The enhanced Best Performance Algorithm for Global Optimization with Applications

by

Mervin Chetty

200270501

Submitted in fulfilment of the academic requirements for the degree of

Doctor of Philosophy

(Computer Science)

at the

School of Mathematics, Statistics and Computer Science

University of KwaZulu-Natal

Durban, South Africa

November 2016

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION

The research described in this thesis was performed at the University of KwaZulu-Natal under the supervision of Prof. A. O. Adewumi. I hereby declare that all materials incorporated in this thesis is my own original work except where acknowledgement is made by name or in the form of a reference. The work contained herein has not been submitted in part or whole for a degree at any other university.

Signed:

Mervin Chetty

Date: November 2016

As the candidate's supervisor, I have approved the thesis for submission

Signed:

Prof. A. O. Adewumi

Date: November 2016

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION – PLAGIARISM

I, _____ declare that

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other University.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    a. Their words have been re-written but the general information attributed to them has been referenced
    b. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signed:

_____

Mervin Chetty

DEDICATION

To my family,
and to the children of our next generation.

ACKNOWLEDGEMENTS

Without my belief in God, and in the Lord Jesus Christ, this thesis would not have been possible. This thesis is a testimony of the goodness of God in my life. To God I am grateful for the completion of this thesis.

I am very grateful to my supervisor Prof. Aderemi O. Adewumi. Thank you for all you have done for me, and for all opportunities given. God bless you!

I am grateful to my mother Shirley; you are honored this day.

I am grateful to my aunty Asothi, who has stood with us through the difficult times.

I am grateful my late brother-in-law Devan and sister Deloshni. Devan would have been proud of this achievement.

I acknowledge my cousin Gansen and his wife Linda for their selfless deed at the very beginning.

I also thank my employer I.T. Dynamics for the financial assistance. I have been blessed being a part of this organization.

The financial assistance of the National Research Foundation (DAAD-NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the DAAD-NRF.

ABSTRACT

This work has focused on deriving inspiration from the social context of nature in having developed a new stochastic algorithm called the enhanced Best Performance Algorithm (eBPA). The eBPA aims at solving complexed discrete and continuous optimization problems, and is a further development of an algorithm introduced in our earlier work called the Best Performance Algorithm (BPA).

As opposed to similar algorithms that drew inspiration from biological and natural elements, the eBPA has derived its inspiration from human cognitive decision-making processes such as the selection of the best team for game playing. The eBPA tries to capture the competitive element of an individual in trying to achieve the objective of ultimate personal success by way of learning and adaption. The eBPA uses this intelligence for efficient, consistent, and robust search, within a complexed global search space.

This thesis presents the fundamental principles and foundations on the eBPA. The efficiency and robustness of this algorithm is tested on two common discrete optimization problems, namely the symmetric Travelling Salesman Problem and the Just-in-Time machine-scheduling problem. The algorithm is further applied to solve a newly defined real-world Annual Crop Planning (ACP) problem. A new mathematical formulation of the ACP problem, based on the market economic factors of the economy of scale and the demand and supply relations, is introduced in this work. This problem seeks to determine optimal resource allocations for crop planning in considering irrigation and other requirements. Solutions from the ACP problem intend to assist crop planners in making resource allocation decisions for the forthcoming crop production year cycle.

The performance of the eBPA on the stated problems was evaluated empirically via simulation experiments. The results obtained have been compared to those of other standard metaheuristics with the eBPA showing promising and efficient results. The proposed optimization technique thus shows strength for contribution in the field of optimization, being a metaheuristic. Furthermore, it opens further doors for optimization researchers to seek inspiration in the area of human cognitive decision-making, in metaheuristic design.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Nomenclature	Definitions
ACO	Ant Colony Optimization
ACP	Annual Crop Planning
AFV	Average Fitness Value
AI	Artificial Intelligence
AVG	Average
BFV	Best Fitness Value
BPA	Best Performance Algorithm
CI	Confidence Interval
CL	Candidate List
CP	Cost of Production
CS	Cuckoo Search
CWR	Crop Water Requirement
eBPA	enhanced Best Performance Algorithm
FA	Firefly Algorithm
GA	Genetic Algorithm
GSO	Glowworm Swarm Optimization
IS	Initial Solution
IWR	Irrigated Water Requirement
JIT	Just-in-Time
LS	Local Search
NN	Nearest Neighbor
NP	Non-deterministic Polynomial time
P	Polynomial
PL	Performance List
PSO	Particle Swarm Optimization
SA	Simulated Annealing

SAP	Space Allocation Problem
TL	Tabu List
TS	Tabu Search
TSP	Travelling Salesman Problem
VCP	Variable Costs of Production
VIS	Vaalharts Irrigation Scheme

OUTCOME OF RESEARCH WORK (PUBLICATIONS)

Article Accepted or Published in accredited Peer-reviewed Journals

1. Chetty, S. and Adewumi, A. O. (2013). "Studies in Swarm Intelligence Techniques for Annual Crop Planning Problem in a New Irrigation Scheme", South African Journal of Industrial Engineering, Vol. 24(3), pp. 205-226.
2. Chetty, S. and Adewumi, A. O. (2014). "On the Performance of New Local Search Heuristics for Annual Crop Planning: Case Study of the Vaalharts Irrigation Scheme", Journal of Experimental & Theoretical Artificial Intelligence, Vol. 27(2), pp. 159-179.
3. Chetty, S. and Adewumi, A. O. (2015). "A Study on the Enhanced Best Performance Algorithm for the Just-In-Time Scheduling Problem," Discrete Dynamics in Nature and Society, Vol. 2015, pp. 1-12.
4. Chetty, S. and Adewumi, A. O. (2017). "The enhanced Best Performance Algorithm for the Travelling Salesman Problem", Contemporary Engineering Sciences, Vol. 10(3), pp. 129-144.

Articles under consideration for Peer-reviewed Journals

5. Chetty, S. and Adewumi, A. O. "Theoretical Analysis on the enhanced Best Performance Algorithm", to be submitted the International Journal of Applied Mathematics and Computer Science.

6. Chetty, S. and Adewumi, A.O. “Investigation on the enhanced Best Performance Algorithm for the Annual Crop Planning Problem Based on Economic Factors”, submitted to PLOS ONE Journal (revised version submitted).

Chapter One
Introduction and Background

1.1 Introduction

Mathematical optimization is the science of determining the optimal solution from amongst a set of feasible solutions to a mathematically formulated problem. Essentially, the process involves first mathematically formulating the problem, and then to determine optimal or near optimal solutions using appropriate optimization techniques. The solutions found relate to the scarce resources that are required to be optimized. The process of optimization is usually subjected to certain constraints.

Mathematical optimization is an important tool used in decision making and system analysis. It is practically applicable in numerous fields; examples include the fields of Mathematics, Computational Science, Operations Research, Engineering, Economics, Physics, and Biology, etc. (Boyd and Vandenberghe, 2004). For example, in Microeconomics, the utilization problem addresses the issue of how to spend money in a way that will maximize utility. Another example exists in the fields of Science and Engineering; it is the problem of determining the minimum energy configuration of metallic structures (Snyman, 2005). To formulate an optimization problem, the objective (or objectives) of the problem need to be identified, along with the design variables and constraints that govern feasible solutions.

An objective function is a measure of the quality of a system in evaluating a solution. For example, in crop production the objective in evaluating the design variables may be to maximize the profits earned from the sale of the harvests. The design variables are therefore the inputs to the system and are the unknowns which need to be optimized. As an example, the design variables in crop production could be the area of land allocated for the production of each crop. Feasible solutions are also commonly governed by constraints. Constraints are the functions that describe the relationships between the different decision variables of the system. For example, in allocating land-area for the production of each crop, the quantity allocated cannot lay beyond the minimum and maximum bounds allowed. An optimization problem is formally defined as follows (Snyman, 2005):

$$\underset{w.r.t.: x}{\text{minimize}}\, f(x) \tag{1.1}$$

subject to: $g_i(x) \leq 0, i = 1,2,\dots,m$ (1.2)

$h_j(x) = 0, j = 1,2,\dots,r$ (1.3)

where: $x \in \mathbb{R}^n$

In equation (1.1), f is the objective function, and x is a solution which is a representation of the design variables. Solution x is selected from within a set of feasible solutions which exist within the domains of the solution space. $g_i(x)$ represents the inequality constraints, while $h_j(x)$ represents the equality constraints. This mathematical formulation represents a minimization problem. This problem can be made a maximization problem by putting a negative sign in front of $f(x)$.

The design variables could either be continuous (real-number values), or discrete (integer values). If the design variables are continuous, then the problem is referred to as a continuous optimization problem. If the design variables are discrete, then the problem is referred to as a discrete or combinatorial optimization problem.

The constraints associated with the variables of the system could either be hard or soft constraints (Domshlak *et al.*, 2006). Hard constraints are those constraints that cannot be broken. On the other hand, soft constraints are negotiable. The objective in determining solutions is to satisfy all hard constraints, while satisfying as many soft constraints as possible. If no constraints govern the problem, then the problem is referred to as an unconstrained optimization problem. However, if constraints govern the problem, then the problem is referred to as a constrained optimization problem. Most real-world optimization problems are multi-constrained. However, many optimization problems exist without constraints.

If the objective function and its constraints are all linear equations then the problem is categorized as a Linear Programming problem. If the object function, and/or one or more of the constraints are non-linear, then this problem is referred to as a Non-linear Programming problem. Similarly, there are other types of problem categories that exist; examples include the Geometric and Quadratic Programming problems, amongst others (Raju and Kumar, 2007).

The objective of the problem could be either single or multi-objective. If the problem has a single objective function, then the problem is referred to as a Single-objective Programming problem. Similarly, if multiple objective functions exist, then the problem is referred to as a Multi-objective Programming Problem. Most real-world optimization problems are multi-objective in nature.

Also, if the optimal solution for the optimization problem can be determined within polynomial time complexity (P), then the problem is characterized as being deterministic. With deterministic optimization, there are clear relationships between the constructs of the design variables and their solution qualities. In other words, the same design variables used to evaluate the objective function, within P, will determine the same result each time. On the other hand, if the optimal solution cannot be determined within P, then the problem is categorized as being a non-deterministic polynomial (NP) problem.

1.2 Exact and Heuristic Methods

Exact methods are used to determine the optimal solution to deterministic optimization problems. Examples of exact methods include Branch and Bound, Linear Programming, and the Divide and Conquer algorithms, amongst others (Adewumi, 2010). These methods determine the optimal solution by performing an exhaustive search of the solution space, irrespective of computational time complexity (Trevisan, 2011). Examples of problems that are solved to optimality, in using exact algorithms, are the decision problems in Linear Programming.

However, for NP-type optimization problems, heuristic methods are the preferred methods of choice. This is because performing an exhaustive search of the solution space is considered impractical if the computational time complexity increases exponentially. Therefore, for NP type optimization problems, computational time complexity is a factor. Meanwhile, most real-world optimization problems are NP in nature especially when the search space is large. Examples of NP-Hard optimization problems include the Travelling Salesman Problem (TSP), and the Just-in-Time (JIT) scheduling problem, amongst others. For these problems, heuristically determining sub-optimal solutions, within P, is considered acceptable in trading accuracy for reductions in computational time complexity (Syam and Al-Harkan, 2010). Heuristic algorithms proceed by employing trial and error

techniques in searching for solutions. Examples of heuristic algorithms include tour construction heuristics, such as the Nearest Neighbor and Greedy algorithms (Davendra, 2010).

One problem with heuristic algorithms, however, is premature convergence (Rocha and Neves, 1999). Premature convergence occurs when an algorithm gets trapped within a local neighborhood region of the solution space, in believing it has found the optimal solution, when in fact it has not. To overcome this occurrence, research has undergone to develop heuristic algorithms which embed greater levels of intelligence in performing the search. Heuristic algorithms that embed greater levels of intelligence are referred to as metaheuristic algorithms. Metaheuristic algorithms minimize the risk of premature convergence by performing more robust search. Metaheuristic algorithms fall under a category of algorithms known as Monte Carlo algorithms (Krauth, 1998).

1.3 Monte Carlo Metaheuristics

Monte Carlo algorithms are computational algorithms which rely on strategies of randomness to heuristically determine solutions. They are applicable to optimization problems where numerical methods are expected to fail, for example the NP-Hard optimization problems. For these problems, Monte Carlo algorithm determine sub-optimal solutions within P.

Examples of Monte Carlo metaheuristic algorithms include: the Evolutionary Algorithms such as the Genetic Algorithm (GA) (Holland, 1975); Swarm Intelligence algorithms such as the Ant Colony Optimization (ACO) (Dorigo, 1992; Dorigo and Gambardella, 1997), the Firefly Algorithm (FA) (Yang, 2010), Glowworm Swarm Optimization (GSO) (Krishnand and Ghose, 2009a; Krishnand and Ghose, 2009b), and the Cuckoo Search (CS) (Yang, 2010); Local Search algorithms such as Simulated Annealing (SA) (Kirkpatrick, 1983; Tan, 2008) and Tabu Search (TS) (Glover, 1989 and 1990). Descriptions of these algorithms are given in the sub-sections below.

1.3.1 Genetic Algorithm

GA (Holland, 1975) was inspired by natural evolution. With the GA, a population of solutions (phenotypes) evolve from one generation to the next in using techniques such as selection, crossover and mutation. Selection, crossover and mutation are strategically administered to the chromosomes

(genotypes) of the solutions within the population. For discrete optimization problems, the chromosomes are binary encoded while for continuous optimization problems, the chromosomes are real-value encoded (Monyei, et al, 2014; Eiben and Smith, 2003).

1.3.2 Ant Colony Optimization

ACO (Dorigo, 1992) was inspired by observing the natural behavior of ants in search for food. Initially, the ants start off by moving in random directions. Upon finding a food source, the ant will lay down a detective substance called pheromone. Pheromone is an evaporable substance. With time, the pheromone trail will evaporate. However, if this trail is detected by another ant, this ant will likely follow the pheromone trail. If the ant follows this trail, it will lay down more pheromone; the trail will strengthen as more pheromone is laid down. Trails with stronger pheromone emissions will be more attractive for other ants to follow suit. The food source represents a solution.

1.3.3 Firefly Algorithm

The FA (Yang, 2010) was inspired by observing the natural abilities of fireflies in emitting a light source called bioluminescence. Bioluminescence is emitted with the purpose of attracting other fireflies for mating. The FA is designed using these three governing rules (Akinyely and Adewumi, 2014):

1. Fireflies are attracted towards other fireflies with greater levels of bioluminescence than itself. The attraction does not consider the sex of the other fireflies.
2. The attractiveness of a firefly relates to its brightness. However, with increases in distance, the brightness is assumed to diminish. The brightest firefly, from the population, will move in a random direction.
3. The evaluation of the problems objective relates to the brightness of the light emission.

1.3.4 Glowworm Swarm Optimization

GSO is similar to the FA. It was inspired in observing the natural behavior of glow-worms in emitting luciferin (Krishnand and Ghose, 2009a). Luciferin is a luminescent property which gets emitted in order to attract other glow-worms for the purpose of mating. The greater the level of luciferin emitted,

the more attractive the glow-worm would appear to be. A glow-worm will move in the direction of another glow-worm, but only if it falls within its range of view.

1.3.5 Cuckoo Search

CS (Yang, 2010) was inspired in observing the natural behavior of some parasitic cuckoo bird species. These bird species have the behavior of reproducing eggs and then abandoning them in the nests of other host birds. Some birds, upon having realized the intrusion, will throw the alien eggs away. Other birds will simply leave their nests, and build other nests elsewhere. In this algorithm, an egg represents a solution. The intention of the algorithm is to replace the weaker solutions, in the nest of the host bird, with higher quality solutions. CS is designed using these three governing rules:

1. A cuckoo bird will lay one egg at a time. It will then randomly insert this egg in the nest of the host bird.
2. For the next generation, the nest with the highest quality of solutions will be accepted.
3. The host bird will detect the intrusion given a constant probabilistic factor of $p_a \in [0,1]$.

1.3.6 Simulated Annealing

SA (Kirkpatrick, 1983) is modeled on the analogy of the atomic composition of metal. At higher temperatures, the atomic composition of metal is more volatile. Yet, it will stabilize as the metallic structure begins to cool. Stability (or equilibrium) is reached at a temperature close to zero. For the annealing process to be successful, the decrease in the rate of temperature must be slow. Volatility represents SA's ability to accept worst solutions. It is represented with probability $P = exp[(C - C^*)/T]$, where C is the cost of the current solution, C^* is the cost of the candidate solution, and T is the temperature. At higher temperatures, the probability of accepting worst solutions is greater. This allows SA to explore different neighborhood regions of the solution space with more ease. Using this strategy, more promising neighborhood regions can be located. However, as the temperature decreases, this probability also decreases and there is a transition from exploration to exploitation. Greater levels of exploitation presents SA the opportunity to concentrate on those promising neighborhood regions found in trying to identify higher quality solutions. The greatest levels of exploitation are achieved at very low temperatures, where the probability of accepting worst solutions are at its lowest. The strategy of accepting worst solutions is two-fold: new regions are explored, and

a doorway is presented to escape local entrapment. With SA, significant research has been done around the setting of its parameter values; these values significantly influence the performance of the algorithm. The initial temperature (T) importantly controls the transition from exploration to exploitation, and the cooling factor (α) importantly controls the rate at which the algorithm converges to its final solution. The algorithm for SA is given in Algorithm 1.1.

