Journal of Engineering and Technology Sciences, 2(6):497‑521, 2014.

[10] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In Foundations of genetic algorithms, volume 1, pages 69‑93. Elsevier, 1991.

[11] A Hanif Halim and I Ismail. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. Archives of Computational Methods in Engineering, 26(2):367‑380, 2019.

[12] John Henry Holland et al. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. 1975.

[13] James Kennedy and Russell Eberhart. Particle swarm optimization. In Proceedings of ICNN’95-International Conference on Neural Networks, volume 4, pages 1942‑1948. IEEE, 1995.

[14] Mojtaba Ahmadieh Khanesar, Mohammad Teshnehlab, and Mahdi Aliyari Shoorehdeli. A novel binary particle swarm optimization. pages 1‑6, 2007.

[15] Oliver Kramer. Self-adaptive heuristics for evolutionary computation, volume 147. Springer, 2008.

[16] Silvano Martello and Paolo Toth. Lower bounds and reduction procedures for the bin packing problem. Discrete applied mathematics, 28(1):59‑70, 1990.

[17] N Mohamadi. Application of genetic algorithm for the bin packing problem with a new representation scheme. 2010.

[18] Patricia Neri. Mathematical optimization in our daily lives. 2008.

[19] Zehmakanm Abdolahad Noori. Bin packing problem: Two approximation algorithms. ArXiv preprint arXiv: 1508. 01376, 2015.

[20] Diego Oliva, Mohamed Abd Elaziz, Ammar H Elsheikh, and Ahmed A Ewees. A review on metaheuristics methods for estimating parameters of solar cells. Journal of Power Sources, 435: 126683, 2019.

[21] Enrique Onieva. Artificial intelligence techniques at engineering semina. 2015.

[22] Bijaya Ketan Panigrahi, Ponnuthurai Nagaratnam Suganthan, Swagatam Das, and Shubhransu Sekhar Dash. Swarm, evolutionary, and memetic computing. In Proceedings of the Third International Conference SEMCCO, Chennai, India, pages 16‑18. Springer, 2010.

[23] R Gary Parker and Ronald L Rardin. Discrete optimization. Elsevier, 2014.

[24] RishinHalder Perumal and Rajkumar. Bin packing problems: Comparative analysis of heuristic techniques for different dimensions. 4OR, 8(2), 2016.

[25] Daniel Cosmin Porumbel. Heuristic algorithms and learning techniques: applications to the graph coloring problem. 4OR, 10(4):393-394, 2012.

[26] Franz Rothlauf. Representations for genetic and evolutionary algorithms. pages 9-30, 2002.

[27] SN Sivanandam and SN Deepa. Genetic algorithms. pages 15-37, 2008.

[28] Gilbert Syswerda. Uniform crossover in genetic algorithms. inproceed-ings of the third international conference on genetic algorithms. 1989.

[29] Roel van der Bles. A design tool for machinery space arrangement. 2019.

[30] Thomas Weise. Global optimization algorithms-theory and application. Self-Published Thomas Weise, 2009.

[31] Xin-She Yang and Xingshi He. Nature-inspired optimization algorithms in engineering: overview and applications. pages 1-20, 2016.

[32] R. Yesodha and T Amudha. Bio-inspired metaheuristics for bin packing problems. 11 2012.

[33] Mauricio Zambrano-Bigiarini, Maurice Clerc, and Rodrigo Rojas. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. pages 2337‑2344, 06 2013.

[34] Angel Eduardo Munoz Zavala, Arturo Hernandez Aguirre, and Enrique Raul Villa Diharce. Particle evolutionary swarm optimization algorithm (peso). In Sixth Mexican International Conference on Computer Science (ENC’ 05), pages 282‑289. IEEE, 2005.

[35] Zhi-hui Zhan and Jun Zhang. Adaptive particle swarm optimization. pages 227‑234, 2008.

[36] Guochuan Zhang, Xiaoqiang Cai, and CK Wong. Linear time-approximation algorithms for bin packing. Operations Research Letters, 26(5): 217‑222, 2000.

[37] Alan Zheng and Karl Schober. Genetic algorithms. 2018.

Appendix

Table 5.1: Comparison between A1 and modified A1 by the number of bins

Set No.	Items size	Bin capacity	Optimal solution	A1	Modified A1
1	10	20	6	7	10
2	50	120	26	30	49
3	100	150	35	40	42
4	120	150	49	52	65
5	200	120	85	98	101
6	250	150	99	114	124
7	500	150	198	222	240
8	500	150	205	226	237
9	1000	150	399	454	451
10	1000	150	395	414	421

Table 5.2: Comparison between A1, A2, BFD and FFD by the number of bins

Set No.	Set name	Items size	Bin capacity	Optimal solution	A1	A2	FFD	BFD
1	M1	4	6	2	2	2	2	2
2	M2	9	14	6	6	7	6	6
3	M3	10	20	6	7	6	6	6
4	M4	20	45	10	12	10	10	10
5	M5	40	70	19	22	20	19	19
6	N1C1W1_G	50	100	25	30	26	25	25
7	N1C2W1_B	50	120	26	30	26	26	26
8	N2C3W1_H	100	150	35	40	36	35	35
9	BP1	120	150	49	52	50	49	49
10	N3C2W1_D	200	120	85	98	90	98	98
11	N3C3W1_E	200	150	68	78	71	69	69
12	BP2	250	150	99	114	103	100	100
13	BP2	250	150	102	114	107	103	103
14	BP3	500	150	198	222	209	201	201
15	BP4	1000	150	399	454	419	403	403

Table 5.3: Comparison between A1, A2, FFD and BFD by running time

Set No.	Set Name	A1	A2	FFD	BFD
1	M1	1	1	3	2
2	M2	1	1	3	2
3	M3	1	1	2	2
4	M4	1	1	2	4
5	M5	1	3	3	3
6	N1C1W1_G	1	2	3	4
7	N1C2W1_B	1	5	2	3
8	N2C3W1_H	2	4	3	3
9	BP1	2	5	3	3
10	N3C2W1_D	2	9	3	4
11	N3C3W1_E	3	8	4	4
12	BP2	3	16	6	4
13	BP2	3	14	4	5
14	BP3	4	36	7	7
15	BP4	4	74	14	17

Table 5.4: Comparison between GA and PSO by the number of bins

Set No.	Set name	Items size	Bin capacity	Optimal solution	GA	PSO
1	M1	4	6	2	2	2
2	M2	9	14	6	6	6
3	M3	10	20	6	6	6
4	M4	20	45	10	11	10
5	M5	40	70	19	21	19
6	N1C1W1_G	50	100	25	27	26
7	N1C2W1_B	50	120	26	28	26
8	N2C3W1_H	100	150	35	42	39
9	BP1	120	150	49	57	54
10	N3C2W1_D	200	120	85	102	96
11	N3C3W1_E	200	150	68	81	79
12	BP2	250	150	99	117	109
13	BP2	250	150	102	123	115
14	BP3	500	150	198	233	232
15	BP4	1000	150	399	510	499