From Algorithm 1.1, it is seen that SA starts off with equivalent $best$ and $current$ solutions. The execution of the algorithm starts off at the initial temperature of T. T then reduces by the rate of $T \times \alpha$, until the final temperature F is reached. At each point of decrease in T, a $stepsPerChange$ number of iterations is executed. At each of these iterations, local search moves are applied to the $current$ solution; this will produce a $working$ solution. If this $working$ solution improves upon the $current$ solution, then it will become the next $current$ solution. However, given a certain probability, even if the $working$ solution does not improve upon the $current$ solution, it could still become the next $current$ solution. If the $current$ solution has been updated, a check is performed to see if the $best$ solution has been improved upon. If it has, the $current$ solution will become the next $best$ solution.

Algorithm 1.1: Simulated Annealing

---

1. Initialize $best$ to be the initial tour
2. Set $current = best$
3. Evaluate the fitness of $best = f\_best$
4. Set $f\_current$ (the fitness of $current$) $= f\_best$
5. Initiate starting temperature T and final temperature F
6. **while** $T \geq F$ **do**
    6.1. **for** i to $stepsPerChange$ **do**
        6.1.1. $working$ = Determine_Solution ($current$)
        6.1.2. $f\_working$ = Determine_Fitness($working$)
        6.1.3. **if** $f\_working$ better then $f\_current$ **then**
            6.1.3.1. $use\_solution$ = true
        6.1.4. **else**
            6.1.4.1. Calculate acceptance probability P
            6.1.4.2. **if** $P >$ random[0,1] **then**
                6.1.4.2.1. $use\_solution$ = true
            6.1.4.3. **end if**
        6.1.5. **end else**

6.1.6. **if** $use\_solution$ **then**

6.1.6.1. $use\_solution$ = false

6.1.6.2. $f\_current = f\_working$

6.1.6.3. $current = working$

6.1.6.4. **if** $f\_current$ better than f_best **then**

6.1.6.4.1. $best = current$

6.1.6.4.2. $f\_best = f\_current$

6.1.6.5. **end if**

6.1.7. **end if**

6.2. **end for**

6.3. Update T according to cooling schedule α

7. **end while**

8. **return** $best$

1.3.7 Tabu Search

TS (Glover, 1989) is based on the analogy of something that should not be touched or interfered with. This is achieved by maintaining a limited number of recently found best candidate solutions in a list called the Tabu List (TL). The TL is commonly implemented in a First-In-First-Out (FIFO) way. Candidate solutions are determined in searching the neighborhood region of the current solution x, i.e. $N(x)$. Therefore, the maximum number of candidate solutions considered will be $N(x) - |TL|$, as any solution recorded in the TL has a tabu status and will not be interfered with. The decision to reject the TL solutions minimize the risk of cycling. Thus, TS makes use of memory in intelligently directing the search. The algorithm for TS is given in Algorithm 1.2.

Algorithm 1.2: Tabu Search

---

1. Initialize $best$ to be the initial tour
2. Set $current = best$
3. Evaluate the fitness of $best = f\_best$
4. Set $f\_current$ (the fitness of $current$) $= f\_best$
5. Set the size of the Tabu List, i.e. $tabuListSize$
6. Set the size of the Candidate List, i.e. $candidateListSize$
7. Initiate the Tabu List (TL) and the Candidate List (i.e. $CandidateList$)
8. **for** i to $noOfIterations$ **do**

8.1. $CandidateList$ = Generate_New_Candidate_List($current$)
8.2. $current$ = Find_Best_Candidate($CandidateList$)
8.3. $f\_current$ = Determine_Fitness ($current$)
8.4. **if** $f\_current$ better then f_best **then**
8.4.1. $f\_best = f\_current$
8.4.2. $best = current$
8.4.3. Update TL with $current$
8.5. **else**
8.5.1. **if** Intensification_Criterion_Met() **then**
8.5.1.1. $current$ = Reset_Current()
8.5.2. **end if**
8.6. **end if**
9. **end for**
10. **return**

In Algorithm 1.2, a candidate list of solutions ($CandidateList$) is generated from the $current$ solution. The best candidate from the $CandidateList$ is then determined, and will become the new $current$ solution for the next iteration. If this solution improves upon the $best$ solution overall, then it will become the next $best$ solution. If the $best$ solution is updated, then it will get inserted into the TL. If the intensification criterion has been satisfied, the next $current$ solution will be re-determined in using a randomly selected solution from the TL.

1.3.8 A New Metaheuristic Algorithm

This thesis introduces a new Monte Carlo metaheuristic algorithm in the literature, namely the enhanced Best Performance Algorithm. The eBPA stems from its predecessor–the Best Performance Algorithm (BPA) which had been proposed earlier by the researcher (Chetty and Adewumi, 2013a). Details on both algorithms are provided in chapter two.

1.4 Rational and Motivation

The eBPA has been developed to improve on the BPA in making up for a few gaps identified in the latter, especially when applied to discrete optimization problems. Although the BPA performs competitively for continuous optimization problems, it performs poorly for discrete optimization problems due to an apparent weakness in its exploitative ability (exploration and exploitation will

also be discussed in detail in chapter two). Thus, research had been undergone to design a more complete metaheuristic algorithm which performs competitively for both discrete and continuous optimization problems. The resultant algorithm is the eBPA. The eBPA is an enhancement over its predecessor–the BPA. The eBPA has a completely different design to that of the BPA, yet being modelled on similar analogical principles.

With the global challenge in the agricultural sector, especially in developing countries, it becomes highly imperative for optimization researchers to develop models and methods that help manage activities and processes in this sector within the limitations of available resources. Economic challenges, coupled with drought and a host of other problems, have impacted negatively on food production and consumption. This is why the study on the ACP problem becomes highly essential, especially within a developing country context such as South Africa which is recently experiencing high levels of water shortage.

Increases in the costs associated with crop production, and the scarcity of natural resources such as fresh water supplies and agricultural land, make it essential to seek an optimal way for crop production per unit of the resources utilized. The need for more output is directly related to increases in the population growth. This has placed greater demands on the agricultural sector for food products. From all sectors of the industry, the agricultural sector is the primary supplier of food globally. Yet, determining optimized solutions in crop production is no simple task as there are many stochastic factors to be considered. This makes determining optimized solutions very challenging for both the producers and researchers alike. Therefore, to try and contribute to the solutions of this problem, the ACP problem had been introduced. The ACP problem provides solvable yet scalable solutions in considering both the stochastic and predictable factors involved with crop production.

Another reason for introducing the ACP problem is due to the fact that the scarcity of fresh water supplies is becoming a great concern especially in South Africa. From all sectors of the industry, the agricultural sector is placed under increased pressure to use fresh water supplies more conservatively. This is due to the fact that it is the most accused of excessive water wastage from all other sectors of the industry (Schmitz *et al.*, 2007). Yet, it is important that fresh water supplies to the agricultural sector do not deplete below acceptable levels as fresh water is essential for optimized agricultural production. Also, it is realized that increases in the costs associated with food products will have negative socioeconomic effects on the global society. Of this, the hardest hit will be that of the poor.

Therefore, to try and combat these challenges, the ACP problem had been introduced, and is further evolved in this thesis.

1.5 Aims and Objectives

The primary aim of this thesis is the presentation of the eBPA, and the new ACP mathematical formulation based on market economic factors.

The objectives of presenting the eBPA are as follows:

1. To pioneer research into the modelling of human behavioral traits in developing a metaheuristic algorithm within the AI framework.
2. To make comparisons with its predecessor, the BPA, for optimization problems.
3. To present theoretical insight into the technical and strategic differences between the algorithmic designs of both the eBPA and the BPA.
4. To investigate the potentials of the eBPA for NP-Hard optimization problems.

The objectives of presenting the ACP problem are as follows:

1. Formally describe the ACP problem, and its mathematical model.
2. To seek a better realistic model for the ACP problem that incorporates market economic factors along with other constraints which make for an enhancement over the previous version of the ACP problem introduced earlier by the author.
3. To investigate possible solutions to the ACP problem in considering irrigation constraints based on a real-life scenario obtainable in South Africa.
4. To provide benchmark datasets to aid further research in this area.

1.6 Methodology

This thesis studies metaheuristics in investigating three *NP*-Hard optimization problems, namely the ACP problem, instances of sTSP's, and an instance of the JIT scheduling problem. For the ACP problem presented, computational simulations were performed based on the dataset from a case study. The dataset, together with the results of all simulations, will serve as a benchmark for further research

conducted on this problem. Benchmark datasets available in literature and online were used for the TSP and JIT problems. Statistical analysis and comparisons were done to determine the efficiency of the proposed technique over existing methods.

All programs developed in this thesis were written using the Java programming language. It was programmed in using the Netbeans® 7.0 Integrated Development Environment. All simulations were run on the same platform. The computer used had a Windows® 7 Enterprise operating system, an Intel® Celeron® Processor 430, 3GB of RAM and a 500GB hard-drive.

1.7 Scope of the Thesis

This thesis presents the eBPA for global optimization problems. The enhancement of the eBPA over the BPA will be discussed. The strength of the proposed algorithm is evaluated on two common discrete optimization problems, namely the TSP and JIT machine scheduling problems. A real-world instance of the ACP problem is also formulated, and the eBPA employed to find optimized solutions. The results show that the eBPA competes favorably well within the space of both discrete and continuous optimization problems.

Being a new metaheuristic algorithm, this thesis constitutes initial investigations into the potentials of the eBPA. To test its abilities, the solutions determined by the eBPA, for all problem instances investigated, have been compared with that of the well-known TS and SA algorithms. TS and SA have been known in literature as efficient and competitive metaheuristics in determining high quality solutions to arrays of difficult real-world optimization problems. The comparisons of the eBPA against TS and SA is motivated by the fact that the former is designed based on similar underlying principles implemented by both these algorithms.

SA is a single-point stochastic and memory-less search technique; it is based primarily on randomization. On the other hand, TS is a single-point memory-based search technique which performs a search in a more deterministic way. The eBPA lay in-between both of these search techniques in that it performs the search stochastically, yet employs adaptive memory strategies to influence the direction of the search trajectory. The eBPA thus differs from memory-less search algorithms which are modelled primarily on randomization, such as SA, and memory-based search

algorithms which are modelled primarily on determinism, such as TS. The eBPA thus embeds characteristics of both stochastic and deterministic search strategies. The solutions determined for the problems to be investigated will show the abilities of the eBPA in executing an effective, consistent and robust search when being compared to that of the TS and SA algorithms.

The ACP problem had also previously been introduced in the literature (Chetty and Adewumi, 2013b; 2013c; 2013d; and 2014). The problem seeks to determine optimized resource allocations in crop planning at irrigation scheme level. The objective is to determine solutions that will maximize the total gross profits that could be earned from the sale of the harvests in the forthcoming production year. The intent of the problem is to advise crop planners in making resource allocation decisions at the land allocation stage of the crop production process. The research presented by Chetty and Adewumi (2013b; 2013c; 2013d; and 2014) was an attempt to introduce the ACP problem as an optimization problem in the literature. However, this thesis introduces a new mathematical formulation for the ACP problem. This ACP problem accommodates for the market economic factors of the economy of scale, along with demand and supply relations. Introducing these factors provide for more scalable solutions in advising crop planners regarding resource allocations in crop production.

1.8 Contributions to Knowledge

In metaheuristic design, the complexity of real-world optimization problems (especially with added dimensionality) require algorithms that make smarter decisions during the decision making process in problem solving. The objective is to determine higher quality solutions. However, developing such algorithms is no easy task as flexibility, simplicity and efficiency need to be balanced in the design.

To achieve these objectives in metaheuristic design, research has strongly leaned towards the direction of Artificial Intelligence (AI). AI attempts to simulate the intelligent behavior of biological agents (or occurrences) in nature which behave systematically in achieving an overall objective. However, although metaheuristic algorithms have been biologically inspired within the AI framework, not much research has been done by way of human modelling. Yet, significant research on human modelling has been done within the framework of Computational Intelligence (CI). CI

relates to soft computing techniques, and include fields such as fuzzy logic systems, artificial neural networks, learning theory, evolutionary computing, and probabilistic methods.

Also, an important aspect in real-world mathematical formulation is to model the problem as realistically as possible. To achieve this, all decision variables must be considered. However, the more decision variables added to the mathematical model in trying to achieve realism, the more complexed the mathematical model would appear to be. Reason being, the dimensionality of the solution space will increase.

Thus, for mathematical models that require a large number of decision variables, the objective would be to represent the most important elements of the problem in trying to achieve realism, without making the problem overly complexed to solve. An example of such problems are the decision problems in crop planning.

The overall contributions to knowledge are summarized as follows:

1. The introduction of a new stochastic metaheuristic named the eBPA. This has been designed to mimic the planned cognitive decision making abilities of an individual, whom attempts to achieve the objective of ultimate personal success within the context of a competitive environment. An example is a soccer coach seeking to put together the best team amidst a large pool of talented players. The eBPA encapsulates the competitive nature of an individual through mimicking intelligent ideas of learning and adaption.
2. Furthermore, the proposed eBPA incorporates features that make it problem-independent, simple, efficient, and have a good explorative and exploitative balance. This makes it a good candidate for researchers in the field of optimization to apply the algorithm in solving other optimization problems.
3. The eBPA has been designed based on human modelling. To the best of the knowledge of the author, no other metaheuristic has been spotted in the literature that encapsulates the competitive behavior of a human being in the way that the eBPA has, within the AI framework. The eBPA therefore has the potential to open doors to further research on the incorporation of human modelling in metaheuristic design, within the AI framework.
4. This work further presents a new model of the ACP real-world problem, with potential solutions that can have great imparts on decision making for government, farmers and other

interested parties. The incorporation of the market economic factors of the economy of scale and the demand and supply relations make the ACP mathematical model more practical in the field of crop production. The solution obtained with the proposed metaheuristics are very promising, and can serve as the underlying algorithms to drive decision support systems in this area especially for developing countries.

5. Finally, being not too common in literature, the results obtained by the eBPA for the ACP problem (and those of the other problems considered) will serve as benchmarks for further study.

1.9 Thesis Overview

Chapter two introduces the eBPA. Theoretical analysis is given on the fundamental principles underlying the strategic and technical designs of both the eBPA and the BPA. Investigations will be performed in highlighting the differences between these metaheuristics for a simple discrete optimization problem.

Chapter three will present the new ACP mathematical formulation. This formulation is based on the market economic factors of the economy of scale, and the demand and supply relations. To present the new mathematical formulation, the ACP problem will be explained. The results show the potentials of the BPA and the eBPA algorithms for a continuous optimization problem.

Chapter four further investigates the potentials of the eBPA in testing its abilities to ten benchmark test instances of sTSP's. eBPA's performances will be compared against that of TS and SA in testing the sequences of instructions constituting the algorithmic designs of these algorithms.

Chapter five investigates a particular instance of the JIT scheduling problem. Chapter five takes the opportunity to correct the previous mathematical formulation of this particular problem instance. It then further investigates the potentials of the eBPA in determining solutions.

Finally, chapter six draws conclusions. It also discusses the possibilities of future research on the eBPA.

Chapter Two
Theoretical Analysis of the enhanced Best Performance Algorithm

2.1 Introduction

Due to the complexities of optimization problems that exist, developments in computational science have led to the introduction of many non-standard optimization algorithms. Non-standard algorithms are more flexible in their designs, and are also applicable to a variety of problem settings, in being non-problem specific. An additional benefit is that the solutions determined by these algorithms are guaranteed within polynomial time horizons, for NP-Hard type optimization problems.

Typically, for these difficult to solve optimization problems, the classical numerical methods would fail to determine the optimal solution within P (Kougias and Theodosiou, 2010). Reason being, the numerical methods would experience difficulty due to the non-convex nature of the complexed solution spaces (Aspremont and Boyd, 2003).

The term "non-convex" implies that a solution space is characterized by multiple local optimum points; amongst these, the global optimum point would exist. The appearance of multiple local optimum points could be very deceptive in searching for the global optimum point. This deceptivity could easily cause an algorithm to prematurely converge to a point that is not the global optimum. Algorithms therefore need to be intelligent enough to be able to escape from local entrapment (Rocha and Neves, 1999).

There are numerous types of non-convex optimization problems that exist; these include practical applications in sectors such as Mathematics, Computational Science, Engineering, Economics, and others. To address the challenges of non-convex type optimization problems, many non-standard optimization algorithms have been developed. This study provides theoretical insight into one such non-standard optimization algorithm.

Loosely speaking, "meta" in the word "metaheuristic" means a higher level, and "heuristic" means to discover by trial and error (Yang, 2010). Hence, metaheuristic algorithms are more advanced heuristic algorithms. These algorithms have commonly been developed using Artificial Intelligence techniques. Many algorithms have been developed from observing the natural behavior of biological 'agents' and/or 'elements' in nature (Blum and Merkle, 2008).

Examples of behavioral patterns that have been studied include: flocks of birds, schools of fish, swarms of wasps, colonies of ants, particles of nature, the atomic composition of objects, and sound amongst many others. From these, metaheuristic algorithms that have been developed artificially include the GA, SA, TS, Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995; Arasomwan and Adewumi, 2013, 2014a, 2014b), the ACO, the FA, the BPA, and more recently the eBPA (the eBPA and the BPA are the metaheuristic algorithms that will be discussed in this study).

The intelligence that is packaged in metaheuristic algorithms are intended to effectively and robustly search the complex regions of the solution spaces. However, metaheuristic algorithms do not guarantee optimal solutions; yet, they do guarantee near-optimal solutions within polynomial time horizons for exponentially complexed problems. In this context, near-optimal solutions are considered acceptable in trading accuracy for reductions in computational time complexity.

This study presents theoretical analysis on the fundamental design principles of the eBPA. The eBPA is an enhancement of its predecessor, the BPA. The eBPA has been developed due to further research having been undergone in trying to improve upon the efficiency aspects of the BPA. Although the eBPA is conceptually similar to that of the BPA, the underlying strategies of the eBPA are altogether different. All-in-all, the eBPA is a new and more intelligently designed metaheuristic.

This chapter formally presents and describes the eBPA. It also highlights the strategic differences in the designs of both the BPA and the eBPA. To show the effects of these strategic differences, a comprehensive set of experiments will be performed in investigating the performances of these algorithms. The intent is also to assist the reader in better understanding the design techniques of the eBPA, and to highlight the advantages of employing the eBPA to practical applications.

Conceptually, the eBPA is modeled on the competitive nature of professional athletes, in them desiring to improve upon their best known performances. This analogy is similar to a heuristic

algorithm that seeks improved solutions. Competitive athletes desire to improve on their best performances through learning, strategizing and practice. In comparison, the eBPA seeks higher quality solutions, from iteration to iteration, in 'learning' from previously visited solutions. Learning is implemented through the concept of memory.

Memory techniques in metaheuristic design provide additional benefits over pure memory-less techniques (Glover, 1995). The eBPA uses the advantage of implementing memory strategies to direct the trajectory of the search, and to penetrate complexed regions of the solution spaces which may confine other methods. The eBPA also draws from the strength of stochasticity.

The rest of this chapter is structured as follows; Section 2.2 presents a brief background into local search metaheuristic algorithms. Section 2.3 presents the eBPA, and the BPA. It also discusses the technical and strategic differences between both these algorithms. Section 2.4 presents further analysis on the design principles of the eBPA. Section 2.5 briefly discusses the investigations to follow. Section 2.6 presents the results of the experiments performed, which will then be used to discuss the differences in the designs of both the algorithms. Finally, section 2.7 draws on the conclusions.

2.2 Local Search Metaheuristic Algorithms

Local Search (LS) metaheuristic algorithms determine solutions to computationally difficult optimization problems. Basically, they search through a solution space X, of objective function f, by repeatedly making slight adjustments (or local moves) from one solution x to another x' (x' will be chosen from a set of candidate solutions associated with x); the intent is to direct the search towards the global optima point. A key element of modern-day metaheuristic algorithms is to accept both improved and dis-improved solutions. Accepting dis-improved solutions is a strategic way of escaping local entrapment.

A local move is an adjustment to the design variables of solution vector x. This could include: the inversing of binary digits; adding, deleting, or the swapping of elements within the solution vector; and real number alterations. The set of candidate solutions associated with solution x is called the

neighborhood of x. The neighborhood of x is denoted as $N(x)$. It is defined as follows (Blum and Roli, 2003):

Definition 2.1: Let $\mathbb{N}: x \rightarrow 2^x$ be a function that assigns to every feasible solution $x \in X$ a subset of feasible solutions $j \in \mathbb{N}(x) \subseteq X$. $\mathbb{N}(x)$ is called the neighborhood of solution x if each neighbor $j \in \mathbb{N}(x)$ is in some way close to x within the domains of the solution space X.

The best solution found within the neighborhood structure of $N(x)$ is called the local optima. Local optimum are defined as follows (Hancock, 2005):

Definition 2.2: A local optimum point $x^* \in \mathbb{R}$ exists for some error value $\varepsilon > 0$ such that for a minimization problem $f(x^*) \leq f(x)$, and for a maximization problem $f(x^*) \geq f(x)$. These are subjected to $|x - x^*| < \varepsilon$, $\forall\ x \in \mathbb{R}$. Here, f represents the objective function, $\mathbb{R}$ represents a solution space of real numbers, and $|x - x^*|$ is the absolute value of the difference between x and x^*.

Within solution space X, several local optimum points may exist. The best local optimum point from this lot is called the global optimum point. Global optimum points are local optimum points, but not necessarily vice versa. A global optimum point is defined as follows (Snyman, 2005):

Definition 2.3: A global optimum point $x^* \in \mathbb{R}$ exists for a minimization problem if $f(x^*) \leq f(x)$, and for a maximization problem if $f(x^*) \geq f(x)$, $\forall\ x \in \mathbb{R}$.

Local and global optimum points are visually seen in Figure 2.1.

LS metaheuristic algorithms search for local optimum points, in trying to determine the global optimum point. Consequently, LS metaheuristic algorithms are considered as improvement techniques (Liberti, 2008). An obvious attempt to determine all local optimum points is to perform an exhaustive search of the solution space. However, for large to complexed solution spaces, this may be impractical. The reason is due to the computational time taken to examine every possible solution; ultimately, this may prove to be too expensive. In this scenario, examining subsets of feasible solutions, within the neighborhood regions of $N(x)$, is the alternative. This alternative is computationally more acceptable for exponentially complexed optimization problems. The challenge

then is to intelligently examine subsets of the most attractive solutions within these neighborhood regions, within the bounds of polynomial time complexity.

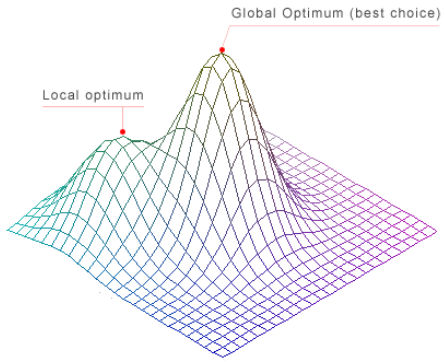


Figure 2.1: The global optimum point is the extreme local optimum point. A local optimum point is the best point within a neighborhood region. This image is for a maximization problem.

The techniques employed by metaheuristic algorithms are to strategically refine and explore subsets $S \in N(x)$ in ways that are efficient and computationally feasible. Metaheuristic algorithms thus employ a level of intelligence in searching for the best solution $x^* \in S \in N(x)$. In doing, these algorithms typically make use of the knowledge acquired from examining other neighboring solutions, in trying to determine x^*.

Hence, the goal of metaheuristic algorithms is to narrow the visited regions to subsets of solutions that are more representative of the local optimum points (Glover, 1993). For this reason, metaheuristic algorithms are justifiably more advanced than standard heuristic techniques (Yagiura and Ibaraki, 2001).

In addition to effectively and intelligently guiding the search, metaheuristic algorithms also need to be intelligent enough to escape from premature convergence. Premature convergence is when the algorithm believes that it has found the global optimum point, when in fact the global optimum point is not within the vicinity of the local neighborhood region being searched.

To escape from premature convergence, LS metaheuristic algorithms strategically allow for dis-improved solutions to be accepted. The dis-improved solutions accepted will also be used to traject through the solution space. Accepting dis-improved solutions mean that solution x', which is determined from solution x, is accepted, although $f(x')$ is worse off than $f(x)$. This strategy allows

for a re-direction of the search path. This re-direction may cause a break out of possible local entrapment; it could also lead the search towards other neighborhood regions which could potentially contain higher quality solutions.

However, accepting dis-improved solutions is accompanied with a risk; this risk is an effect called cycling (Glover, 1990). Cycling occurs when one solution consequently leads to another solution, in a repeated cycle. Cycling does not necessarily mean a repetition after one move transition, but could also be as a result of some interval of intermediate steps. Metaheuristic algorithms need to be watchful for the effect of cycling.

The stopping criterion of metaheuristic algorithms include:

1. Stop when the optimal solution is found.
2. Stop when a solution is found that falls within an acceptable degree of error.
3. Stop when the number of iterations exceed the upper bound.
4. Stop when a count of the iterations exceed a certain value since the best solution was last updated.

Once the stopping criterion is satisfied, the metaheuristic algorithm will return the final result.

2.3 Proposed Metaheuristic Algorithm

This section formally presents the eBPA. The eBPA has been developed using similar design principles to that of the BPA; these principles include memory and probability. As mentioned previously, the BPA had been introduced in the literature by Chetty and Adewumi (2013a). In that article, a detailed description is given on the BPA.

However, it was realized that the BPA has a weakness to discrete optimization problems. Hence, further research was conducted to improve on its efficiency. The new metaheuristic developed was the eBPA. The eBPA is an improvement over the BPA. It is also a new metaheuristic in having a different design to that of BPA.

In the next section, the eBPA will be presented. Thereafter, to highlight the differences, both the eBPA and the BPA algorithms will be presented side-by-side in Table 2.1. After that, the conceptual differences in the design techniques of both algorithms will be discussed.

2.3.1 The enhanced Best Performance Algorithm

The eBPA is modeled on the analogy of professional athletes desiring to improve upon their best registered performances within competitive environments. Numerous sporting disciplines exist, however the principles are the same in that professional athletes desire to improve upon their skill levels with the purpose of trying to supersede their previous best known performances. To start off with, all professional athletes develop an interest in the particular sport; they then realize the potential to succeed. Thereafter, with constant practice and strategizing, their skill levels increase. This happens as a result of learning from trial and error. In using trial and error, refined skills are developed by improving upon their strengths and weaknesses of the sport. The ultimate goal of the athlete is to develop a level of skill that would result in the athlete giving off a performance that would ultimately surpass any previous best performance.

Apart from other learning strategies, an effective strategy could be to maintain an archive of a limited number of the best performances delivered by the athlete; for example, video recordings could be archived. Video recordings contain the history of the way a previous best performance had been delivered. This also includes the technique (or techniques) that was employed, and the result determined. Knowledge of this information could be used to motivate the athlete to deliver higher quality performances. For example, the information of the worst performance on the list could motivate the athlete to at least improve upon this benchmark. Hence, if a performance is delivered which improves upon that of the worst performance already registered on the list, then the list could be updated by replacing the performance of the worst with that of the improved performance. In this way, the archive size will be maintained, but it also so happens that the quality of the worst performance on the list is now of a higher benchmark standard. Since this improved performance is the latest delivered, the athlete could then continue to work with the technique that was used to deliver that improved performance in trying to improve upon strengths and weaknesses.

Given the increased benchmark of the worst performance registered in the archive, the athlete is now presented with the challenge of working harder in order to deliver further improved performances. In

maintaining the archive, the athlete may want to be challenged further by decreasing the archive size. Decreasing the archive size will make it increasingly more difficult for the athlete to register further improved performances.

"Technique" or "skill" in this context refers to a solution determined by an optimization technique. Also, the "result" of a performance refers to the quality of a solution in it being used to evaluate the objective function f. Therefore, notable similarities can be seen between an athlete delivering performances and an optimization technique determining solutions. Based on the analogy of professional athletes desiring to improve upon their best archived performances, the eBPA was conceptualized. There are six foundational rules governing the design of the eBPA:

1. An athlete maintains an archive of a collection of a limited number of best performances.
2. From this collection, the record of the worst performance is identified. This becomes the benchmark for the athlete to try and improve upon.
3. If a new performance is delivered which improves upon (or is at least equivalent to) that of the worst performance, then the archive is updated by replacing the current worst performance with the new. However, upon performing the update, if it is realized that the result of the new performance is identical to that of any other performance in the archive, but different in terms of the technique that had been employed, then the new performance will replace the one with the identical result.
4. An athlete will endeavor to improve upon the performance that caused the most recent update of the archive.
5. All performances registered in the archive must be unique in terms of result and technique.
6. The archive size is strategically reduced until only one performance remains.

To artificially simulate this analogy, the eBPA maintains a limited number of the best solutions found by the algorithm in a list called the Performance List (PL). From all solutions, the design variables constituting the construction of each solution must be adjacently different; therefore, only unique solutions are allowed admittance into the PL. Dis-allowing duplicate solutions will prevent the algorithm from working with previously visited solutions. Also, the best, the worst and (what would be called) the working solutions in the PL must be indexed. The best and worst solutions are identified according to their solution qualities. Henceforth, the best, the worst and the working solutions will be referred to by the variables $best$, $worst$ and $working$.

To try and improve upon the $worst$ solution registered in the PL, local search moves will be applied to a copy of the $working$ solution; hence, a new solution $working'$ will be realized. $working'$ is chosen from a subset of candidate solutions within the neighborhood region of $working$. If $working'$ at least improves upon $worst$, or is at least equivalent in solution quality, yet unique in terms of its design variables, then the PL will be updated. If the solution quality of $working'$ is different from all solutions in the PL, then the $worst$ solution will be replaced by $working'$. However, if the solution quality of $working'$ is identical to that of another solution in the PL, then $working'$ will replace that particular solution in the PL; this will ensure that there are no two solutions in the PL with the same solution quality.

Being newly inserted into the PL, $working'$ will then become the next $working$ solution. Also, if $working'$ has improved upon $best$, it will also be indexed as the new $best$ solution. Upon this update, the $worst$ solution would then need to be re-determined and re-indexed. The solution quality of the latest $worst$ solution will now become the new benchmark standard to try and improve upon. If an update of the PL has not been made, then local search moves will continue to be applied to the copy of $working$.

However, given a certain probabilistic factor, the next $working$ solution could also be that of $working'$ even though an update of the PL had not been performed. The probabilistic factor represents the desire of the athlete to try out a new technique; this will continue indefinitely as determined by the probabilistic factor.

These strategies represent the eBPA's ability to accept both improved and dis-improved $working'$ solutions. $working'$ is considered improved if it at least improves upon $working$ in the PL. $working'$ is considered to be a dis-improved solution in two ways: if $working'$ is accepted into the PL yet is not an improvement over $working$; if $working'$ is not accepted into the PL and the probabilistic factor has been satisfied (i.e. this is a wayward solution that falls out of the scope of the solutions registered in the PL). Accepting dis-improved solutions is the eBPA's strategy of escaping premature convergence. Additionally, the admittance criterion shields against cycling.

An additional strategy is to dynamically reduce the PL size, until a PL size of one is achieved. Strategically decreasing the PL size allows for the admittance criterion to constrain further. Further constraining the PL size will intensify the search in sifting out higher quality solutions. This strategy also eliminates the possibility of cycling for PL sizes greater than one.

Also, to strategically try to break out of premature convergence (other than the strategies already encapsulated within the eBPA), an option exists to temporarily increase the size of the PL. However, temporarily increasing the PL size could open up the possibility of cycling in redirecting the search trajectory. The option of temporarily increasing the size of the PL is out of the scope of this initial research.

After the termination criterion is satisfied, the $best$ solution will be returned. This solution is representative of the best performance delivered by the athlete. The eBPA is presented in Table 2.1, as Algorithm 2.1. In Algorithm 2.1: resize() checks to strategically resize the PL; is_PL_Populated() checks to see if the memory structure has been fully populated, and if not then it will populate it with $working'$ by calling method populate(…); perform_Update(…) inserts $working'$ into the memory structure and re-indexes the $best$, $worst$ and $working$ solutions where applicable.

The flowchart diagram of the eBPA is seen in Figure 2.2. Also, Appendix A presents a hypothetical illustration of how the eBPA would typically traject through a solution space. The BPA is also presented in Table 2.1, as Algorithm 2.2.

Table 2.1: The eBPA is presented as Algorithm 2.1, and the BPA is presented as Algorithm 2.2

Algorithm 2.1: enhanced Best Performance Algorithm	**Algorithm 2.2**: Best Performance Algorithm
1. Initialize variables: $bestIndex = 0,\ workingIndex = 0,\ worstIndex = 0$ 2. Set the size of the Performance List, i.e. PL_size 3. Set probability p_a 4. Set the first solution in the Performance List, i.e. $PL_{workingIndex}$ 5. Calculate the fitness value of $PL_{workingIndex}$, i.e. $PL\_Fitness_{workingIndex}$ 6. Set Boolean variable $toggle = true$ 7. **while** not Stopping_Criterion_Met() **do** 7.1. **if** resize() **then** 7.1.1. resize_PL() 7.2. **end if** 7.3. **if** $toggle$ **then** 7.3.1. $working$ = Determine_Solution($PL_{workingIndex}$) 7.4. **else** 7.4.1. $working$ = Determine_Solution($working$) 7.4.2. $toggle$ = true 7.5. **end if** 7.6. $f\_working$ = Determine_Fitness ($working$) 7.7. **if** ($f\_working$ better than or equal to $PL\_Fitness_{worstIndex}$) and is_PL_Populated() **then** 7.7.1. perform_Update($working$, $f\_working$) 7.8. **else** 7.8.1. **if** not is_PL_Populated() **then** 7.8.1.1. populate($working$, $f\_working$) 7.8.1. **end if** 7.9. **end if** 7.9. **if** random[0,1] $\leq p_a$ **then** 7.9.1. $toggle = false$ 7.10. **end if** 8. **end while** 9. return $PL_{bestIndex}$	1. Set the index variable, $index = 0$ 2. Set the size of the Performance List , $listSize$ 3. Initialize probability, p_a 4. Populate the Performance List (PL) with random solutions 5. Calculate the fitness values of the solutions in PL, i.e. $PL\_Fitness$ 6. Sort PL and $PL\_Fitness$ according to $PL\_Fitness$ 7. Initialize $working$ to PL_{index} 8. **while** not Stopping_Criterion_Met() **do** 8.1. $working$ = Perform_Local_Search($working$) 8.2. $f\_working$ = Evaluate ($working$) 8.3. **if** $f\_working$ better then $PL\_Fitness_{listSize-1}$ **then** 8.3.1. Update PL with $working$ 8.3.2. Update $PL\_Fitness$ with $f\_working$ 8.4. **end if** 8.5. **if** random[0,1] > p_a **then** 8.4.1. $index$ = Select index, e.g. Random[0,$listSize$] 8.4.2. $working = PL_{index}$ 8.6. **end if** 9. **end while** 10. **return** PL_0