Table 5.5 : comparison between GA and PSO by running time

Set No.	Set name	GA	PSO
1	M1	10.8 s	16.9 s
2	M2	11.7 s	22.5 s
3	M3	12.5 s	23.9 s
4	M4	12.8 s	30.7 s
5	M5	16.7 s	43.5 s
6	N1C1W1_G	27.4 s	50 s
7	N1C2W1_B	24.6 s	51.4 s
8	N2C3W1_H	26.7 s	85.2 s
9	BP1	39.3 s	100.3 s
10	N3C2W1_D	56.2 s	162.6 s
11	N3C3W1_E	47.1 s	199.4 s
12	BP2	62.3 s	197.5 s
13	BP2	62.8 s	194.2 s
14	BP3	141.9 s	483 s
15	BP4	306.8 s	1118 s

Table 5.6: Comparison between all algorithms by the number of bins for easy datasets

Set No.	Set Name	Optimal solution	A1	A2	FFD	BFD	GA	PSO
	u120_01	48	59	50	48	48	55	51
	u120_02	49	52	50	49	49	55	50
	u120_03	46	52	50	46	46	52	50
	u120_04	49	60	50	49	49	56	55
	u120_05	50	59	51	50	50	56	54
	u120_06	48	56	51	48	48	56	54
	u120_07	48	52	50	48	48	53	52
	u120_08	49	56	50	49	49	56	55
	u120_09	51	61	56	51	51	56	55
	u120_10	46	52	50	46	46	54	51

Table 5.7: comparison between all algorithms by the running time for easy datasets

Set No.	Set Name	Optimal solution	A1 (ms)	A2 (ms)	BFD (ms)	FFD (ms)	GA (s)	PSO (s)
1	u120_01	48	1	5	2	3	183	154
2	u120_02	49	2	5	3	2	162	133
3	u120_03	46	1	4	3	2	155	134

4	u120_04	49	1	6	2	2	174	151
5	u120_05	50	3	3	4	2	196	160
6	u120_06	48	2	3	3	3	194	161
7	u120_07	48	3	5	3	3	189	171
8	u120_08	49	2	5	4	2	300	238
9	u120_09	51	1	6	2	4	185	148
10	u120_10	46	2	4	2	3	320	290

Table 5.8: Comparison between all algorithms by the number of bins for hard datasets

Set No.	Set Name	Optimal solution	A1	A2	FFD	BFD	GA	PSO
1	hard0	56	96	58	59	59	62	63
2	hard1	57	101	59	60	60	66	63
3	hard2	56	101	58	60	60	65	64
4	hard3	55	97	58	59	59	64	62
5	hard4	57	97	59	60	60	64	64
6	hard5	56	97	58	59	59	64	62
7	hard6	57	100	59	60	60	63	62
8	hard7	55	94	57	59	59	63	62
9	hard8	57	100	59	60	60	65	64
10	hard9	56	100	58	60	60	65	63

Table 5.9: comparison between all algorithms by the running time for hard datasets

Set No.	Set Name	Optimal solution	A1 (ms)	A2 (ms)	BFD (ms)	FFD (ms)	GA (s)	PSO (s)
1	hard0	56	985	98	1	1	916	745
2	hard1	57	10852	60	1	1	897	660
3	hard2	56	6894	154	1	1	726	761
4	hard3	55	13716	27	3	1	871	754
5	hard4	57	5833	125	4	4	375	689
6	hard5	56	14396	255	12	7	1516	2933
7	hard6	57	22141	30	1	2	360	684
8	hard7	55	13463	158	3	4	638	785
9	hard8	57	11652	32	1	1	475	725
10	hard9	56	5152	56	2	1	865	2041

Datasets

Set M1

N = 4, C = 6

2, 2, 4, 4

Set M2

N = 9, C = 14

5, 7, 3, 5, 12, 11, 10, 11, 9

Set M3

N = 10, C = 20

14, 15, 12, 2, 4, 8, 13, 19, 20, 7

Set M4

N = 20, C = 45

17, 19, 12, 11, 17, 18, 17, 4, 5, 21, 10, 23, 37, 32, 29, 40, 41, 30, 21, 11

Set M5

N = 40, C = 70

12, 13, 11, 40, 22, 60, 61, 63, 11, 10, 19, 31, 32, 37, 25, 14, 21, 38, 51, 59, 40, 45, 54, 62, 59, 40, 13, 31, 17, 20, 26, 36, 15, 12, 9, 10, 27, 31, 55, 40

Set N1C1W1_ G

N = 50, C = 100

99, 99, 96, 96, 92, 92, 91, 88, 87, 86, 85, 76, 74, 72, 69, 67, 67, 62, 61, 56, 52, 51, 49, 46, 44, 42, 40, 40, 33, 33, 30, 30, 29, 28, 28, 27, 25, 24, 23, 22, 21, 20, 17, 14, 13, 11, 10, 7, 7, 3

Set 7 N1C1W1_ B

N = 50, C = 120

99, 96, 96, 96, 95, 95, 94, 90, 90, 88, 87, 84, 82, 78, 77, 77, 77, 75, 75, 70, 70, 69, 68, 56, 54, 53, 53, 50, 50, 49, 48, 47, 45, 38, 36, 35, 34, 28, 25, 21, 19, 18, 16, 13, 13, 7, 7, 6, 3, 3

Set N1C1W1_ H

N = 100, C = 150

98, 97, 97, 97, 96, 95, 95, 95, 95, 93, 92, 88, 87, 86, 86, 85, 81, 81, 80, 78, 78, 78, 77, 77, 76, 75, 74, 72, 71, 70, 70, 69, 69, 67, 67, 67, 65, 65, 65, 64, 64, 63, 62, 58, 58, 56, 56, 56, 55, 52, 51, 50, 50, 50, 49, 49, 47, 45, 43, 43, 43, 42, 41, 40, 40, 40, 39, 38, **36, 35, 33, 33, 32, 30, 29, 28, 28, 25, 25, 22, 22, 20, 20, 18, 17, 16, 15, 11, 11, 10, 8, 5, 5, 5, 4, 4, 2, 2, 2, 1**

Set BP1

N = 120, C = 150

97, 57, 81, 62, 75, 81, 23, 43, 50, 38, 60, 58, 70, 88, 36, 90, 37, 45, 45, 39, 44, 53, 70, 24, 82, 81, 47, 97, 35, 65, 74, 68, 49, 55, 52, 94, 95, 29, 99, 20, 22, 25, 49, 46, 98, 59, 98, 60, 23, 72, 33, 98, 80, 95, 78, 57, 67, 53, 47, 53, 36, 38, 92, 30, 80, 32, 97, 39, 80, 72, 55, 41, 60, 67, 53, 65, 95, 20, 66, 78, 98, 47, 100, 85, 53, 53, 67, 27, 22, 61, 43, 52, 76, 64, 61, 29, 30, 46, 79, 66, 27, 79, 98, 90, 22, 75, 57, 67, 36, 70, 99, 48, 43, 45, 71, 100, 88, 48, 27, 39