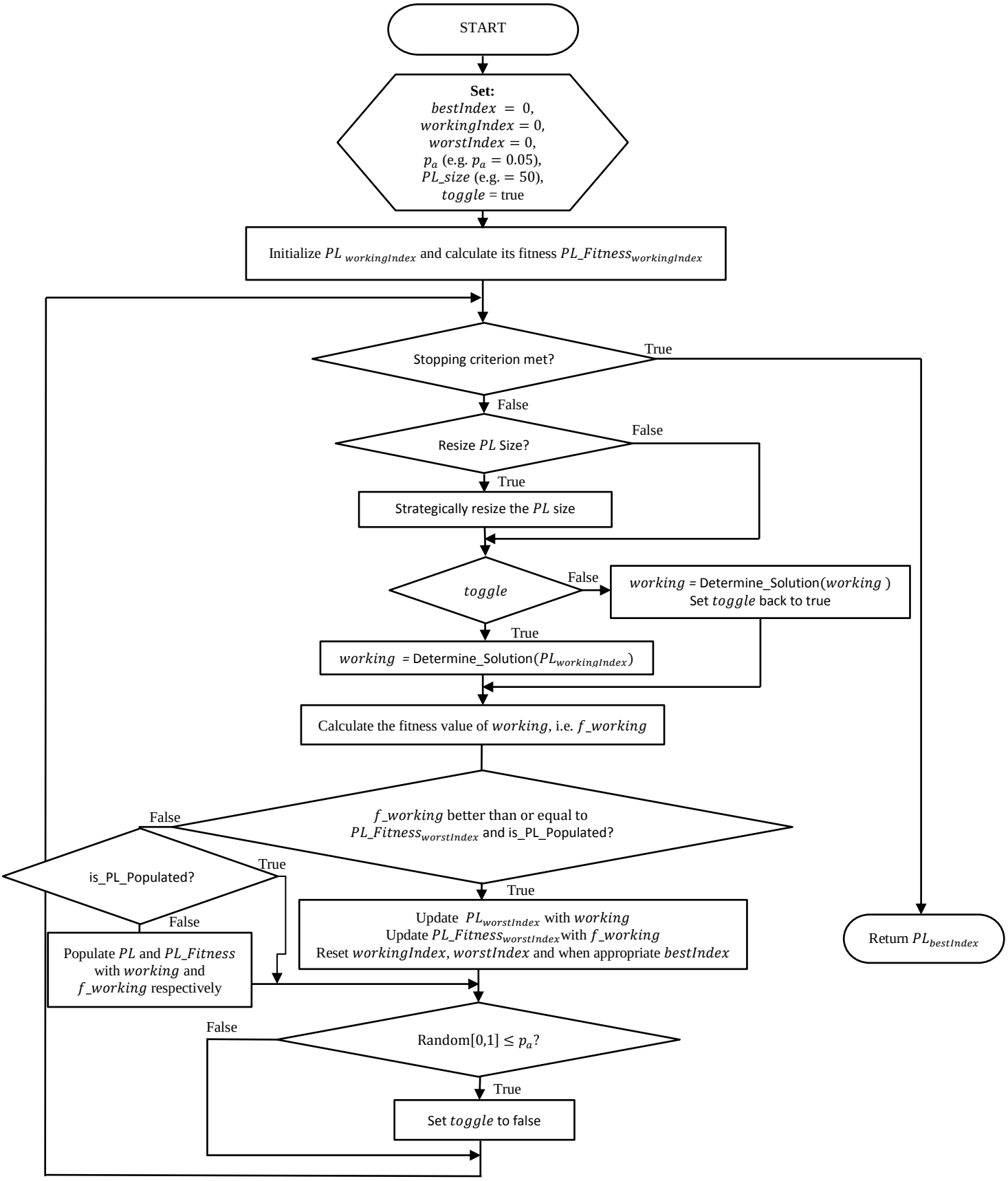


Figure 2.2: Flowchart diagram of the eBPA

2.3.2 Enhancement over the BPA

This section analyses the conceptual differences in the design techniques of both the BPA and the eBPA. The primary differences include the following:

- Maintenance of the memory structure.
- Admittance criterion.
- Search strategy.
- Size of the memory structure.

2.3.2.1 Maintenance of the Memory Structure

BPA – The BPA starts off with having the memory structure pre-populated with random solutions. These solutions are then arranged in a sorted order according to their fitness values; a fitness value refers to the quality of a solution. The sorted order is always maintained such that the best solution is positioned at the first index in the memory structure, and the worst solution is positioned at the last index in the memory structure. Therefore, in executing the algorithm, only one index is required to be maintained; this is the index which references the $working$ solution. Given a certain probability, the next $working$ solution will be randomly selected from the memory structure. Upon the termination criterion being satisfied, the solution residing at the first position in the memory structure will be returned as the best solution found by the algorithm.

eBPA – The eBPA starts off with a single solution in the memory structure. It then populates the memory structure until it becomes fully populated. Throughout the process of getting populated, and beyond the point of being fully populated, the $best$ and $worst$ solutions will be appropriately indexed. All newly inserted solutions will always be referenced as the next $working$ solution. Up until the point of the memory structure being fully populated, solutions get inserted irrespective of their fitness value. However, the aggressive condition of enforcing uniqueness still applies as all solutions must differ in fitness and in design variables. If a solution is found to have an identical fitness value to that of a solution already registered in the PL, yet differs in terms of its design variables, then the new solution replaces that of the old. This strategy ensures that only unique solutions are registered in the PL. The eBPA does not maintain a sorted order of the memory structure. In contrast, it employs indices which references $best$, $worst$ and $working$ in the list. Once the

memory structure is fully populated, the admittance criterion of meeting the minimum benchmark requirements will apply. The process of maintaining the memory structure continues until the stopping criterion is satisfied. At this point, the $best$ solution will be returned.

Maintaining the sorted order and in randomly reselecting the working solution, given a certain probability, results in additional computational expense experienced by the BPA. In comparison, the eBPA does not maintain a sorted order of the memory structure. Rather, it maintains indices which reference the $best$, $worst$ and $working$ solutions. This strategy is computationally more efficient.

With each update of the memory structure, the BPA replaces the $worst$ solution (which is located at the last position in the memory structure) with $working'$. Thereafter, the PL needs to be reordered in moving $working'$ to its correct location; this maintains the sorted order of the PL. With the eBPA, the $worst$ solution is replaced with $working'$ at the location referenced by the $worst$ index. The $working$ index is then re-assigned to this location, as $working'$ will be used as the next $working$ solution. At this point, a simple check is performed to see if the newly inserted solution is the $best$ solution found. If it is, then this location will be referenced by the $best$ index. Thereafter, the location of the $worst$ solution will be re-determined and also indexed by the $worst$ index.

In the cases of both the BPA and the eBPA, the admittance criterion is the screen for admissibility. Also, the prevention of duplicate solutions into the memory structures is a level of precaution against cycling.

2.3.2.2 Admittance Criterion

BPA – The BPA admittance criterion is that the $worst$ solution must be improved upon. Thereafter, the BPA enforces uniqueness in ensuring that every solution in the PL differs in terms of their fitness value and design variables. The advantage of this approach, compared to that of the eBPA, is that this restrictiveness requires less processing effort. The disadvantage, however, is that solutions with identical fitness values, yet being unique in terms of their design variables, could have led to higher quality solutions.

eBPA – The eBPA enforces uniqueness in terms of fitness value and design variables of every solution registered in the PL. However, it differs from the BPA in that it allows for new solutions

with identical fitness values to replace those solutions in the memory structure which have the identical fitness values. Hence, the minimum criterion for admittance is that the fitness value of $working'$ must be at least equivalent to that of the $worst$. The disadvantage of accepting solutions with identical fitness values is that it requires extra processing effort in ensuring uniqueness. However, the advantage of this strategy is that it could possibility redirect the search in finding higher quality solutions.

2.3.2.3 Search Strategy

BPA – The intent of the BPA, in maintaining a population of solutions, is to try not to lose good solutions found along the way; the belief is that one solution from the memory structure will at least lead to the global optimum point. The direction of the population itself is controlled by the admittance criterion. The admittance criterion only allows solutions that improve upon the $worst$ solution to be allowed admittance into the memory structure. This will influence the direction of the population. The reason why only unique solutions are allowed admittance into the memory structure, with the BPA being so restrictive, is that each solution identifies a different area within the searched region. In employing a population to perform the search, the BPA is thus stronger in its explorative ability. However, it lacks in exploitation as the next $working$ solution will be randomly selected from the PL given a certain probability.

eBPA – The eBPA search strategy contrasts with the BPA in that a single solution is directed in searching for the global optimum point. The direction of the search is primarily controlled by the admittance criterion. However, given a certain probability, the next $working$ solution could be that of a dis-improved solution which had not been admitted into the memory structure. The admittance criterion plays a critical role in balancing the rate of transition from exploration and exploitation. The less restrictive the admittance criterion, the greater the level of exploration would be. Likewise, the more restrictive the admittance criterion, the greater the level of exploitation would be.

2.3.2.4 Size of the Memory Structure

The BPA uses a static memory size, as a population of solutions are used to search for the global optimum point. In contrast, the eBPA employs dynamic reduction of the memory structure. This

strategy is additionally used to tweak the rate of exploitation, and eliminates the possibilities of cycling for PL sizes greater than one.

2.4 The Strategic Design of the eBPA

The eBPA is designed to use intelligence to seek out solutions effectively and economically. Its core design is structured around adaptive memory in maintaining a list of elite solutions.

The memory structure is governed by a set of underlying principles: the dynamic resizing of the memory structure; the admittance criterion; and the maintenance of the indices which reference the $best$, $worst$ and $working$ solutions. Another governing principal is the probability factor. The probability factor strategically allows dis-improved solutions to redirect the search beyond the point of the $worst$ solution.

Memory is primarily used as the mechanism to direct the trajectory of a single $working$ solution through the solution space. The technique of random search, coupled with adaptive memory, embeds a blend of stochastic and deterministic search strategies. The eBPA technique is similar to a memory-less search technique, such as SA, in that a single solution is used to stochastically drive the search. However, the eBPA uses determinism, similar to that of the TS, in that memory is used to decide on the next $working'$ solution to be used to advance the search. Hence, the eBPA search techniques differs from memory-less techniques which are modeled primarily on randomization, and memory-based techniques which are modeled primarily on being deterministic. Therefore, the eBPA takes advantage of both stochastic and deterministic search strategies in balancing the computational time spent in locating promising neighborhood regions via exploration, and the time spent identifying the most attractive solutions contained within a local neighborhood region via exploitation.

To expound further, more clarity is given on the eBPA memory technique, its search strategies, its explorative and exploitative balance, and the strategic resizing of its memory structure.

2.4.1 Memory Technique of the eBPA

The fitness value of a solution registered in the memory structure refers to its strength upon having been evaluated by objective function f. Solutions with better fitness values exist closer to the global optimum point. The qualities of the fitness values are important in differentiating one solution from the next. Importantly, in using the fitness values, the $best$ and $worst$ solutions can be identified and indexed. The index of the $working$ solution will always be that of the most recent solution inserted into the memory structure. The maintenance of these indices are critical in implementing the search strategies of the eBPA. These indices relate to the way the memory structure will adapt as the search progresses. The enforced restrictions of the admittance criterion, coupled with the maintenance of these indices, is core to the design of the eBPA.

The admittance criterion directly influences the trajectory path of the search, as this feature controls the quality of the solutions registered in the memory structure. If $working'$ is allowed admittance, and has a higher fitness value compared to that of $working$, then a more attractive position within the solution space has been identified; this implicitly could also refer to the best solution found. If $working'$ has a lower fitness value (i.e. a dis-improved solution) compared to that of $working$, and has been allowed admittance, then a less attractive but acceptable position has been identified in the trajectory of the search. This strategy could possibly lead to an alternative route in locating the local optimum point; it could also cause a redirection to other neighboring regions in escaping from premature convergence.

As the search matures, the quality of the solutions in the memory structure are increasingly refined as higher quality solutions get accepted. With improved solutions, the admittance criterion would become increasingly restrictive. The increase in the restrictiveness of the admittance criterion controls the trading-off between exploration and exploitation.

2.4.2 Search Strategy of the eBPA

The search techniques employed by the eBPA causes a loosely knitted relationship between the neighborhood region being searched (i.e. the neighborhood region of $working$) and that of the other solutions registered in the memory structure. The neighborhood region gets redefined upon $working'$ being accepted, as this will become the next $working$ solution.

However, concerning the trajectory of the search, $working'$ solutions get accepted in two ways: if it meets the minimum admittance criterion in being accepted into the memory structure; or, in having been chosen given a certain probabilistic factor. If $working'$ is admitted into the memory structure, and has a fitness value that is better than that of the $worst$ solution, then the next $worst$ solution will be of an improved quality. If this occurs, the admittance criterion will become more restrictive as the minimum criterion of admittance would increase.

Resultantly, this will also cause the local optimum points to become more clearly defined (see Appendix A for a clearer explanation). With the admittance criterion becoming increasingly restrictive, greater levels of number-crunching would be required to determine further improved solutions. Hence, the decisions made by the admittance criterion is strategically used to influence the behavior of the search.

Within a neighborhood region itself, the ultimate objective is to locate the local optimum point. However, the eBPA uses intelligence in open-mindedly accepting dis-improved solutions; accepting dis-improved solutions attempts to redirect the search path. This strategy protects against premature convergence in directly attempting to lead away to other neighboring regions. The intent of accepting dis-improved solutions is to balance the effort invested in sifting out the local optimum point from within a local neighborhood region, and in searching for other promising neighborhood regions via exploration. With the neighborhood regions being restructured dynamically, upon updates of the memory structure, the possibility of revisiting previously found solutions remains unlikely.

2.4.3 Exploration and Exploitation of the eBPA

Metaheuristic algorithms are characterized by two important yet contrasting search strategies– exploration and exploitation (Syam and Al-Harkan, 2010).

Exploration is a global search technique. Its intent is to visit as many neighborhood regions as possible within the confines of the solution space. Ideally, exploration needs to be more influential during the initial phases of the search.

On the other hand, exploitation is a local search technique. Its intent is to search within a local neighborhood region in trying to locate the local optimum point. Exploitation needs to be more influential during the latter stages of the search, as it aggressively sifts out higher quality solutions.

Striking a balance between exploration and exploitation, throughout the different phases of a search, is critical to the success of any metaheuristic algorithm. Also, this balance is of paramount importance in implementing effective and economical search. Reason being, there is a fine balance between the computational time spent in exploring for the most attractive neighborhood regions, and the computational time spent in exploiting within a local neighborhood region for the optimum point.

The eBPA uses adaptive memory to intelligently control the rate of the transition from exploration to exploitation. During the initial phases of the search, the admittance criterion is less restrictive as the memory structure consists of lower quality solutions; hence, greater levels of exploration is experienced. With the fitness of the solutions in the memory structure being improved upon, with matured search, the admittance criterion becomes increasing restrictive. This allows for greater levels of exploitation to be experienced. Exploitation attempts to incorporate the stronger elements of the $working$ solutions into new $working'$ solutions, while discouraging the weaker elements. In performing exploration, the eBPA supposes that strategically accepting dis-improved solutions is more beneficial than a good random solution in influencing the trajectory of the search.

2.4.4 Strategic Reduction of the eBPA Memory Structure

The strategic reduction of the eBPA memory structure is critical to its success. It is also considered strategically more beneficial than maintaining a static memory structure size. The intelligence of strategically reducing the memory structure size will influence greater levels of exploitation as the admittance criterion would constrain further. The advantage of further intensifying exploitation is to place additional pressure in attempting to identify higher quality solutions.

A recommended strategy is to strategically reduce the memory structure size by one, until a memory size of one is reached. Using this technique, every solution admitted into the memory structure will be given a chance to act as the $worst$ solution. Therefore, every solution will be given a chance to influence the trajectory of the search.

The initial size of the memory structure is problem-specific. The results section (i.e. section 2.6) below gives an idea of how to set the memory structure size appropriately.

2.5 Experimental Setup

The results section (i.e. section 2.6) essentially investigates the sequences of instructions constituting the algorithmic designs of both the BPA and the eBPA. Although the BPA and the eBPA have conceptually similar designs, the fundamental differences in their designs cause each algorithm to behave and execute differently. To discuss the effects of these differences, we present a comprehensive investigation by performing a series of experiments to highlight the differences. Regarding the eBPA, further experiments will be performed in examining how differences in its parameter settings will affect its performance.

For the experiments, the eil101 symmetric Travelling Salesman problem (sTSP) will be investigated. Reason being, this is a discrete optimization problem and will prove the weakness of the BPA. This problem, together with a host of others, can be found in the TSPLIB collection made available by Gerhard Reinelt online. The TSP problem is a well-studied discrete optimization problem. It is the problem of determining the minimal tour which traverses a list of n cities in a way in which every city is visited exactly once, except for the original city of departure which is the starting and finishing point of the salesman (Lin, 1965). The complexity of the TSP is NP-Hard.

The move mechanism employed, in implementing the eil101 problem, is to swap two randomly selected vertices in generating a new tour. This simple move mechanism is sufficient for the purpose of this investigation. The purpose is to fairly compare the performances of both algorithms, and to discuss their differences. Overall, the performances of both algorithms will indicate the effectiveness of the core sequences of instructions constituting their algorithmic designs; this is the purpose of the investigation. Their performances will also indicate their abilities to accurately, consistently and robustly determine solutions for discrete optimization problems.

In performing the comparison tests, the parameter settings, and the number of runs per experiment, for both the eBPA and the BPA, will be identical. This is to ensure fairness for comparative purposes. For the comparison investigation on convergence, the eBPA will also be compared against the TS

metaheuristic. Here, TS will represent a typical single-point metaheuristic in comparing differences with that of the eBPA. Thereafter, two experiments will be performed in investigating the correlation between the eBPA parameter values and its performance.

2.6 Results and Discussion

All runs will be performed within a window frame of a limited number of iterations; each run will execute for 10^6 objective function evaluations. Although this particular benchmark problem is investigated in this study, it is expected that the algorithms will perform similarly for other types of discrete optimization problems. Also, it should be noted that the BPA and the eBPA are general purpose metaheuristic algorithms; they are expected to be applied to other types of optimization problems of which other metaheuristic algorithms are applicable.

2.6.1 Simulation Experiments

The first fundamental difference is that the BPA uses a population of solutions to collectively move towards the global optimum point, whereas the eBPA uses a single $working$ solution for the same purpose. The underlying rules governing the designs of both algorithms facilitate these search strategies. Amongst these are the rules related to the admittance criterion, and the rules for maintaining the memory structures. The execution time used to determine admissibility, and the time taken to maintain the memory structures weigh heavily on the overall execution times of the algorithms.

2.6.1.1 Experiment 1

Therefore, we start off the first experiment from this perspective: monitor the average execution time consumed by each algorithm in performing a single update of its memory structure. To perform this experiment, various PL sizes will be investigated: 1, 10, 25, 50, 75 and 100. In using different PL sizes, the effects on the average execution times per PL update will be monitored.

For each PL size, each algorithm was executed 100 times. The probability factor (p_a) remained constant at 0.2 per algorithm for the duration of this experiment. For the sake of comparability, the PL size for the eBPA remained constant; however, it is expected that with strategic reduction of the

PL size, the execution time of the eBPA would have decreased due to the maintenance of a smaller memory structure. The results of the experiment are given in Table 2.2.

Table 2.2: Average execution time, in milliseconds, to perform a single update of the *PL* memory structure

PL Size	eBPA	BPA
1	9.55E-09	1.60E-05
10	6.88E-08	3.95E-05
25	2.37E-05	6.17E-05
50	5.98E-06	1.14E-04
75	4.03E-06	1.72E-04
100	3.34E-06	1.72E-04

Figure 2.3 displays graphical representations of the statistics given in Table 2.2. Figure 2.3 shows that for each *PL* size, the eBPA shows significant gains in average execution time performances, per *PL* update. Clearly, maintaining the indices which reference the $best$, $worst$ and $working$ solutions show to be significantly more efficient than maintaining the sorted order of the BPA memory structure. The eBPA also allows for solutions with duplicate fitness values to be allowed admittance into the memory structure; this feature demands additional processing effort in ensuring uniqueness. However, despite this additional processing effort, the eBPA still shows to be computationally more efficient. In contrast, the BPA gains in execution time by simply rejecting solutions which do not have unique fitness values.

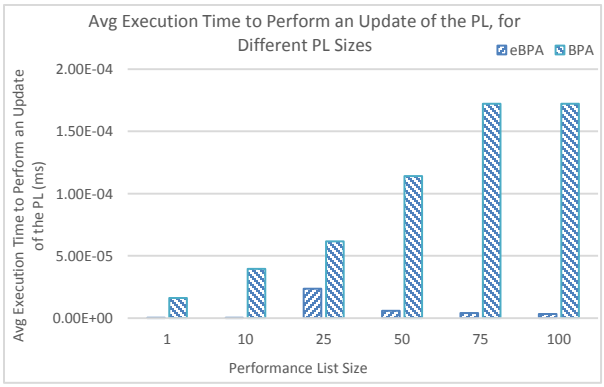


Figure 2.3: Comparison of average execution times, in milliseconds, to perform a single update of the *PL*'s of the eBPA and the BPA, for different *PL* sizes

2.6.1.2 Experiment 2

For the second experiment, we investigate the number of times each PL got updated, within the interval of iterations taken to reduce the PL size of the eBPA by 1. The reduction strategy employed by the eBPA was: reduce the PL size by 1 until a PL size of 1 is reached, after every $10^6/PL\_size$ number of iterations. For this experiment, a PL size of 50 was used; therefore, the reduction was done after every 2,000 iterations. To make the comparison with the BPA, the number of times the BPA memory structure got updated was also recorded after every segment of 2,000 iterations. For this experiment, the probability of $p_a = 0.2$ was used per algorithm. Each algorithm had been executed once. The comparison of the algorithmic performances are seen graphically in Figure 2.4.

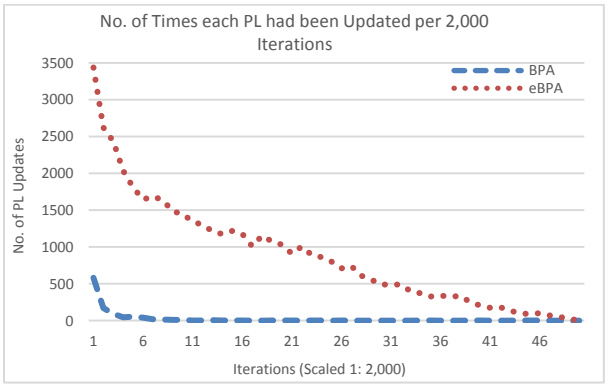


Figure 2.4: The number of times each PL had been updated, per segment of 2,000 iterations

From Figure 2.4, it is clearly seen that the eBPA determined a significantly larger number of PL updates, per segment of 2,000 iterations, compared to that of the BPA. The eBPA depicted plot shows evidence of its ability to balance exploration and exploitation. The pattern of the plotted slope is seen to have a somewhat concaved shape; the slope itself shows the drop in the number of PL updates throughout the lifespan of the execution. The figure shows that the level of exploration was highest during the initial phase of the search, and thereafter reduced as the admittance criterion became more difficult to satisfy. The admittance criterion would have become more restrictive with improved solutions, and in the strategic reduction of the PL size. The increase in the admittance criterion would have been accompanied by greater levels of exploitation. Greater levels of exploitation would have caused the eBPA to fight harder in determining further improved solutions. The slope illustrates how the eBPA balanced its transition from exploration to exploitation. In comparison, the weaknesses of

the BPA in its explorative and exploitative abilities have been exposed in observing its inability to have determined larger numbers of PL updates per segment.

2.6.1.3 Experiment 3

For the third experiment, a head-to-head performance comparison test was investigated. For this experiment, the best and average fitness value solutions (i.e. BFV and AFV respectively), as well as the average execution time performances have been documented. Tests were performed in using the PL sizes of 1, 10, 25, 50, 75 and 100. For each PL size, each algorithm was executed 100 times. Again, the p_a value of 0.2 was used per algorithm for all simulations. The strategic reduction of the eBPA PL size was the same as was implemented in experiment two above: reduce the PL size by 1 after every $10^6/PL\_size$ number of iterations. The statistics of the results are given in Table 2.3.

Table 2.3: The best (BFV) and average (AFV) fitness value solutions, together with the average execution time performances (AVG) in milliseconds (ms), per PL size

PL Size	eBPA			BPA		
	BFV	**AFV**	**AVG**	**BFV**	**AFV**	**AVG**
1	837	947	7,149	863	982	8,432
10	695	726	7,155	876	955	8,227
25	674	695	8,119	869	953	8,910
50	686	710	7,841	856	969	8,591
75	683	720	7,508	865	973	7,836
100	695	746	7,224	904	975	8,575

From Table 2.3, it is seen that the eBPA delivered superior performances for the fitness value solutions (at best and on average), and for the average execution time performances across all PL sizes. Graphical comparisons of the best and average fitness value solutions from Table 2.3, per PL size, are seen in Figures 2.5 till 2.10.

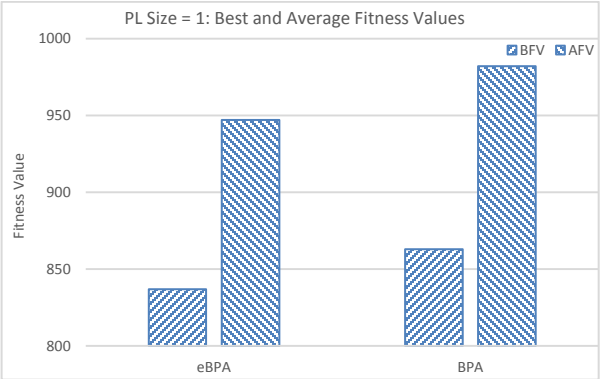


Figure 2.5: Best and average fitness values, per *PL* size of 1

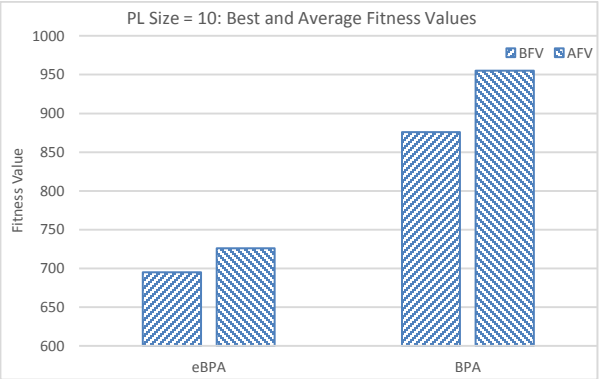


Figure 2.6: Best and average fitness values, per *PL* size of 10

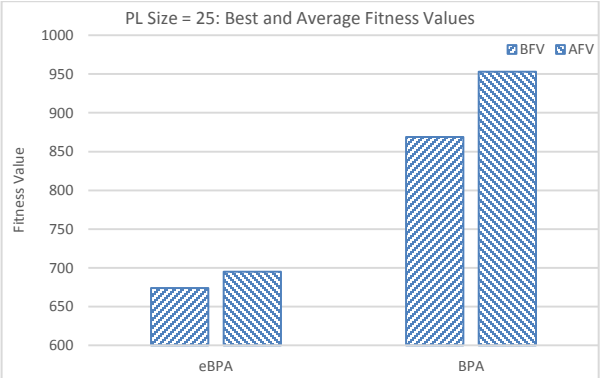


Figure 2.7: Best and average fitness values, per *PL* size of 25

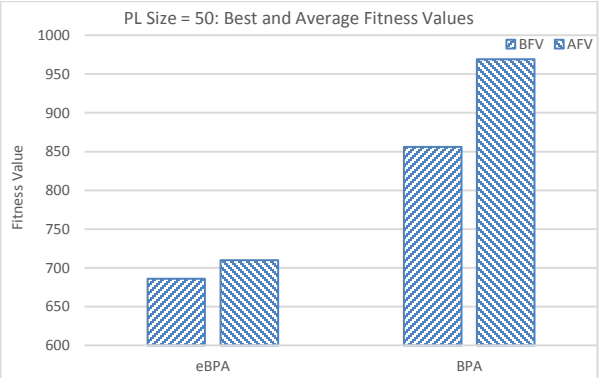


Figure 2.8: Best and average fitness values, per *PL* size of 50

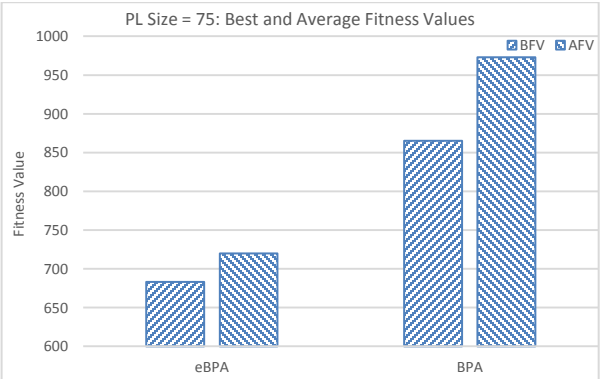


Figure 2.9: Best and average fitness values, per *PL* size of 75

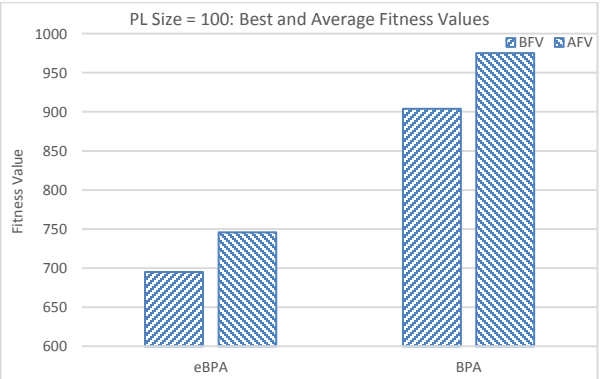


Figure 2.10: Best and average fitness values, per *PL* size of 100

The graphical comparison of the average execution time performances from Table 2.3 is seen in Figure 2.11.

Figure 2.11: Average execution time performances, for each *PL* size

For each of the BFV solutions, as documented in Table 2.3 for the different *PL* sizes, Figures 2.12 and 2.13 depict the behavior of each algorithm in converging to their best solutions. Each slope represents the fitness drop over the number of iterations executed. Each figure shows the basic behavior of the algorithms for the different *PL* sizes. The fitness values were recorded every 2,000 iterations. The plotted slopes in Figure 2.12 relate to the performances delivered by the eBPA. The plotted slopes in Figure 2.13 relate to the performances delivered by the BPA.

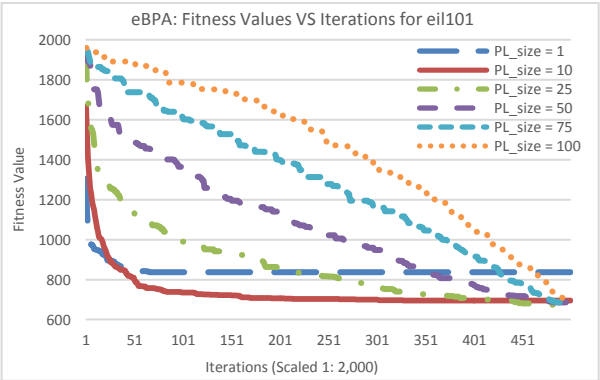


Figure 2.12: Convergence of eBPA in having determined its best solutions, per *PL* size

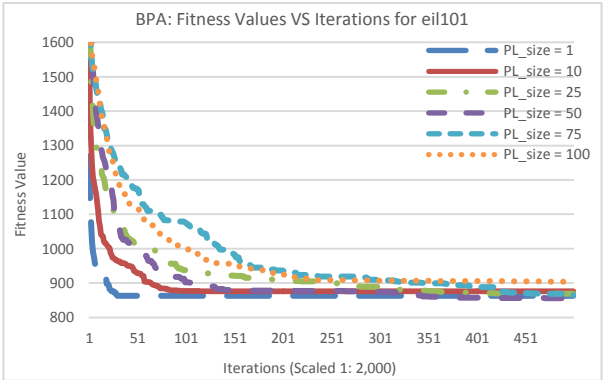


Figure 2.13: Convergence of BPA in having determined its best solutions, per *PL* size

From Figure 2.12, it is seen that all plots started off with the same initial solution. The slope with the *PL* size of 1 shows an immediate fitness drop, yet it converged quickly beyond the fitness value of 1,000; it ultimately determined a relatively poor solution. The slope with the *PL* size of 10 also shows an immediate fitness drop, yet converged at a slower rate; it determined a much improved solution compared to that with the *PL* size of 1. The slope with the *PL* size of 25 shows an even slower fitness drop. The slope with the *PL* size of 50 shows an even slower fitness drop, in being compared to that

with the PL size of 25. Interestingly, as the PL sizes increase, the slopes clearly seem to be moving away from having a concaved shape towards being linear. The slope with the PL size of 75 shows a slope which is somewhat linear. The slope with the PL size of 100 shows the slowest rate of convergence; its slope is clearly convexed.

In observing the slopes for the different PL sizes, it is distinctly realized that in order to alter the rate of convergence, one would simply need to adjust the PL size appropriately. This experiment also demonstrates that the setting of the PL size for the eBPA is relatively simple. Yet, it is also observed that reducing the convergence rate would not necessarily yield better results. However, in observing the performances of the PL sizes between the ranges of 10 to 100, it is realized that amending the PL sizes within this range does not significantly hinder the overall performance of the eBPA.

From Figure 2.13, it is seen that the BPA slopes show a similar trend in convergences, yet at much faster rates. Also, the convergences of the BPA slopes are not as predictable, or balanced, as that of the slopes of the eBPA. For example, the slope with the PL size of 50 initially weaves with the slope having the PL size of 25. Thereafter, it converged at a faster rate compared to the slope having the PL size of 25. A similar scenario is seen with the slopes with the PL sizes of 100 and 75. Hence, in making a comparison to Figure 2.12, the setting of the BPA memory structure size is not as simple as that of the eBPA. For the BPA, there is the element of added uncertainty in setting different PL sizes, as it is not completely obvious how the convergence will progress.

To further investigate, we compare the similarities of the convergences with a typical single-point metaheuristic algorithm. The algorithm investigated is the TS. TS also implements the benefits of memory.

For TS, the Tabu List size of 7 remained constant (Glover, 1986), while tests were performed in using the Candidate list sizes of 50, 100, 250, 500, 750 and 1,000. For each CL size investigated, 100 runs were performed. From the 100 runs, per CL size, the convergence of the best solution had been recorded, and is graphically depicted in Figure 2.14. Similar to Figures 2.12 and 2.13, Figure 2.14 likewise represents the fitness drop over the number of iterations executed. As can be seen, TS comparatively shows similar behavioral traits to that of the BPA, but not as comparable to that of the

eBPA. Its slopes show fast convergences for the different CL sizes investigated. They also interweave each other in progressing towards their best solutions.

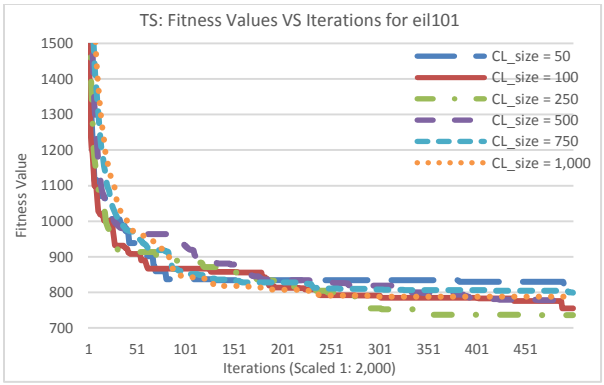


Figure 2.14: Convergence of TS in having determined its best solutions, per CL size

In comparing the eBPA convergences to that of the BPA and the TS, it is observed that the eBPA has performed somewhat similarly to that of the BPA, yet not nearly as comparable to that of TS. Similar to TS, for this problem instance, the typical metaheuristic algorithm would likely show the behavioral pattern of a quick drop in fitness, and then would dramatically slow down beyond a certain fitness point until the slope vertically flattens out. However, it is noted that the rate of convergence would also be controlled by the parameter settings of the algorithm. Yet, the behavior of the rate of convergence is unlikely to be as obvious as that experienced by the eBPA. This feature distinguishes the eBPA from the typical metaheuristic algorithm, and will be very beneficial in its implementation to practical applications.