Set N3C2W1_ D

N = 200, C = 120

100, 100, 100, 99, 99, 98, 98, 98, 97, 96, 95, 95, 95, 94, 94, 93, 93, 93, 93, 92, 92, 92, 91, 90, 90, 89, 89, 88, 87, 86, 86, 85, 85, 84, 84, 84, 83, 83, 83, 83, 81, 79, 78, 78, 77, 77, 76, 76, 75, 75, 75, 75, 75, 74, 74, 74, 74, 74, 73, 73, 73, 72, 71, 71, 70, 69, 69, 68, 68, 66, 65, 65, 65, 65, 65, 64, 64, 63, 61, 61, 61, 61, 60, 60, 60, 60, 60, 59, 59, 58, 58, 57, 57, 56, 55, 54, 53, 53, 52, 51, 51, 51, 50, 49, 48, 47, 46, 46, 45, 44, 44, 43, 41, 41, 39, 39, 38, 38, 38, 37, 37, 37, 36, 36, 35, 35, 35, 34, 34, 34, 34, 34, 33, 32, 32, 32, 31, 29, 28, 28, 28, 27, 27, 26, 25, 25, 23, 23, 23, 23, 23, 22, 22, 22, 22, 21, 20, 18, 18, 17, 17, 17, 16, 16, 15, 15, 14, 13, 13, 12, 12, 12, 11, 11, 11, 11, 11, 10, 8, 8, 8, 8, 8, 6, 6, 6, 6, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1, 1, 1

Set N3C3W1_ E

N = 200, C = 150

100, 100, 100, 99, 99, 99, 98, 98, 98, 98, 97, 97, 97, 97, 95, 95, 94, 94, 93, 93, 92, 92, 91, 91, 90, 90, 90, 90, 89, 89, 89, 89, 88, 88, 87, 86, 85, 84, 84, 84, 84, 83, 83, 82, 82, 82, 82, 81, 80, 79, 78, 78, 77, 76, 76, 75, 74, 74, 74, 73, 72, 71, 71,

70, 70, 70, 70, 70, 70, 69, 69, 68, 68, 68, 67, 66, 65, 64, 64, 63, 63, 62, 62, 61, 60, 59, 57, 57, 57, 56, 55, 55, 55, 55, 54, 54, 53, 53, 52, 52, 52, 52, 50, 48, 48, 48, 47, 47, 46, 46, 45, 45, 44, 44, 43, 43, 43, 42, 42, 42, 42, 41, 41, 40, 40, 39, 39, 36, 35, 34, 33, 32, 32, 31, 30, 29, 29, 28, 28, 27, 27, 24, 24, 24, 24, 23, 23, 23, 23, 23, 23, 21, 21, 20, 20, 19, 19, 18, 17, 17, 17, 16, 16, 15, 15, 15, 15, 14, 14, 13, 13, 13, 12, 12, 12, 12, 11, 11, 11, 10, 10, 9, 9, 8, 8, 8, 7, 7, 7, 7, 6, 5, 4, 4, 3, 3, 1, 1, 1, 1

Set BP2

N = 250, C = 150

42, 69, 67, 57, 93, 90, 38, 36, 45, 42, 33, 79, 27, 57, 44, 84, 86, 92, 46, 38, 85, 33, 82, 73, 49, 70, 59, 23, 57, 72, 74, 69, 33, 42, 28, 46, 30, 64, 29, 74, 41, 49, 55, 98, 80, 32, 25, 38, 82, 30, 35, 39, 57, 84, 62, 50, 55, 27, 30, 36, 20, 78, 47, 26, 45, 41, 58, 98, 91, 96, 73, 84, 37, 93, 91, 43, 73, 85, 81, 79, 71, 80, 76, 83, 41, 78, 70, 23, 42, 87, 43, 84, 60, 55, 49, 78, 73, 62, 36, 44, 94, 69, 32, 96, 70, 84, 58, 78, 25, 80, 58, 66, 83, 24, 98, 60, 42, 43, 43, 39, 97, 57, 81, 62, 75, 81, 23, 43, 50, 38, 60, 58, 70, 88, 36, 90, 37, 45, 45, 39, 44, 53, 70, 24, 82, 81, 47, 97, 35, 65, 74, 68, 49, 55, 52, 94, 95, 29, 99, 20, 22, 25, 49, 46, 98, 59, 98, 60, 23, 72, 33, 98, 80, 95, 78, 57,

67, 53, 47, 53, 36, 38, 92, 30, 80, 32, 97, 39, 80, 72, 55, 41, 60, 67, 53, 65, 95, 20, 66, 78, 98, 47, 100, 85, 53, 53, 67, 27, 22, 61, 43, 52, 76, 64, 61, 29, 30, 46, 79, 66, 27, 79, 98, 90, 22, 75, 57, 67, 36, 70, 99, 48, 43, 45, 71, 100, 88, 48, 27, 39, 38, 100, 60, 42, 20, 69, 24, 23, 92, 32

Set BP2

N = 250, C = 150

64, 42, 86, 65, 47, 68, 20, 45, 69, 78, 44, 96, 50, 27, 58, 55, 81, 87, 76, 38, 79, 71, 60, 76, 91, 69, 77, 57, 33, 22, 76, 51, 66, 90, 34, 46, 74, 62, 93, 74, 29, 22, 73, 26, 72, 41, 91, 88, 95, 35, 84, 32, 59, 56, 84, 71, 78, 82, 78, 52, 71, 26, 66, 84, 76, 95, 80, 50, 53, 30, 82, 38, 45, 99, 51, 98, 100, 88, 81, 77, 99, 97, 31, 54, 47, 45, 36, 96, 96, 74, 77, 98, 69, 22, 40, 39, 81, 90, 73, 84, 53, 73, 81, 51, 38, 43, 64, 28, 83, 28, 66, 22, 56, 61, 72, 69, 55, 20, 50, 52, 95, 89, 32, 60, 29, 90, 20, 90, 41, 37, 95, 20, 84, 33, 28, 40, 91, 39, 63, 66, 29, 74, 97, 41, 81, 53, 22, 32, 91, 61, 33, 91, 55, 56, 57, 44, 60, 55, 92, 39, 38, 100, 30, 65, 22, 78, 84, 32, 51, 52, 47, 62, 63, 25, 42, 59, 24, 88, 61, 71, 23, 48, 78, 85, 92, 39, 31, 76, 87, 54, 61, 66, 40, 22, 74, 99, 96, 73, 24, 43, 93, 47, 51, 22, 49, 39, 21, 72, 93, 72, 49, 68, 71, 82, 44, 25, 82, 74, 59, 28, 33, 61, 90, 97,

62, 42, 100, 50, 31, 84, 81, 27, 45, 84, 54, 34, 79, 100, 63, 48, 68, 46, 74, 65, 35, 66, 53, 27, 70, 86