2.6.2 eBPA Parameter Experiments

The next two sets of experiments specifically investigate the correlation between the settings of the eBPA parameter values and its performances. Fortunately, the eBPA has only two parameter values: the probability factor (p_a) and the size of the PL (PL_size).

2.6.2.1 Experiment 4

This experiment investigates the relationship between different probability values and fitness. For this experiment, 500 runs were executed in randomly selecting probability values from within the

range of $0 \leq p_a < 1$. For all simulations, the PL_size of 100 remained constant. The results determined are plotted in Figure 2.15.

In Figure 2.15, each point represents the probability-fitness relation of a single run. Figure 2.16 is a zoomed in image of Figure 2.15. From Figures 2.15 and 2.16, it is observed that as probability increases, the likelihood of the eBPA determining higher quality solutions decreases. Higher levels of probability encourage greater levels of exploration; however, this should not be beyond the point of what is ideal to experience that essential balance between exploration and exploitation. The correct balance between exploration and exploitation is needed to determine the highest quality of solutions. It is evident that smaller probability values best suit the explorative and exploitative balance.

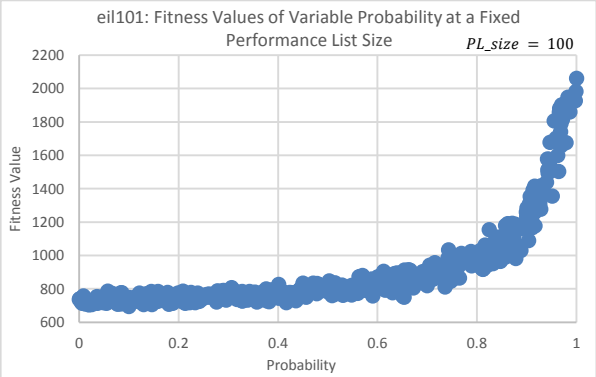


Figure 2.15: Correlation between probability and fitness

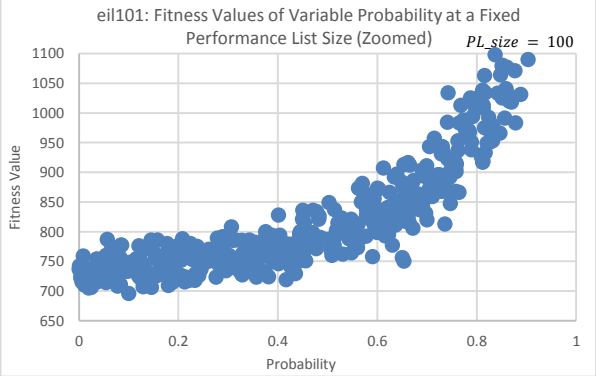


Figure 2.16: Zoomed in image of Figure 2.15

2.6.2.2 Experiment 5

For this experiment, we investigate the relationship between the *PL* size and fitness, and the *PL* size and execution time performance. For this experiment, a total of 500 runs had been executed. For each run, the p_a value of 0.2 remained constant while the *PL* size was randomly selected from with the range of $1 \leq PL\_size \leq 200$. For each run, the *PL* size, fitness and execution time had been documented. The results of the *PL* size and fitness correlation is plotted in Figure 2.17. In Figure 2.17, each point represents the coordinate of the *PL* size and fitness value relation. Figure 2.18 is a zoomed in image of Figure 2.17. Likewise, the results for the *PL* size and execution time performance correlation is plotted in Figure 2.19. Figure 2.20 is a zoomed in image of Figure 2.19.

Figures 2.17 and 2.18 show that the best and most consistent performances fall between the *PL* size range of 10 to 100. However, in increasing the *PL* size within this range, the probability of determining higher quality solutions marginally decreases. The most competitive solutions arguably fall within the *PL* size range of 20 to 50.

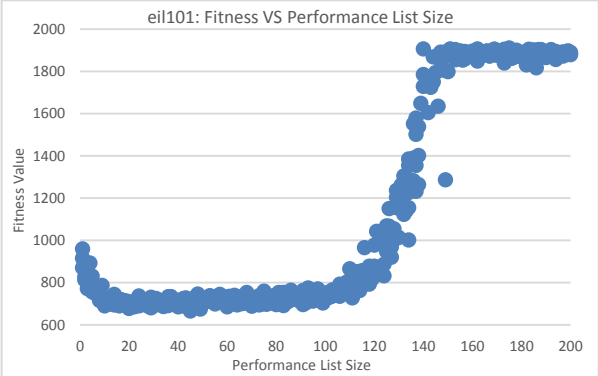


Figure 2.17: Correlation between *PL* size and fitness

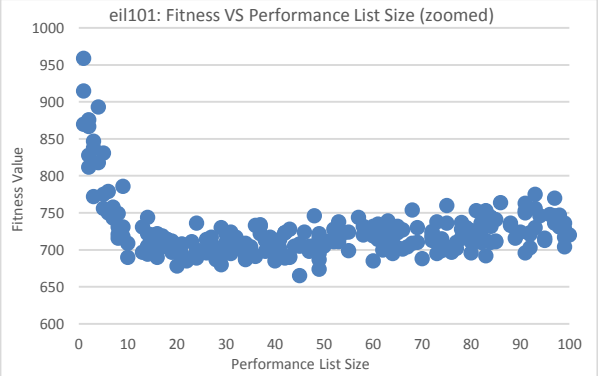


Figure 2.18: Zoomed in image of Figure 2.17

Figures 2.19 and 2.20 show that although some results are scattered (which would also be typical for other metaheuristic algorithms), there is evidence to stipulate that increasing the *PL* size will also increase the execution time performances. The relation is more clearly seen with the red-dotted trend line across the face of Figure 2.20.

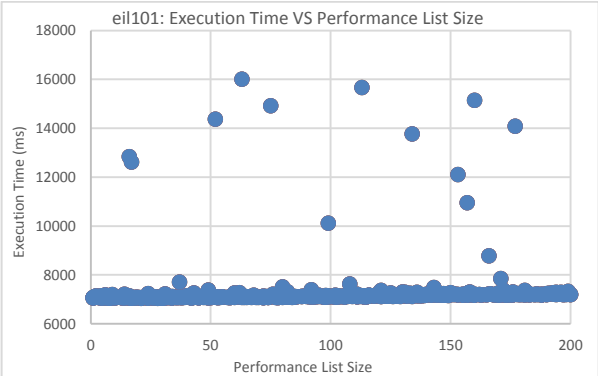


Figure 2.19: Correlation between *PL* size and execution time

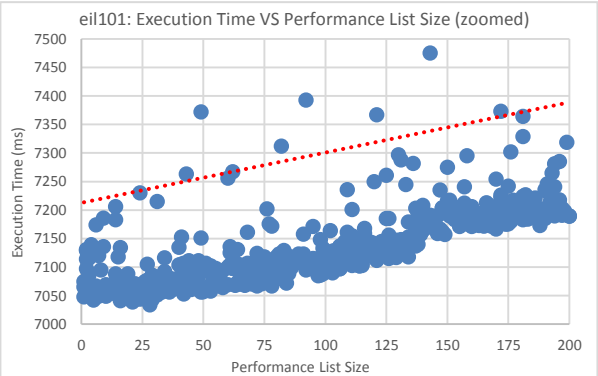


Figure 2.20: Zoomed in image of Figure 2.19

The strength of the eBPA surrounds its core feature which is adaptive memory. The techniques used to maintain this memory structure, and the governing principles of admittance surrounding it, appear as the key elements for the efficiency of the eBPA.

2.7 Conclusion

This study has presented theory and analysis on the eBPA, and has highlighted the underlying principles that govern the design of the algorithm. The eBPA had been developed due to further research having undergone on the BPA. The eBPA was developed to try and improve upon the efficiency aspects of the BPA, particularly for discrete optimization problems.

The objective of this study was to formally present and highlight the benefits of the eBPA. The eBPA core design is structured around adaptive memory. Adaptive memory is used as a tool to strategically direct the search, and also to penetrate complexed regions of the solution space which may confine other methods. The eBPA design embeds characteristics of both stochastic and deterministic search strategies. These strategies are used to finely balance its rate from exploration and exploitation.

The conceptual differences in the design techniques of both the BPA and the eBPA have been comprehensively analyzed. To compare the differences in the design techniques, a comprehensive set of experiments have been performed. The results show the strength of the eBPA in having delivered effective and economical search, compared to that of the BPA, for this discrete optimization problem.

The added advantages of the eBPA is its simplistic design; it also only has two parameter values that need to be set. The settings of the parameter values have been shown to be quite simplistic.

Chapter Three

The enhanced Best Performance Algorithm for the Annual Crop Planning Problem Based on Economic Factors

3.1 Introduction

At present, the world is faced with great challenges of water and food scarcity. Water scarcity can be described as the occurrence when the demands on fresh water exceeds its supply (Schmitz *et al*., 2007). The ever increasing world population growth contributes to this problem. As a result, there are greater demands of fresh water supply from all sectors of the industry. Major industry consumers of fresh water supply are those of the agricultural, domestic and industrial sectors. The more fresh water supplies consumed by the other sectors of the industry, the less will be available for agricultural consumption. In spite of this present challenge, the agricultural sector–being the most important sector in that it is the primary producer of food globally–has been placed under increased pressure to use fresh water supplies more conservatively (Schmitz *et al*., 2007).

Currently, it is estimated that around 70% of all fresh water supplies globally are used up by the agricultural sector. Of this, around 90% is estimated to be for consumptive use (Schmitz *et al*., 2007). Thus, if reduced volumes of fresh water is supplied to the agricultural sector, it will threaten the sustainability of food production.

In crop production, fresh water supplies are essential in order to realize optimal crop development. Optimal crop development is necessary in order to receive maximum yields. Thus, any form of water depletion in the crop development process will negatively affect crop growth; this will resultantly affect harvests, and ultimately food supplies. Food supply shortages would result in increased food prices. Increased food prices would result in increases in the costs of living. Increases in the costs of living will have a direct hand in contributing to further socioeconomic problems already faced by the world.

To alleviate these challenges, it is imperative that the agricultural sector determine scalable solutions to the problem of resource allocations in crop production. Ideally, optimized solutions are required.

Interestingly, in spite of the shortages of resources available for crop production, more returns are expected per unit of the resources utilized. This is primarily due to the increases in the population growth.

As part of the attempt to contribute to the solution, the ACP problem had been previously introduced in the literature by Chetty and Adewumi (2013b; 2013c; 2013d; 2014). The ACP problem focuses at the level of an irrigation scheme. The scope is resource allocation solutions in food crop production. Notably, no optimal solutions are guaranteed in crop production. This is due to the uncertainties of several stochastic factors that are associated with the crop production process. The stochastic factors include climatic conditions, soil characteristics, market demand and supply conditions, and cultivation practices, etc. The ACP problem aims to advise crop planners on making resource allocation decisions for the forthcoming year of crop production. The studies initiated by Chetty and Adewumi (2013b; 2013c; 2013d; and 2014) were a first attempt to present the ACP problem as an optimization problem in the literature.

Interesting studies on crop and irrigation planning, as found in the literature, include those by Mohamad and Said (2011), Sunantara and Rimirez (1997), Wardlaw and Bhaktikul (2004), Georgiou and Papamichail (2008), Sarker and Ray (2009), Adeyemo and Otieno (2010a), Adeyemo *et al.* (2010b), Pant *et al.* (2008), Pant *et al.* (2010), Raju and Kumar (2004) and Reddy and Kumar (2007). Descriptions of these articles are also given in the studies by Chetty and Adewumi (2013b; 2013c; 2013d; and 2014).

This study further expounds on the ACP problem by reformulating the ACP mathematical model. The reformulated mathematical model considers two fundamental market economic factors: the economy of scale, and the demand and supply relations. The economy of scale, and the demand and supply relations have always had a notable presence in crop production. With the economy of scale influence, crop production on a larger scale has always been more profitable, as unit costs are lower (Faris, 1961). Especially with the advent of farming technologies, such as machinery, fertilizers, irrigation practices, etc., the economy of scale influence in crop production has been considerable. Almost every aspect of modern crop production favors production on a larger scale. Concerning the market demand and supply factors in crop production, the sale of the harvests are done within a deregulated marketing environment. Therefore, in an environment where there is no governmental control over the market prices, the market prices are determined by demand and supply relations.

Therefore, for these reasons, the ACP mathematical model has been enhanced in considering these important economic factors required to provide more realistic solutions.

Furthermore, this study sees an opportunity to investigate the potentials of the eBPA and the BPA for a continuous optimization problem. The solutions determined by the eBPA and the BPA will be compared against each other and those of the TS and SA algorithms.

The rest of this chapter is structured as follows: Section 3.2 gives a detailed insight into the ACP problem. Section 3.3 formally presents the problem. Section 3.4 explains the mathematical model used to formulate the ACP. Section 3.5 describes the economy of scale, and the demand and supply relational factors. These will be mathematically implemented as part of the new ACP mathematical model. The new ACP mathematical model is presented in section 3.6. Sections 3.7 and 3.8 summarizes previous research work done on the ACP for the publications listed. Section 3.9 describes the experimental results obtained. Finally, section 3.10 draws on conclusions.

3.2 Background to ACP Problem

Crop production is a multi-staged process which includes: crop selection, land allocation, planting, crop development, harvesting, crop storage, and the marketing stages (Acquaah, 2004). Ultimately, to achieve maximum returns within a production year, effective decisions need to be made at each stage of the crop production process. Yet, this is no simple task as several stochastic factors affect the crop production process.

Notable stochastic factors include the climatic conditions, soil characteristics, the market demand and supply conditions, and cultivation practices, etc. The climatic conditions primarily include temperature, humidity, wind-speeds and rainfall (Brouwer and Heibloem, 1986). These importantly effect the rates of evaporation from the soil surface back into the earth's atmosphere. It also influences transpiration through the stomata of the crops.

Soil characteristics are those of soil texture, the soil nutrition, the soil moisture content levels, the rate of transitivity of water through the soil, etc. (Astera, 2010). The content structure of the soil texture influences the soil moisture holding capacity, and the transitivity rate. The soil moisture holding

capacity is the volume of water that can be contained within the soil. The transitivity rate is the rate at which water is absorbed by the root system of the crop. Soil nutrition is vital for optimal crop growth. Furthermore, concerning the cost of the sale of the harvests, the demand and supply conditions have a major influence (Whelan and Msefer, 1996).

During the crop selection stage (notably the first), several factors need to be considered in determining the most appropriate crops to be cultivated. Firstly, crop selection is location specific. Reason being, crops adapt well to the environmental conditions at specific geographical locations (Mustafa *et al*., 2011). Also, it is necessary that there be sufficient demand for the crops to be produced in order to be counted profitable; it should also be sustainable for the future production.

Upon the crops having been selected, decisions would thereafter need to be made concerning the resource allocations amongst the various competing crops required to be produced. Resource allocations occur at the land allocation stage of the crop production process. This embeds the scope of the ACP problem.

The intent of the ACP problem is to determine resource allocation solutions amongst the various competing crops which are required to be produced. The limited resources concerned with the ACP problem include land area, irrigated water supply, and the various costs associated with the production of each crop. The objective is to maximize the total gross profits that could be earned from the sale of the harvests at the end of the forthcoming production year.

The ACP mathematical model considers several important factors in determining scalable solutions: the area of agricultural land available for crop production, rainfall estimates, the Crop Water Requirements (CWR's) per crop, the irrigated water supply and its cost, the production costs, the crop yields (this is under the assumption of what the yields are expected to be given the previous year's statistics), the producer prices per crop, and the market demand conditions.

The area of land available for crop production can be segmented into different farm-plot types. Farm-plot types are appropriate for the production of different types of crops. For sequential cropping (which is the current scope for ACP problem), the single-crop farm plots are used to produce all the crops that grow all year around on the same farm plot. These are called perennial crops. Examples include crops such as Lucerne and fruit trees. Perennial crops can be harvested once or several times

within the year. The double-crop farm plots are likewise used to produce crops that grow in sequence of each other on the same farm plot within the production year. Examples are the summer and winter crop groups. For instance, Maize (being a summer crop) is grown in sequence with Wheat (being a winter crop) on the same farm plot within a production year in South Africa. Similarly, triple-crop farm plots are used to produce three crop groups that grow in sequence of each other on the same farm-plot, etc. Sequential cropping is a highly beneficial cultivation practice: it yields higher returns per farm plot; it provides additional protection against pests, bacteria and weed development; it adds to the nutritional value of the soil, which in turn reduces fertilization and pesticide costs (Charles, 1986).

Each crop cultivated additionally differs in CWR needs. The CWR need of each crop differ due to the diversity in crop characteristics. It also differs due to the fact that the CWR need of the same crop grown at different geographical regions may differ due to the differences in the climatic conditions. The difference between the CWR need of each crop and precipitation is the volume of irrigated water that is required for optimal crop growth throughout its lifespan.