Set BP3

N = 500, C = 150

42, 69, 67, 57, 93, 90, 38, 36, 45, 42, 33, 79, 27, 57, 44, 84, 86, 92, 46, 38, 85, 33, 82, 73, 49, 70, 59, 23, 57, 72, 74, 69, 33, 42, 28, 46, 30, 64, 29, 74, 41, 49, 55, 98, 80, 32, 25, 38, 82, 30, 35, 39, 57, 84, 62, 50, 55, 27, 30, 36, 20, 78, 47, 26, 45, 41, 58, 98, 91, 96, 73, 84, 37, 93, 91, 43, 73, 85, 81, 79, 71, 80, 76, 83, 41, 78, 70, 23, 42, 87, 43, 84, 60, 55, 49, 78, 73, 62, 36, 44, 94, 69, 32, 96, 70, 84, 58, 78, 25, 80, 58, 66, 83, 24, 98, 60, 42, 43, 43, 39, 97, 57, 81, 62, 75, 81, 23, 43, 50, 38, 60, 58, 70, 88, 36, 90, 37, 45, 45, 39, 44, 53, 70, 24, 82, 81, 47, 97, 35, 65, 74, 68, 49, 55, 52, 94, 95, 29, 99, 20, 22, 25, 49, 46, 98, 59, 98, 60, 23, 72, 33, 98, 80, 95, 78, 57, 67, 53, 47, 53, 36, 38, 92, 30, 80, 32, 97, 39, 80, 72, 55, 41, 60, 67, 53, 65, 95, 20, 66, 78, 98, 47, 100, 85, 53, 53, 67, 27, 22, 61, 43, 52, 76, 64, 61, 29, 30, 46, 79, 66, 27, 79, 98, 90, 22, 75, 57, 67, 36, 70, 99, 48, 43, 45, 71, 100, 88, 48, 27, 39, 38, 100, 60, 42, 20, 69, 24, 23, 92, 32, 84, 36, 65, 84, 34, 68, 64, 33, 69, 27, 47, 21, 85, 88, 59, 61, 50, 53, 37, 75, 64, 84, 74, 57, 83, 28, 31, 97, 61, 36, 46, 37, 96, 80, 53, 51, 68, 90, 64, 81, 66, 67, 80, 37, 92, 67, 64, 31, 94, 45, 80, 28, 76, 29, 64, 38, 48, 40, 29, 44, 81, 35, 51, 48, 67, 24, 46, 38, 76, 22, 30, 67, 45, 41, 29, 41, 79, 21, 25, 90, 62, 34, 73, 50, 79, 66,

59, 42, 90, 79, 70, 66, 80, 35, 62, 98, 97, 37, 32, 75, 91, 91, 48, 26, 23, 32, 100, 46, 29, 26, 29, 26, 83, 82, 92, 95, 87, 63, 57, 100, 63, 65, 81, 46, 42, 95, 90, 80, 53, 27, 84, 40, 22, 97, 20, 73, 63, 95, 46, 42, 47, 40, 26, 88, 49, 24, 92, 87, 68, 95, 34, 82, 84, 43, 54, 73, 66, 32, 62, 48, 99, 90, 86, 28, 25, 25, 89, 67, 96, 35, 33, 70, 40, 59, 32, 94, 34, 86, 35, 45, 25, 76, 80, 42, 91, 44, 91, 97, 60, 29, 45, 37, 61, 54, 78, 56, 74, 74, 45, 21, 96, 37, 75, 100, 58, 84, 85, 56, 54, 71, 52, 79, 43, 35, 27, 70, 31, 47, 35, 26, 30, 97, 90, 80, 58, 60, 73, 46, 71, 39, 42, 98, 27, 21, 71, 71, 78, 76, 57, 24, 91, 84, 35, 25, 77, 96, 97, 89, 30, 86

Set BP4

N = 1000, C = 150

42, 69, 67, 57, 93, 90, 38, 36, 45, 42, 33, 79, 27, 57, 44, 84, 86, 92, 46, 38, 85, 33, 82, 73, 49, 70, 59, 23, 57, 72, 74, 69, 33, 42, 28, 46, 30, 64, 29, 74, 41, 49, 55, 98, 80, 32, 25, 38, 82, 30, 35, 39, 57, 84, 62, 50, 55, 27, 30, 36, 20, 78, 47, 26, 45, 41, 58, 98, 91, 96, 73, 84, 37, 93, 91, 43, 73, 85, 81, 79, 71, 80, 76, 83, 41, 78, 70, 23, 42, 87, 43, 84, 60, 55, 49, 78, 73, 62, 36, 44, 94, 69, 32, 96, 70, 84, 58, 78, 25, 80, 58, 66, 83, 24, 98, 60, 42, 43, 43, 39, 97, 57, 81, 62, 75, 81, 23, 43, 50, 38, 60, 58, 70, 88, 36, 90, 37, 45, 45, 39, 44, 53, 70, 24, 82, 81, 47, 97, 35, 65, 74, 68, 49, 55, 52, 94, 95, 29, 99, 20,

22, 25, 49, 46, 98, 59, 98, 60, 23, 72, 33, 98, 80, 95, 78, 57, 67, 53, 47, 53, 36, 38, 92, 30, 80, 32, 97, 39, 80, 72, 55, 41, 60, 67, 53, 65, 95, 20, 66, 78, 98, 47, 100, 85, 53, 53, 67, 27, 22, 61, 43, 52, 76, 64, 61, 29, 30, 46, 79, 66, 27, 79, 98, 90, 22, 75, 57, 67, 36, 70, 99, 48, 43, 45, 71, 100, 88, 48, 27, 39, 38, 100, 60, 42, 20, 69, 24, 23, 92, 32, 84, 36, 65, 84, 34, 68, 64, 33, 69, 27, 47, 21, 85, 88, 59, 61, 50, 53, 37, 75, 64, 84, 74, 57, 83, 28, 31, 97, 61, 36, 46, 37, 96, 80, 53, 51, 68, 90, 64, 81, 66, 67, 80, 37, 92, 67, 64, 31, 94, 45, 80, 28, 76, 29, 64, 38, 48, 40, 29, 44, 81, 35, 51, 48, 67, 24, 46, 38, 76, 22, 30, 67, 45, 41, 29, 41, 79, 21, 25, 90, 62, 34, 73, 50, 79, 66, 59, 42, 90, 79, 70, 66, 80, 35, 62, 98, 97, 37, 32, 75, 91, 91, 48, 26, 23, 32, 100, 46, 29, 26, 29, 26, 83, 82, 92, 95, 87, 63, 57, 100, 63, 65, 81, 46, 42, 95, 90, 80, 53, 27, 84, 40, 22, 97, 20, 73, 63, 95, 46, 42, 47, 40, 26, 88, 49, 24, 92, 87, 68, 95, 34, 82, 84, 43, 54, 73, 66, 32, 62, 48, 99, 90, 86, 28, 25, 25, 89, 67, 96, 35, 33, 70, 40, 59, 32, 94, 34, 86, 35, 45, 25, 76, 80, 42, 91, 44, 91, 97, 60, 29, 45, 37, 61, 54, 78, 56, 74, 74, 45, 21, 96, 37, 75, 100, 58, 84, 85, 56, 54, 71, 52, 79, 43, 35, 27, 70, 31, 47, 35, 26, 30, 97, 90, 80, 58, 60, 73, 46, 71, 39, 42, 98, 27, 21, 71, 71, 78, 76, 57, 24, 91, 84, 35, 25, 77, 96, 97, 89, 30, 86, 81, 39, 75, 66, 85, 36, 60, 56, 50, 75, 75, 37, 87, 95, 21, 99, 42, 57, 31, 37, 42, 40, 69, 91, 45, 97, 84, 90, 52, 43, 68, 53, 37, 65, 79, 73, 92, 87, 20, 20, 73, 42, 52, 20,

24, 76, 71, 72, 21, 21, 82, 92, 78, 87, 50, 41, 31, 73, 89, 59, 88, 40, 71, 69, 45, 57, 49, 68, 84, 32, 69, 77, 92, 98, 57, 39, 32, 23, 99, 91, 48, 21, 70, 43, 73, 69, 65, 57, 67, 28, 84, 42, 61, 92, 82, 34, 74, 55, 60, 69, 26, 25, 67, 77, 67, 79, 47, 84, 50, 21, 87, 83, 44, 88, 78, 53, 78, 37, 47, 52, 32, 88, 85, 82, 55, 41, 60, 66, 78, 72, 34, 64, 20, 60, 100, 62, 80, 34, 68, 38, 32, 32, 37, 82, 98, 90, 58, 97, 56, 34, 70, 39, 56, 69, 36, 20, 99, 84, 53, 27, 88, 53, 42, 45, 42, 31, 54, 60, 55, 27, 36, 31, 39, 91, 45, 97, 26, 80, 41, 56, 70, 97, 48, 87, 23, 32, 75, 100, 97, 51, 78, 78, 21, 72, 72, 79, 46, 30, 48, 27, 95, 48, 67, 58, 46, 92, 21, 82, 91, 40, 56, 24, 94, 44, 91, 92, 81, 24, 84, 44, 83, 37, 98, 85, 88, 95, 29, 35, 100, 55, 48, 27, 20, 66, 62, 52, 88, 59, 97, 91, 81, 81, 86, 48, 43, 60, 72, 88, 90, 48, 38, 60, 53, 55, 90, 48, 55, 57, 59, 25, 51, 22, 43, 31, 52, 89, 96, 58, 63, 27, 46, 43, 30, 44, 71, 66, 64, 28, 83, 88, 42, 92, 95, 36, 24, 62, 44, 82, 59, 31, 96, 44, 61, 78, 72, 62, 76, 65, 22, 41, 27, 85, 80, 72, 100, 29, 27, 43, 83, 32, 33, 53, 95, 99, 20, 23, 72, 50, 50, 27, 89, 53, 75, 81, 34, 27, 69, 48, 84, 37, 69, 54, 51, 49, 49, 54, 100, 55, 45, 83, 61, 96, 91, 37, 53, 76, 50, 66, 70, 87, 92, 35, 53, 95, 47, 56, 55, 86, 32, 99, 83, 88, 41, 63, 77, 60, 66, 53, 79, 81, 96, 34, 99, 47, 74, 87, 44, 77, 52, 99, 69, 64, 94, 38, 69, 61, 98, 40, 84, 89, 49, 64, 53, 41, 34, 85, 35, 55, 61, 68, 100, 75, 98, 36, 44, 57, 24, 60, 45, 48, 60, 94, 71, 70, 64, 62, 93, 20, 69, 37, 63, 61, 26, 54, 89, 46, 54, 50,

32, 71, 62, 40, 26, 59, 62, 27, 60, 50, 74, 34, 40, 70, 56, 23, 66, 57, 43, 45, 65, 25, 82, 82, 37, 66, 47, 44, 94, 23, 24, 51, 100, 22, 25, 51, 95, 58, 97, 30, 79, 23, 53, 80, 20, 65, 64, 21, 26, 100, 81, 98, 70, 85, 92, 97, 86, 71, 91, 29, 63, 34, 67, 23, 33, 89, 94, 47, 100, 37, 40, 58

HARD0

34978, 34849, 34703, 34608, 34598, 34524, 34356, 34308, 34069, 34049, 33895, 33842, 33806, 33738, 33716, 33590, 33546, 33507, 33468, 33465, 33383, 33190, 33075, 32976, 32897, 32762, 32696, 32638, 32553, 32398, 32230, 32176, 31967, 31954, 31903, 31782, 31724, 31686, 31597, 31561, 31532, 31499, 31346, 30943, 30915, 30869, 30766, 30683, 30678, 30644, 30559, 30448, 30315, 30238, 30125, 29974, 29947, 29890, 29886, 29858, 29856, 29783, 29697, 29438, 29427, 29301, 29174, 29173, 29123, 29117, 29116, 29095, 29094, 29063, 29041, 29038, 28977, 28946, 28921, 28910, 28842, 28703, 28360, 28350, 28305, 28302, 28225, 28160, 28094, 28040, 28020, 27901, 27775, 27765, 27688, 27439, 27425, 27394, 27365, 27349, 27284, 27180, 26935, 26881, 26867, 26795, 26703, 26651, 26550, 26432, 26375, 26368, 26244, 26204, 26192, 26181, 26158, 26133, 26067, 25945, 25906, 25759, 25698, 25688, 25652, 25615, 25530, 25528, 25366, 25324, 25273, 25142, 24852, 24846, 24658, 24592, 24564, 24463, 24457, 24374, 24359, 24332, 23987, 23956,

23952, 23932, 23895, 23837, 23795, 23774, 23663, 23621, 23502, 23453, 23430, 23366, 23178, 23090, 22991, 22942, 22743, 22442, 22432, 22415, 22338, 22134, 22081, 22014, 21950, 21948, 21796, 21784, 21727, 21722, 21557, 21498, 21480, 21315, 21193, 21127, 21060, 20997, 20837, 20813, 20693, 20693, 20686, 20677, 20676, 20664, 20663, 20634, 20616, 20570, 20566, 20496, 20441, 20307, 20226, 20114

HARD1

34991, 34949, 34847, 34577, 34461, 34343, 34318, 34316, 34302, 34290, 34282, 34279, 34046, 33944, 33814, 33813, 33753, 33653, 33620, 33584, 33554, 33544, 33426, 33414, 33376, 33273, 33270, 33170, 33034, 33007, 32957, 32897, 32784, 32773, 32528, 32499, 32423, 32400, 32356, 32302, 32090, 31863, 31850, 31841, 31840, 31775, 31773, 31655, 31613, 31608, 31587, 31535, 31378, 31197, 31194, 31179, 30992, 30899, 30780, 30742, 30685, 30645, 30641, 30610, 30498, 30336, 30327, 30271, 30105, 29975, 29957, 29924, 29870, 29815, 29777, 29754, 29658, 29648, 29553, 29481, 29416, 29415, 29410, 29408, 29361, 29316, 29002, 28987, 28947, 28897, 28801, 28636, 28538, 28507, 28435, 28360, 28330, 28063, 28007, 27983, 27937, 27879, 27760, 27715, 27517, 27230, 27146, 27072, 27028, 26985, 26894, 26840,