The scheduling of irrigated water for the production of each crop is out of the scope of this study. However, the feed of fresh water supply, by either rainfall or irrigation to the crops root system need to be well planned throughout the different stages of crop development. For example, a fully grown crop would require more supply of water than a newly planted crop. Also, water supplied to the surface of the crops root system would need to be in accordance with the moisture content level of the soil. The moisture content level should ideally be maintained between wilting point and field capacity; field capacity is the maximum water holding capacity of the soil. At any volume greater than the field capacity, the crop is susceptible to root damages. Also, at any level below wilting point, the crop will no longer be able to absorb water in order to survive (Brouwer and Heibloem, 1986). If a crop suffers water stress, such as mild, moderate or severe, it will affect the physiological processes of respiration, growth, photosynthesis and reproduction within the plant. Therefore, in order to achieve the ideal water balance within the plant for optimal growth and yield, it is essential that the soil moisture content be maintained throughout the lifespan of the crop. Herein lies the importance of irrigation.

Irrigated water is primarily extracted from ground water reserves such as rivers and lakes. In being supplied to the irrigation schemes, it is accompanied by a water charge m^{-3} of the water utilized.

Production costs, coupled with crop yields and producer prices determine the profit earned. This is also in accordance with the market demand of the crop. Production costs consist of the fixed and variable costs of production. Fixed costs relate to the financial outlay irrespective of production. Fixed costs include loan repayments and other types of monthly expenses incurred in order to facilitate the crop production process. Variable costs are the costs incurred in the production of a unit of the harvest. Variable costs include that of tilling the soil, labour costs, the costs of soil nutrition, pest control, irrigated water supply costs, and harvesting costs, etc.

Production costs will differ per crop produced. This is due to crop specific cultivation practices, and the cost of it. A simple example is the cost of harvesting a crop by hand, and harvesting a crop using machinery. In harvesting using these two different ways, the costs will differ. Cultivation practices also affect the crop yields (Dukes *et al*., 2012). Apart from maintaining the soil moisture content level and the nutritional value of the soil, other factors need to be dealt with which will affect the crop yield. These include weeds, pests and bacteria which occur during the lifespan of a crop.

In crop planning, the exact estimates of production costs, yields and producer prices cannot be pre-determined. Rather, statistics from previous years of crop production are used in the ACP mathematical model. These statistics will be used to estimate the figures in attempting to quantify solutions. Amongst others, these statistics can be determined from published literature and/or consultancy services (Kantanantha, 2007). Realistically, the statistics should be location specific. The objective in determining solutions is to advise crop planners on how to better prepare for the production year ahead.

3.3 Formal Description of ACP

The ACP problem is a crop planning problem at the level of an irrigation scheme. Irrigation Schemes are large areas of farming land used for agricultural purposes. Irrigated water supplied to irrigation schemes are extracted from natural resources such as dams and rivers. They are supplied to the farm plots via canal systems (Grove, 2008). The purpose of irrigated water is to supplement the lack of fresh water supply to the crops in order to meet their CWR; this is required for optimal plant development. The lack of fresh water supply is as a result of the shortfalls and inconsistencies of rainfall.

Inconsistent rainfall patterns cause inconsistent soil moisture content levels. If these levels are not maintained, it can be detrimental to crop development. Also, due to the differences in climatic conditions, from one geographical location to the next, the CWR of the same crop could differentiate from one location to the next.

At irrigation schemes, several crops get produced at different time intervals within a production year. To maximize benefits, cultivation practices such as multi-cropping and sequential cropping are thus exercised. For the ACP problem, sequential cropping is the focus at present.

Sequential cropping is the practice of cultivating different, yet complementary types of crops, in sequence of each other on the same farm area within a production year. This is achieved without having the planting and harvesting schedules of the crops being in conflict. Thus, the distinguishing factor in identifying farming areas are the number of crops that are cultivated on it within the production year.

The objective therefore in sequential cropping, given the limited resources available for crop production, is to optimize crop production in trying to maximize the total gross profits that could be earned. The profits earned are from the sale of the harvests of all the crops produced within the production year. This is the objective of the ACP problem.

The gross profits earned are the differences between the producer prices and the production costs of the crops. Production costs consist of fixed and variable costs of production. Fixed costs are the financial outlays irrespective of crop production. The variable costs are the accumulated costs of crop production, per unit of the crop produced.

Due to several types of stochastic factors that are associated with this problem, no mathematical model exists that can determine accurate resource allocation solutions in crop production. Rather, in trying to accommodate the most important factors of this problem, without making it overly complex to solve by introducing too many variables, the ACP mathematical model had been developed. This model determines resource allocation solutions in order to assist crop planners in answering some of the following questions:

1. What is the area of land that should be allocated for producing each crop within the production year?
2. What is the volume of irrigated water that is required per crop for optimal crop development, given the land allocation?
3. What would be the cost of this irrigated water?
4. What would be the total cost of producing each crop, given the land allocation ?
5. Given the market demand and supply conditions, what would be the gross profits earned from each crop?
6. What would be the gross profit earned from producing all crops within a production year?

Hence, the ACP problem focuses on determining the resource allocation solutions related to land, irrigated water supply and the variable costs associated with crop production. It tries to maximize the total gross profits that could be earned from the production of all crops produced within a production year. The complexity to the problem is attached with determining resource allocation solutions for all crops in accommodating their different planting and harvesting schedules within the year.

Solutions for the ACP problem are determined under the following assumptions:

1. ACP solutions are determined for the land allocation stage of the crop production process.
2. ACP solutions are determined at the beginning of the production year.
3. The total area of land available for the production of each crop group is known. For the ACP problem at an existing irrigation scheme, it is considered that this area of land remains fixed due to sequential cropping practices.
4. The statistics from previous years of crop production is known. This includes information of the crop demand, the producer prices, the costs associated with production, and the yields per crop. The demand statistics will be used to estimate the lower and upper bound ranges in order to determine realistic solutions in accordance with actual demands. The pattern of the producer prices can be used to estimate what would be the producer price for the same quantity of goods demanded in the forthcoming production year. Likewise, the same can be said for the production costs; for this, it is assumed that the fixed and variable costs can be differentiated. Concerning the crop yields, it is acceptable that this statistic remains the same.

7. The cost of irrigated water m^{-3} and its supply to the irrigation scheme is known. The supply of irrigated water to the farming plots are assumed to be normal throughout the production year.
8. The CWR of each crop, at the specific geographical location, is known.
9. The average volume of precipitation throughout the lifespan of each crop is known.
10. Crop production throughout the year is assumed to be under favourable conditions. Hence, no unforeseen circumstances such as drought, hail, flooding, etc., will interfere with the crop production process.
12. Crops are planted and harvested according to schedule. These dates are assumed not to overlap with other crops grown in sequence on the same farming area of land.

For the crop demand ranges, the lower bound should be set such that the minimum market demand is satisfied. Likewise, the upper bound should be set such that an excess quantity of harvest is not produced, which would result in losses. For optimized irrigated water allocations, precipitation is considered. Also, excessive applications of irrigated water to the farming plots result in environmental damages. Therefore, to tighten the grip on excessive water wastage, producers are required to pay water charges (Grove, 2008). The strain of paying water charges, and the concern over water wastage, mean that the producers are required to produce more output per m^{-3} of irrigated water utilized.

Several objectives as well as soft and hard constraints must be achieved in order to determine feasible solutions.

Objectives:

1. Maximize the total gross profits earned from the production of all crops within the production year.
2. Determine the resource allocation solutions of land, irrigated water supply and production costs of all crops produced within the year.

Hard constraints:

1. Crop groups must be cultivated on their allocated farm plots. For example, perennial crops must grow on single-crop farm plots, only two crop groups are allowed to grow in sequence of each other on the double crop farm plots, etc.
2. Each crop must be allocated a portion of land.

3. The minimum and maximum market demand conditions must be satisfied.
4. The total volume of irrigated water allocated to each crop produced must not exceed the total volume of irrigated water that can be supplied to the irrigation scheme within a production year.

Soft constraints:

1. Give as much satisfaction to each crop being produced, such as land area and irrigated water allocation.
2. Resource allocations must be done as fairly as possible.

The ACP mathematical model presented in this study implements the market economic factors of the economy of scale and the demand and supply relations.

3.4 The Annual Crop Planning Problem as a Space Allocation Problem

The ACP problem has been mathematically modelled as an instance of the Space Allocation Problem (SAP). SAP's are amongst the hardest to solve to optimality in the literature. Interesting examples include those investigated at tertiary institutions (Silva, 2003; Adewumi and Ali, 2010), and those investigated at the level of supermarkets (Tsai and Wu, 2010; Bai, 2005), amongst others.

SAP's involve the allocation of a limited area of available space amongst the entities that demand for space utilization. In relation to the ACP problem, an entity refers to a crop which competes for land-area in order to be cultivated. With each entity competing for maximum space utilization, the complexity arises in trying to grant as much satisfaction to each entity in trying to optimize the collective benefit from the production of all crops. The common error is that the mismanagement of the limited area of space will negatively impact on the desired benefits.

Associated with these problem instances are hard and soft constraints. The hard constraints have to be satisfied. However, a maximum number of soft constraints should be satisfied if possible. The mathematical formulations commonly used to formulate the SAP's include those of bin-packing, assignment modelling, and knapsack modelling (Silva, 2003). For the ACP problem, knapsack modeling has been used.

3.4.1 Knapsack Model for ACP

The description of a knapsack is a backpack bag with shoulder straps. The ideology of knapsack modeling is to assign items, each having an associated weight and profit value, into one or many knapsacks without having exceeded the maximum weight holding capacity (or capacities). Therefore, the object is to determine a permutation that would maximize the total accumulated profits in assigning the items to the knapsack/s.

Different types of knapsack mathematical models do exist in the literature. Examples include the binary, fractional, bounded and multiple knapsack models (Nyonyi, 2010). The binary model is constrained in allowing for each item to be selected at most once; the fractional model allows for fractions of the items to be selected; the bounded model enforces boundary constraints in the selection of the items; the multiple knapsack model allows for multiple knapsacks to be filled.

Therefore, knapsack models are differentiated based on the way items are selected. Elements of different knapsack models can also be combined together to mathematically formulate a problem. For example, a mathematical formulation could require items to be selected in accordance with the binary constraint, yet may require multiple knapsacks to be filled. In terms of the ACP problem, elements of the binary, bounded, fractional and multiple knapsack models have been combined to mathematically formulate the problem.

In reference to the ACP problem, a knapsack refers to the accumulated area of land available for the production of each crop group. A crop group is a collection of crops that are produced within the same seasonal window, which in-turn are cultivated in sequence with other crop groups grown on the same farm-plot. Each knapsack (i.e. the accumulated area of land available for the production of a crop group) would require multiple crops to be cultivated on it. These individual crops are the items belonging to a knapsack. The weight factor of a crop is the area of land allocated for its production. The profit factor of a crop is the profit earned in the sale of the harvest given the area of land allocated for its production.

Given the multi-knapsack nature of the ACP problem, the objective is to determine hectare allocation solutions, for each crop being produced, such that the knapsack capacities are not exceeded in trying to maximize the accumulated profits earned. Multiple constraints exist in order to determine feasible

solutions: each crop is allowed to be selected at most once (hence, the binary element); a fraction of the maximum allowable area for the production of each crop is to be selected (hence, the fractional element); hectare allocations per crop must satisfy the lower and upper bound constraints (hence, the bounded element); lastly, since individual crops belong to specific crop groups, allocations need to be done for multiple knapsacks (hence, the multiple element). The mathematical formulation of the knapsack model used for the ACP problem is given below (Chetty and Adewumi, 2013e).

Suppose there are a total on m knapsacks of capacities k_j, where $j = 1, \dots, m$. For each knapsack k_j, items x_{ij} can be selected to fill up k_j, $\forall\, i = 1, \dots, n_j$. Each item x_{ij} has an associated weight w_{ij} and profit p_{ij} factor. Item x_{ij} is allowed to contribute a fraction f_{ij} of itself (i.e. $0 < f_{ij} \leq 1$) in being selected. For the fraction of item x_{ij} selected (i.e. $f_{ij}x_{ij}$), the lower bound Lb_{ij} and upper bound Ub_{ij} constraints must be satisfied. The maximum capacity constraint of all knapsacks k_j must be satisfied, and the total capacity of all knapsacks is T. The knapsack model template used to formulate the ACP problem is as follows:

$$\text{Maximize } f(x) = \sum_j^m \sum_i^{n_j} p_{ij} x_{ij} \tag{3.1}$$

Subject to:

$$\sum_{i=1}^{n_j} w_{ij} x_{ij} \leq k_j, \qquad for\ j = 1,2, \dots, m \tag{3.2}$$

$$x_{ij} = \begin{cases} 1 & if\ item\ x_{ij}\ is\ selected \\ 0 & otherwise \end{cases} \tag{3.3}$$

$$0 < f_{ij} < 1 \tag{3.4}$$

$$Lb_{ij} \leq f_{ij} x_{ij} \leq Ub_{ij} \tag{3.5}$$

$$\sum_{j=1}^{m} \sum_i^{n_j} w_{ij} x_{ij} \leq T \tag{3.6}$$

3.5 Economy of Scale and the Demand and Supply Relations

Economy of scale is described as the reduction in unit cost per item being produced, as the volume of output increases (Krugman, 1980). This is well researched in market economics and could occur for several reasons. Some of the primary reasons include the fixed costs per unit decrease as the

volume of items produced increase (for example, the fixed cost of ZAR[1] 100 is calculated to be cheaper per unit in producing 100 units in comparison to 10. The resultant effect of this is increased profit earned per unit of the item produced); fixed costs per unit are calculated to be cheaper in purchasing materials in larger volumes at discounted prices; the utilization of specialized equipment or machinery in manufacture result in more efficiency per unit of production, reducing costs; etc.

Demand and supply relations are also fundamental concepts in market economics. These quantify the mathematical relations between the quantity of goods demanded by the buyers, and those that are supplied by the producers at a specific market price. This price is referred to as the "equilibrium price" (Whelan and Msefer, 1996). Hence, the demand relation refers to the demand of the quantity of goods from buyers at an equilibrium price they are willing to pay. Similarly, the supply relation refer to the supply of the quantity of goods by producers at an equilibrium price at which they are willing to supply at. The demand and supply relations therefore determine the equilibrium price as agreed upon by the buyer and seller in the sale of the harvests. In exercising the trade, producers and buyers will want to maximize their profits in trading at the best possible price. An illustration of the demand and supply relation is given in Figure 3.1 below.

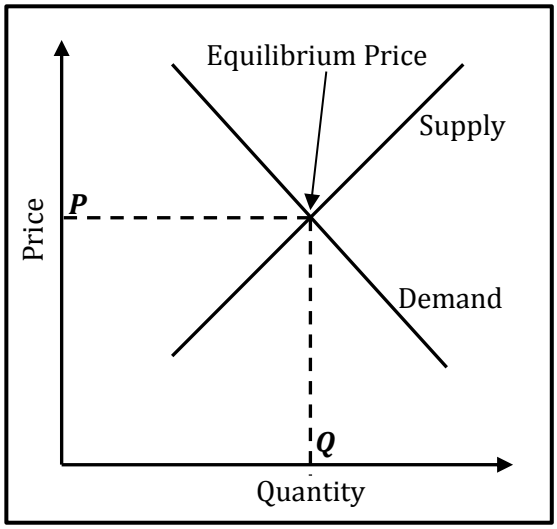


Figure 3.1: Equilibrium market price as determined by the demand and supply relations

[1] ZAR stands for Zuid-Afrikaanse Rand. It is the Dutch translation of saying, "South African Rand." The Rand is the currency in South Africa.

In Figure 3.1, P represents price and Q quantity. The equilibrium price is where P and Q intersect. This means that quantity Q will be traded at price P. As is seen, any price below P will increase demand, and any price above P will decrease demand.

In the reformulation of the ACP mathematical model, both the economy of scale and the demand/supply relational factors are considered.

3.6 ACP Mathematical Model with Economic Factors for an Existing Irrigation Scheme

This section presents the enhancement of the ACP mathematical model. This model includes the market economic factors of the economy of scale, and the demand and supply relations. Explanations on the foundational ACP mathematical models can be found in Chetty and Adewumi (2013b, 2013c, 2013d, 2014). The mathematical model in this study relates to that of an existing irrigation scheme.

To implement the economy of scale influence, a 'fixed cost' variable is introduced. Hence, production costs are now explicitly differentiated as being fixed and variable costs of production. A fixed cost factor associated with the production of each crop will encourage a higher quantity of produce as the unit cost will decrease. This will result in a higher profit earned per crop. However, this influence is challenged by the demand and supply relations, as higher yields beyond the equilibrium point will result in lower producer prices; this will equate to less profit earned per unit (and vice versa).

In this model, equilibrium price is represented in terms of hectare allocations. This is achieved by making use of either (or both) of the demand or supply relational equations. With gross profits earned being dependent on hectare allocations, it is now interesting that hectare allocations and gross profits are influenced by the economy of scale and the demand and supply relational factors. This introduces added complexity, yet allows for more scalable solutions.

The ACP mathematical model, which includes the market economic factors of the economy of scale and the demand and supply relations are as follows:

3.6.1 Mathematical Notations

Indices

- k – Plot types. (1 = single-crop plots; 2 = double-crop plots; 3 = triple-crop plots; etc.).
- i – Indicative of the crop groups that are grown in sequence of each other on the same farming plot of land within the year, on plot type k ($i = 1$ indicates the 1st crop group; $i = 2$ indicates the 2nd crop group; $i = 3$ indicates the 3rd crop group; etc.).
- j – Indicative of the individual crops belonging to crop group i, on plot k.