26799, 26797, 26717, 26582, 26511, 26472, 26469, 26386, 26301, 26117, 26110, 26031, 26030, 25705, 25532, 25524, 25499, 25441, 25421, 25356, 25310, 25227, 25118, 25073, 24989, 24955, 24844, 24792, 24625, 24562, 24526, 24451, 24299, 24290, 23927, 23885, 23873, 23850, 23795, 23583, 23473, 23438, 23408, 23354, 23328, 23260, 23145, 23128, 22994, 22744, 22687, 22596, 22581, 22516, 22467, 22412, 22337, 22253, 22226, 22206, 22177, 22036, 21997, 21933, 21807, 21749, 21669, 21656, 21585, 21525, 21506, 21437, 21415, 21316, 21222, 21214, 21098, 20944, 20819, 20718, 20488, 20458, 20422, 20324, 20233, 20137, 20008

HARD2

34953, 34942, 34849, 34732, 34683, 34640, 34590, 34446, 34315, 34314, 34236, 34088, 34060, 33942, 33861, 33858, 33811, 33800, 33764, 33725, 33709, 33475, 33415, 33402, 33367, 33286, 33280, 33093, 33083, 33047, 33005, 32966, 32931, 32906, 32787, 32731, 32716, 32708, 32670, 32651, 32621, 32560, 32555, 32544, 32387, 32363, 32186, 32143, 32094, 32072, 31982, 31912, 31830, 31759, 31646, 31641, 31548, 31505, 31411, 31408, 31383, 31192, 31155, 31153, 31083, 30955, 30726, 30648, 30531, 30532, 30369, 30250, 30226, 30165, 30111, 29999, 29973, 29899, 29787, 29512,

29509, 29501, 29429, 28933, 28887, 28882, 28849, 28841, 28823, 28595, 28497, 28486, 28399, 28269, 28099, 28021, 28006, 27873, 27850, 27672, 27670, 27607, 27402, 27317, 27290, 27211, 27163, 27104, 27052, 27012, 26866, 26786, 26656, 26598, 26477, 26474, 26470, 26411, 26397, 26352, 26176, 26155, 26076, 26019, 25983, 25932, 25802, 25702, 25474, 25412, 25279, 25253, 25192, 25058, 25039, 24864, 24654, 24595, 24508, 24497, 24496, 24376, 24345, 24324, 24250, 24202, 24093, 24069, 23977, 23833, 23793, 23758, 23407, 23207, 23152, 23080, 23023, 22961, 22772, 22764, 22743, 22739, 22695, 22660, 22655, 22649, 22587, 22582, 22579, 22579, 22576, 22572, 22467, 22412, 22346, 22284, 22190, 21694, 21671, 21599, 21567, 21546, 21502, 21499, 21459, 21338, 21299, 21148, 21132, 21004, 20926, 20822, 20818, 20701, 20654, 20643, 20633, 20474, 20396, 20009

HARD3

34746, 34740, 34738, 34679, 34566, 34566, 34437, 34404, 34037, 33786, 33749, 33609, 33606, 33587, 33508, 33490, 33363, 33346, 33279, 33269, 33211, 33145, 33032, 33000, 32818, 32811, 32703, 32481, 32478, 32414, 32307, 32032, 32009, 31971, 31940, 31937, 31851, 31751, 31678, 31598, 31575, 31503, 31491, 31462, 31449, 31414, 31299, 31232,

31037, 31025, 30940, 30934, 30865, 30720, 30704, 30677, 30499, 30394, 30265, 30264, 30249, 30188, 29896, 29750, 29750, 29623, 29553, 29435, 29404, 29376, 29288, 29280, 29216, 29162, 29068, 29036, 29022, 28885, 28758, 28746, 28566, 28462, 28308, 28077, 27961, 27896, 27800, 27680, 27509, 27509, 27504, 27482, 27474, 27402, 27327, 27302, 27299, 27237, 27205, 27169, 27019, 27008, 26993, 26946, 26737, 26667, 26663, 26635, 26506, 26375, 26310, 26229, 26132, 26075, 26036, 26011, 25993, 25726, 25604, 25579, 25501, 25466, 25454, 25349, 25296, 25225, 25143, 25050, 25028, 24838, 24796, 24724, 24688, 24585, 24518, 24458, 24451, 24312, 24256, 24239, 24212, 24175, 23857, 23791, 23680, 23452, 23406, 23405, 23369, 23367, 23346, 23336, 23290, 23174, 23096, 23070, 23057, 22950, 22917, 22896, 22893, 22823, 22781, 22678, 22352, 22351, 22308, 22268, 22220, 22217, 22195, 22097, 22063, 22036, 21965, 21856, 21751, 21615, 21613, 21585, 21415, 21346, 21328, 21310, 21299, 21269, 21267, 21117, 20919, 20903, 20847, 20778, 20773, 20740, 20664, 20633, 20600, 20530, 20423, 20033

HARD4

35000, 34970, 34839, 34733, 34369, 34328, 34237, 34229, 34225, 34197, 34154, 34002, 33988, 33977, 33958, 33934,

33891, 33839, 33471, 33218, 33149, 32979, 32940, 32936, 32912, 32902, 32900, 32885, 32802, 32802, 32802, 32708, 32637, 32415, 32403, 32200, 32110, 32068, 32067, 32058, 31950, 31946, 31923, 31919, 31690, 31624, 31562, 31482, 31475, 31450, 31432, 31405, 31363, 31187, 31107, 31088, 30940, 30873, 30866, 30750, 30538, 30527, 30497, 30370, 30347, 30290, 30156, 30140, 30118, 30051, 29845, 29750, 29654, 29646, 29552, 29512, 29415, 29403, 29382, 29300, 29271, 29151, 29131, 28998, 28951, 28937, 28867, 28821, 28820, 28724, 28696, 28489, 28380, 28267, 28252, 28225, 28223, 28105, 28104, 28044, 27900, 27864, 27699, 27668, 27661, 27593, 27589, 27570, 27497, 27416, 27322, 27287, 27271, 27221, 26975, 26881, 26813, 26692, 26591, 26520, 26432, 26337, 26290, 26289, 26219, 25966, 25822, 25563, 25546, 25461, 25442, 25361, 25356, 25281, 25259, 25122, 25078, 25024, 24793, 24790, 24789, 24721, 24714, 24424, 24413, 24341, 24325, 24234, 24198, 24149, 24092, 23920, 23907, 23864, 23811, 23799, 23781, 23671, 23662, 23493, 23299, 23206, 23162, 23139, 23119, 23013, 22984, 22983, 22872, 22846, 22771, 22533, 22467, 22246, 22237, 22217, 22166, 22143, 22140, 22095, 22045, 21930, 21774, 21753, 21744, 21500, 21369, 21289, 20986, 20971, 20920, 20899, 20897, 20892, 20788, 20774, 20738, 20368, 20299, 20139

HARD5

34955, 34773, 34641, 34529, 34478, 34453, 34441, 34399, 34131, 34102, 33996, 33978, 33732, 33523, 33445, 33437, 33428, 33386, 33338, 33183, 33140, 33108, 33076, 33005, 32986, 32984, 32859, 32819, 32749, 32681, 32620, 32582, 32504, 32425, 32417, 31766, 31717, 31699, 31648, 31566, 31505, 31373, 31355, 31273, 31264, 31216, 31064, 31008, 30918, 30905, 30751, 30724, 30707, 30689, 30617, 30592, 30519, 30459, 30315, 30297, 30279, 30246, 30246, 30148, 30138, 30069, 29962, 29899, 29898, 29737, 29735, 29626, 29590, 29495, 29434, 29159, 29063, 28917, 28862, 28709, 28678, 28524, 28426, 28296, 28231, 28213, 28210, 28198, 27960, 27628, 27622, 27502, 27473, 27345, 27330, 27323, 27301, 27240, 27120, 27090, 27015, 26845, 26839, 26828, 26636, 26607, 26570, 26554, 26311, 26308, 26270, 26225, 26219, 26211, 26088, 26067, 26060, 25994, 25942, 25920, 25916, 25866, 25827, 25735, 25600, 25561, 25504, 25443, 25437, 25380, 25097, 25077, 25071, 25054, 25037, 24941, 24933, 24871, 24843, 24788, 24751, 24720, 24594, 24565, 24361, 24312, 24168, 24153, 24152, 24145, 24109, 24088, 23852, 23829, 23766, 23654, 23630, 23572, 23482, 23379, 23172, 23012, 22937, 22936, 22897, 22887, 22886, 22876, 22689, 22673, 22670, 22542, 22345, 22262, 22199, 22131, 22109, 22095, 21958, 21712, 21642, 21440, 21345, 21296,

21156, 21147, 21122, 21048, 21036, 21031, 21021, 20960, 20812, 20646, 20500, 20443, 20409, 20385, 20382, 20000

HARD6

34973, 34910, 34885, 34807, 34720, 34655, 34630, 34613, 34536, 34230, 34226, 34172, 34069, 34069, 34066, 33902, 33843, 33761, 33637, 33632, 33429, 33351, 33343, 33303, 33300, 33259, 33070, 33045, 33022, 32986, 32881, 32785, 32759, 32649, 32583, 32560, 32558, 32545, 32380, 32332, 32297, 32113, 32077, 31943, 31916, 31787, 31770, 31719, 31718, 31701, 31652, 31641, 31470, 31269, 31227, 31138, 31006, 30831, 30828, 30814, 30582, 30580, 30561, 30379, 30371, 30339, 30150, 30125, 30104, 30098, 30075, 30039, 29907, 29860, 29627, 29547, 29532, 29516, 29404, 29313, 29268, 29186, 29179, 29139, 9051, 28932, 28820, 28716, 28692, 28436, 28360, 28321, 28298, 28086, 27954, 27911, 27758, 27642, 27627, 27616, 27464, 27393, 27334, 27321, 27202, 27080, 27032, 26978, 26794, 26705, 26671, 26630, 26449, 26409, 26354, 26345, 26307, 26278, 26192, 26188, 26112, 26014, 25959, 25808, 25806, 25741, 25655, 25640, 25611, 25609, 25491, 25344, 25233, 25134, 25028, 24967, 24931, 24870, 24584, 24512, 24507, 24476, 24424, 24413, 24382, 24363, 24356, 24200, 24129, 24089, 24064, 24043, 23991, 23866, 23765, 23632, 23595, 23547, 23483, 23378, 23335, 23324, 23302, 23232, 23224, 23147, 23088, 22948, 22922, 22886, 22778, 22618, 22513, 22487, 22450, 22433, 22345, 22237, 22232, 22149, 22041, 21753, 21720, 21711,

21649, 21634, 21577, 21473, 21472, 20895, 20817, 20619, 20613, 20598, 20565, 20433, 20395, 20348, 20081, 20050

HARD7

34808, 34689, 34603, 34583, 34336, 34297, 34244, 34192, 34092, 34045, 34030, 33976, 33959, 33872, 33820, 33736, 33641, 33592, 33405, 33362, 33333, 33299, 33253, 33242, 33223, 33120, 33093, 33067, 32733, 32256, 32193, 32094, 32003, 31894, 31788, 31746, 31734, 31720, 31675, 31651, 31648, 31618, 31611, 31599, 31598, 31312, 31095, 31062, 30853, 30793, 30691, 30599, 30567, 30537, 30462, 30436, 30264, 30246, 30218, 30053, 30037, 29942, 29941, 29879, 29779, 29746, 29688, 29682, 29641, 29633, 29563, 29462, 29461, 29450, 29356, 29299, 29288, 29280, 29235, 29169, 29129, 28955, 28954, 28671, 28437, 28336, 28269, 28200, 28000, 27973, 27968, 27914, 27885, 27759, 27741, 27653, 27567, 27563, 26904, 26550, 26402, 26366, 26361, 26348, 26225, 26139, 26108, 25991, 25718, 25683, 25639, 25462, 25290, 25228, 25136, 25043, 25038, 24962, 24892, 24823, 24803, 24768, 24621, 24559, 24441, 24419, 24381, 24250, 24235, 24093, 24083, 24065, 24060, 23974, 23868, 23833, 23636, 23633, 23581, 23523, 3445, 23413, 23317, 23202, 23160, 23150, 23117, 22977, 22959, 22955, 22947, 22915,

22833, 22755, 22739, 22603, 22592, 22557, 22554, 22530, 22354, 22313, 22306, 22095, 22092, 22021, 21948, 21934, 21913, 21855, 21594, 21564, 21543, 21518, 21440, 21389, 21370, 21205, 21174, 21027, 20984, 20969, 20932, 20900, 20844, 20816, 20721, 20694, 20584, 20533, 20490, 20476, 20343, 20332, 20260, 20173, 20162, 20157, 20131, 20017

HARD8

34992, 34948, 34868, 34591, 34582, 34127, 34077, 34055, 34007, 34004, 33990, 33918, 33813, 33780, 33756, 33744, 33700, 33659, 33496, 33484, 33443, 33428, 33369, 33354, 33347, 33191, 33185, 33162, 33110, 32988, 32968, 32879, 32846, 32797, 32708, 32656, 32584, 32486, 32466, 32456, 32440, 32390, 32373, 32353, 32352, 32282, 32187, 32111, 32097, 32084, 32017, 31990, 31917, 31880, 31817, 31752, 31540, 31528, 31471, 31309, 31267, 31232, 31204, 30773, 30703, 30552, 30549, 30515, 30305, 30221, 30162, 30115, 30107, 30072, 30010, 29972, 29704, 29550, 29547, 29547, 29457, 29418, 29325, 29226, 29155, 29034, 28859, 28837, 28652, 28535, 28502, 28423, 28421, 28388, 28386, 28348, 27930, 27919, 27793, 27703, 27669, 27365, 27266, 27096, 26928, 26868, 26848, 26677, 26676, 26673, 26658, 26559, 26507, 26476, 26424, 26421, 26320, 26251, 26224, 26214,

26128, 25943, 25900, 25879, 25852, 25821, 25720, 25655, 25625, 25495, 25455, 25174, 25150, 25104, 25028, 24917, 24898, 24860, 24813, 24682, 24659, 24475, 24370, 24301, 24283, 24273, 24251, 24230, 24199, 24088, 24086, 24084, 24023, 23947, 23872, 23736, 23725, 23609, 23562, 23515, 23453, 23414, 23235, 23078, 23036, 22937, 22932, 22897, 22826, 22680, 22664, 22646, 22523, 22404, 22287, 22240, 22151, 21978, 21963, 21921, 21866, 21747, 21655, 21560, 21464, 21403, 21046, 21041, 21020, 20796, 20778, 20774, 20622, 20603, 20410, 20371, 20248, 20236, 20146, 20091

HARD9

34991, 34941, 34922, 34866, 34849, 34771, 34768, 34748, 34544, 34358, 34254, 34155, 34098, 34076, 34055, 34048, 34029, 33990, 33871, 33780, 33750, 33654, 33612, 33581, 33430, 33260, 33197, 33155, 33115, 33007, 32989, 32795, 32708, 32394, 32384, 32309, 32193, 32039, 32038, 32008, 31995, 31961, 31946, 31865, 31839, 31829, 31692, 31633, 31354, 31169, 31141, 31006, 30929, 30843, 30842, 30807, 30741, 30514, 30395, 30387, 30341, 30296, 30287, 30284, 30140, 30135, 30063, 29975, 29933, 29859, 29735, 29730, 29703, 29525, 29518, 29423, 29378, 29234, 29218, 29178, 29092, 29089, 28947, 28647, 28574, 28550, 28547, 28471,

28461, 28299, 28267, 28252, 28251, 28159, 28009, 28003, 27967, 27852, 27811, 27664, 27508, 27413, 27409, 27184, 27162, 27113, 27099, 27048, 27041, 26733, 26506, 26362, 26183, 25997, 25976, 25897, 25856, 25784, 25700, 25668, 25641, 25522, 25490, 25433, 25408, 25322, 25299, 25237, 25091, 25057, 25015, 24990, 24974, 24939, 24834, 24777, 24743, 24625, 24555, 24449, 24367, 24340, 24329, 24126, 24085, 24050, 24020, 23999, 23989, 23974, 23928, 23837, 23836, 23565, 23491, 23422, 23417, 23205, 23195, 23156, 23092, 22712, 22644, 22417, 22392, 22281, 22239, 22212, 22067, 22045, 22042, 22003, 21866, 21851, 21849, 21713, 21674, 21608, 21607, 21594, 21401, 21296, 21239, 21180, 21128, 21059, 20954, 20948, 20947, 20813, 20755, 20725, 20693, 20585, 20513, 20431, 20338, 20310, 20296, 20081

جامعة النجاح الوطنية

كلية الدراسات العليا

خوارزميات لتقنيات التحسين لمشكلة تعبئة الحاويات: دراسة مقارنة

إعداد

ياسمين الكرمي

إشراف

د. بكر عبد الحق

قدمت هذه الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في الحوسبة المتقدمة، بكلية الدراسات العليا، في جامعة النجاح الوطنية، نابلس، فلسطين.

2021

خوارزميات لتقنيات التحسين لمشكلة تعبئة الحاويات:

دراسة مقارنة

إعداد

ياسمين الكرمي

إشراف

د. بكر عبد الحق

الملخص

قد اقترحت مشكله مهمه جدا من مشاكل التحسين بحيث تهدف الى ترتيب مجموعه من العناصر كل عنصر له وزن محدد في اقل عدد ممكن من الحاويات، كل حاوية لها سعة معينه لا يمكن تجاوزها، هذه المشكلة تسمى مشكلة تعبئة الحاويات. لذلك قام الباحثون بتقديم وتطوير العديد من خوارزميات التحسين لحل هذه المشكلة. هذا البحث يهدف لعمل مقارنة بين بعض الخوارزميات الارشادي' المقترحة مؤخرا وبعض الخوارزميات الارشادية العليا المشهورة لحل مشكلة تعبئة الحاويات ذات البعد الواحد. تم اقتراح خوارزميتين ارشاديتين من قبل zehmakan وهما خوارزميات تقارب لها نسبة تقارب تقدر ب 2/3 هذه الخوارزميات واعده حيث انها لها أداء أكثر فاعليه وأفضل عن باقي الخوارزميات الارشادية بالإضافة الى خوارزميتين ارشاديتين تقليديتين هما اول ملائم مرتبا تنازليا (FFD)، وأفضل ملائم مرتبا تنازليا(BFD)، تم مقارنتها مع اثنتين من الخوارزميات الارشادية العليا وهما الخوارزميات الوراثية وخوارزميات تحسين سرب الجسميات مع ضبط العوامل المتغيرة.

في هذا البحث، تمت الاستعانة بالعديد المشاكل المرجعية لها حل مثالي معروف. بعض هذه المشاكل مرتبة تنازليا والبعض الاخر عشوائية. هذه المشاكل تختلف بالحجم، بعضها صغيرة مكونة من 9 او 20 عنصر، او متوسطة مكونة من 100، 150 عنصر مثلا، أو كبيره مكونة من 500 او 1000 عنصر. بالإضافة الى انها تختلف بالصعوبة بين سهل ومتوسط. لذلك تم مقارنة الخوارزميات من حيث عدد الحاويات والوقت المستهلك لقياس مدى فاعلية الخوارزميات. بعد مقارنة الخوارزميات الارشادية، تبين ان الخوارزمية التقريبية الثانية كان أدائها أفضل من

الأولى ولكنها تسهلك وقت أطول. وأن الخوارزميات التقليدية تفوقت على الخوارزميات التقريبية في المشاكل السهلة بينما الخوارزمية التقريبية الثانية تفوقت على التقليدية في المشاكل الصعبة وعندما تمت مقارنة الخوارزميات الارشادية العليا تبين ان في بعض المشاكل المرجعية تفوقت خوارزميات تحسين سرب الجسميات على الخوارزميات الوراثية من حيث عدد الحاويات لكنها تستهلك وقت أطول وفي البعض الاخر من المشاكل المرجعية، أداء الخوارزميتان كان متشابه جدا من حيث عدد الحاويات الوقت المستهلك مع العلم ان خوارزميات تحسين سرب الجسيمات تفوق بمقدار بسيط جدا على الخوارزميات الوراثية. وعند مقارنة جميع العناصر تبين ان الخوارزميات الارشادية تفوقت على الخوارزميات الارشادية العليا من حيث عدد الحاويات والوقت المستهلك.