Input Parameters

- l – Number of different farming plot types.
- N_k – Number of sequential crop groups cultivated on plot k.
- M_{ki} – Number of individual crops cultivated at stage i, on plot k.
- H_{kij} – Hectare allocation of crop j, at stage i, on plot k as determined from the previous year.
- L_{ki} – Total area of land allocated for crop production at stage i.
- FR_{kij} – Average fraction per hectare of crop j, at stage i, on plot k, which needs to be irrigated (1 = 100% coverage, 0 = 0% coverage).
- R_{kij} – Averaged rainfall estimates that fall during the growing months for crop j, at stage i, on plot k.
- CWR_{kij} – Crop water requirements of crop j, at stage i, on plot k.
- A – Volume of irrigated water that can be supplied per hectare (ha^{-1}).
- P – Price of irrigated water m^{-3}.
- O_{kij} – Operational cost ha^{-1} of crop j, at stage i, on plot k. This cost excludes the cost of irrigated water per crop.
- F_{kij} – Fixed cost of production for crop j, at stage i, on plot k.
- YD_{kij} – The expected yield in tons per hectare (t ha^{-1}) of crop j, at stage i, on plot k.
- MP_{kij} – Producer price per ton of crop produced for crop j, at stage i, on plot k. This is the equilibrium price from the previous year of trading, at the hectares allocated. It is determined by the demand/supply relation.

- Lb_{kij} – Lower bound of crop j, at stage i, on plot k. This reflects the minimum expected market demand in order to meet supply needs. This should to be determined by the producers.
- Ub_{kij} – Upper bound of crop j, at stage i, on plot k. This reflects the maximum expected market demand. The producers should also determine this.

Calculated Parameters

- TA – Total volume of irrigated water that can be supplied to the total area of farming land within the year ($TA = T * A$).
- IR_{kij} – Volume of irrigated water that should be supplied to crop j, at stage i, on plot k. ($IR_{kij}m^3 = (CWR_{kij}m - R_{kij}m) * 10000m^2 * FR_{kij}$).
- C_IR_{kij} – The cost of irrigated water ha^{-1} of crop j, at stage i, on plot k. ($C\_IR_{kij} = IR_{kij} * P$).
- C_{kij} – Variable cost ha^{-1} of crop j, at stage i, on plot k. ($C_{kij} = O_{kij} + C\_IR_{kij}$).

Variables

- X_{kij} – Area of land, in hectares, that can be feasibly allocated for the production of crop j, at stage i, on plot k.
- AV_{kij} – Average cost ha^{-1} in considering the fixed and variable costs of production for crop j, at stage i, on plot k. ($AV_{kij} = (X_{kij}C_{kij} + F_{kij})/X_{kij}$).
- EP_{kij} – Equilibrium price that is substituted by using either the demand or supply relations, which has dependency on X_{kij} (e.g. Demand relation: X_{kij}(D) = a + bEP_{kij}; Supply relation: X_{kij}(S) = c + dEP_{kij} where a, b, c and d are constants).

3.6.2 Optimization Model

Objective Function

Maximize $f =$

$$\sum_{k=1}^{l}\sum_{i=1}^{N_k}\sum_{j=1}^{M_{ki}} X_{kij}(EP_{kij} * YD - AV_{kij})$$

$$= \sum_{k=1}^{l} \sum_{i=1}^{N_k} \sum_{j=1}^{M_{ki}} X_{kij}(EP_{kij} * YD - C_{kij}) - F_{kij} \quad (3.7)$$

Equation 3.7 gives the objective function. The fixed cost variable F_{kij} implements the economy of scale influence. The equilibrium price variable EP_{kij} (substituted in terms of hectare allocations, by using either of the demand or supply relational equations) are used to implement the market demand or supply influence. The constraints to the problem remain the same as found in Chetty and Adewumi (2013b, 2014).

Land Allocation Constraints

All solutions must satisfy the lower and upper bounds of each crop.

$$Lb_{kij} \leq X_{kij} \leq Ub_{kij} \quad \forall k, i, j \quad (3.8)$$

The summation of the land allocated for each crop j, at stage i, on plot k, must not exceed the total area of land available for crop production at stage i, on plot k.

$$\sum_{j}^{M_{ki}} X_{kij} \leq L_{ki} \quad \forall k, i \quad (3.9)$$

Irrigated Water Constraints

The summation of the volume of irrigated water allocated to each crop must be less than the total volume that can be supplied to the irrigation scheme within the year.

$$\sum_{k} \sum_{i} \sum_{j} IR_{kij} \leq TA \quad (3.10)$$

Non-negative Constraints

Arbitrarily, the lower and upper bound settings as well as the gross profits earned per crop must be non-negative.

$$Lb_{kij}, Ub_{kij}, (EP_{kij} * YD - \ AV_{kij}) > 0 \quad \forall k, i, j \tag{3.11}$$

3.7 Summary of, "On the Performance of new Local Search Heuristics for Annual Crop Planning: Case Study of the Vaalharts Irrigation Scheme[2]"

Having initially introduced the ACP problem for an existing irrigation scheme, in Chetty and Adewumi (2013b), this study investigated the potentials of three new LS metaheuristic algorithms in determining ACP solutions for the same case study. The three LS metaheuristic algorithms included the BPA, the Iterative Best Performance Algorithm (IBPA), and the Largest Absolute Difference Algorithm (LADA). These algorithms had been newly introduced in Chetty and Adewumi (2013a). To test the merits of the solutions determined by these new metaheuristics, their solutions were matched against those of TS and SA. The results concluded that from all metaheuristics, the BPA and the IBPA delivered the overall best solutions. The BPA delivered the best fitness solution, and the IBPA marginally outperformed the BPA on average.

3.7.1 The Vaalharts Irrigation Scheme Case Study

The case study investigated was that of the Vaalharts Irrigation Scheme (VIS), which is located in South Africa. Comprising of approximately 36,950 hectares of prime agricultural land, the VIS is one of the largest irrigation schemes found in the world. Figure 3.2 below shows an image of the VIS, as

[2] Chetty, S. and Adewumi, A.O. (2014). "On the performance of new local search heuristics for annual crop planning: case study of the Vaalharts irrigation scheme", Journal of Experimental & Theoretical Artificial Intelligence, Vol 27, pp. 2.

well as the neighboring Taung Irrigation Scheme (TIS). The Figure also shows the locations of the Vaal River, and the Taung Dam.

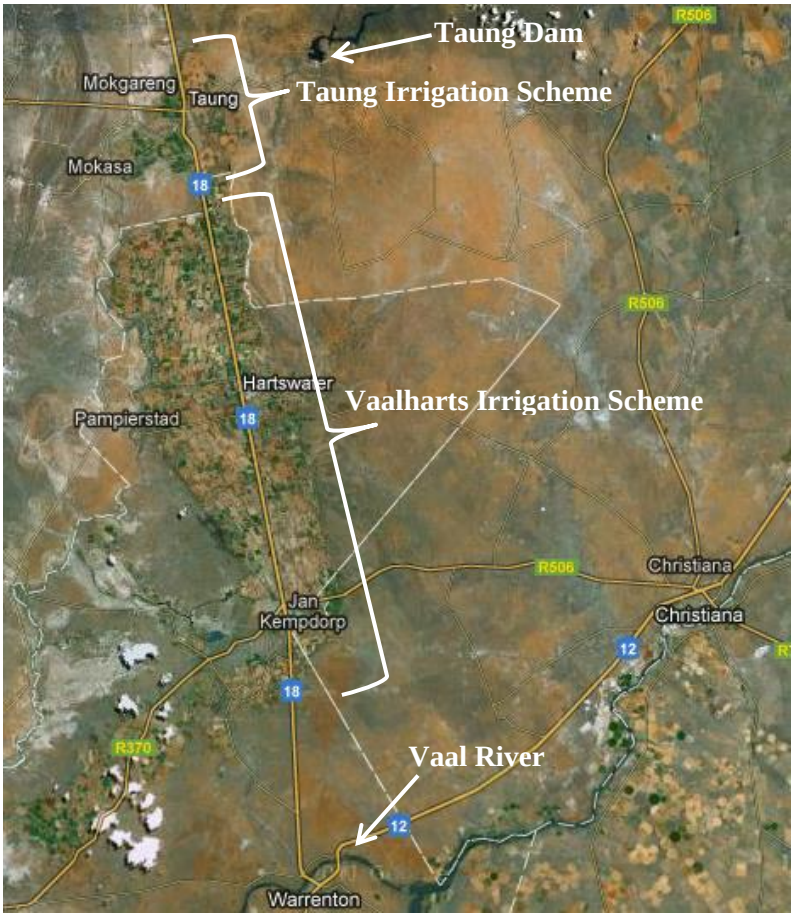


Figure 3.2: Satellite image of the Vaalharts Irrigation Scheme, Taung Irrigation Scheme, Vaal River and Taung Dam

This geographical region is known for cold and frosty winters, warm summers and irregular rainfall patterns. With irregular rainfall patterns, and having a low rainfall average of 440 millimeters (mm) annum^{-1}, irrigated water is necessary for optimized crop production at the VIS. Table 3.1 below shows the average rainfall patterns as determined over a period of 36 years.

Table 3.1: Mean rainfall statistics as determined over a 36 year period

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Mean Rainfall	75.9	63.5	71.8	51.6	19.9	9.5	4.3	8.6	11.3	24.6	45.7	58.0

The irrigated water supplied to the farm plots get extracted from the nearby Vaal River. It is supplied at a maximum rate of 9,140 $m^3\,ha^{-1}\,annum^{-1}$. A water charge of 8.77 cents m^{-3} needs to be paid to the Vaalharts Water User Association (WUA).

Table 3.2 shows the statistics of the primary crops grown at the VIS. The table lists the crop names, together with their types given in brackets; these crop types are either perennial (p), summer (s) or winter (w) crops. The table also gives the hectare allocations per crop (ha's $crop^{-1}$), the tons of yield per hectare (t ha^{-1}), the Crop Water Requirements (CWR's), the average rainfall statistics (AR), the producer prices per ton of yield (ZAR t^{-1}), the average fraction of irrigated water applied per hectare per crop with $FR_{kij} \in [0,1]$, the cost of the irrigated water per hectare (C_IR_{kij}), and the operational costs of production per crop (O_{kij}). From Table 3.2 it is calculated that the total area of land for the perennial crops, summer crops and winter crops are 8,300 ha's, 15,500 ha's, and 12,200 ha's respectively.

Table 3.2: Dataset for the Vaalharts Irrigation Scheme Case Study

Crops	ha's $crop^{-1}$	t ha^{-1}	CWR	AR	ZAR t^{-1}	FR_{kij}	C_IR_{kij}	O_{kij}
Pecan Nuts (p)	100	5.0	1,600	444.7	3,500.00	1	1,013.20	5,833.35
Wine Grapes (p)	300	9.5	850	350.8	2,010.00	1	437.80	6,365.00
Olives (p)	400	6.0	1,200	444.7	2,500.00	1	662.40	4,999.98
Lucerne (p)	7,500	16.0	1,445	444.7	1,185.52	1	877.26	6,322.72
Cotton (s)	2,000	3.5	700	386.4	4,500.00	1	275.03	5,250.00
Maize (s)	6,500	9.0	979	279.0	1,321.25	1	613.90	3,963.78
Ground Nuts(s)	7,000	3.0	912	339.5	5,076.00	1	502.08	5,076.00
Barley (w)	200	6.0	530	58.3	2,083.27	1	413.68	4,166.52
Wheat (w)	12,000	6.0	650	58.3	2,174.64	1	518.92	4,349.28

The dataset given in Table 3.2 is referenced for further research in this area. It is also the same dataset that will be used for the experimental study in the results section (i.e. section 3.9).

3.8 Summary of, "Studies in Swarm Intelligence Techniques for Annual Crop Planning Problem in a New Irrigation Scheme[3]"

In this publication, the ACP problems was further enhanced in presenting a mathematical formulation for determining solutions at new irrigation schemes. The case study investigated related to the TIS. In researching this problem, it was realized that optimized solutions at current agricultural practices were important, yet not enough to meet the future demands for food crops.

The ACP mathematical model for new irrigation schemes differ from the ACP mathematical model for existing irrigation schemes in that not only do optimized solutions need to be determined for the hectare allocations per crop, but also for the plot types within which these crops get cultivated. This added dimensionality adds to the complexity, and makes the problem interesting to solve.

To determine solutions, three relatively new SI metaheuristic algorithms were investigated. These included the CS, FA and the GSO. To benchmark the relative merits of the solutions determined by these metaheuristics, the GA was implemented. The results showed that the GSO delivered the best fitness solution, although the FA performed the best on average. It was concluded that in a solution space of constantly changing dimensions, the FA was the most consistent algorithm. However, the FA also proved to be the most costly in terms of execution time performance.

3.8.1 The Taung Irrigation Scheme Case Study

The case study that was investigated related to the TIS, which neighbors the VIS. The TIS consisted of a total of 3,764 ha's of irrigated land, yet another 1,750 ha's were being allocated for the production of 10 different crops for restitution purposes. The irrigated water supplied to the TIS is also supplied via the Vaalharts Canal System, although the Taung Dam lay nearby. Irrigated water is supplied to the TIS at a quota of 8,417 $m^3ha^{-1}annum^{-1}$. A water charge of 8.77 cents/m^3 needs to be paid to the Vaalharts Water User Association (WUA). Table 3.3 presents the crops statistics of the dataset used.

[3] Chetty, S. and Adewumi, A.O. 2013. "Studies in Swarm Intelligence Techniques for Annual Crop Planning Problem in a New Irrigation Scheme", South African Journal of Industrial Engineering, Vol 24, pp. 3.

Table 3.3 lists the Crop Water Requirements (CWR's), the average rainfall statistics (AR), the producer prices per ton of yield (ZAR t^{-1}), the expected yield per crop (t ha^{-1}), the average fraction of irrigated water applied per hectare per crop with $FR_{kij} \in [0,1]$, the cost of the irrigated water per hectare (C_IR_{kij}), and the operational costs of production per crop (O_{kij}).

Table 3.3: Dataset for the Taung Irrigation Scheme Case Study

Crops	CWR (mm)	AR (mm)	ZAR t^{-1}	t ha^{-1}	FR_{kij}	C_IR_{kij}	O_{kij}
Lucerne (p)	1,445	444.7	1,185.52	16.0	1	877.26	6,259.52
Tomato (s)	1,132	350.8	4,332.00	50.0	1	685.11	71,478.00
Pumpkin (s)	794	279.0	1,577.09	20.0	1	451.66	10,408.80
Maize (s)	979	279.0	1,321.25	9.0	1	613.90	3,924.09
Ground Nut (s)	912	339.5	5,076.00	3.0	1	502.08	5,025.24
Sunflower (s)	648	314.9	3,739.00	3.0	1	292.13	3,701.61
Barley (w)	530	58.3	2,083.27	6.0	1	413.68	4,124.88
Onion (w)	429	177.0	2,397.90	30.0	1	221.00	23,739.30
Potato (w)	365	152.8	2,463.00	28.0	1	186.10	22,758.12
Cabbage (w)	350	152.8	1,437.58	50.0	1	172.94	23,720.00

The dataset presented in Table 3.3 is referenced for future research in this area.

3.9 Experimental Results

The dataset used for this experiment is the dataset listed in Table 3.2 under section 3.7.1. This dataset relates to the VIS. Table 3.4 gives the lower and upper bound settings, the fixed costs of production (F_{kij}), as well as the demand equations used for the experiment. For the purpose of simulation, demand equations were formulated for each crop using the statistics of the equilibrium price ton^{-1} of yield (i.e. the MP_{kij}), and the hectares allocated (i.e. the H_{kij}).

The parameter settings of metaheuristic algorithms influence their performance per problem instance. Therefore, for fair algorithmic comparisons for this problem instance, experiments will be performed to determine the appropriate parameter settings for each metaheuristic algorithm. Determining the parameter settings will be the first set of experiments. Once the parameter setting for the algorithms have been determined, the second set of experiments will be performed for the algorithmic

comparisons. The *working* solution at each iteration, for all experiments will be determined as follows: randomly select a crop, and thereafter randomly select its hectare allocation.

Table 3.4: Parameter settings per crop

Crops	Lb_{kij}	Ub_{kij}	F_{kij} **(**ZAR**)**	EP_{kij} (Demand Eq.)
Pecan Nuts (p)	50	300	875,000	30*X + 500
Wine Grapes (p)	100	500	2,864,250	5*X + 510
Olives (p)	100	800	2,700,000	7*X – 300
Lucerne (p)	7,000	8,000	948,416	(2/5)*X + 1814.48
Cotton (s)	1,000	3,000	393,750	2*X + 500
Maize (s)	5,000	8,000	8,323,875	X/4 - 303.75
Groundnuts (s)	4,500	9,500	1,522,800	X/2 + 1576
Barley (w)	100	300	7,249,779.6	10*X + 83.27
Wheat (w)	10,000	15,000	1,565,740.8	X/6 + 174.64

For problem instances where the optimal solution is known, the objective in comparing algorithmic performances is to monitor which algorithm will determine the optimal solution in the shortest computational time. Therefore, with this being the intent, the parameter settings would need to be adjusted accordingly. Another alternative, in comparing algorithmic performances, is to run simulations for a fixed number of iterations. With this approach, the parameter settings would need to be adjusted to make the most effective use of the limited computational time available. One possible problem with this approach is that if the metaheuristic algorithm shows a clear convergence, in leading towards its best solution, this strategy would be ineffective if the termination were to be done before this point of convergence. Therefore, for these reasons, the stopping criterion adopted in this study is to execute the termination of the algorithms at their points of convergence.

Convergence is the point where further improvements in the solution quality would yield minimal benefits compared to the relatively large number of iterations required to yield those minimal benefits. Therefore, in this study, convergence will be detected when no further improved best solution is found for a large number of iterations. For the experiments to determine the parameter settings, a total of 30,000 idle iterations will be used to detect convergence. Thereafter, in comparing algorithmic performances, a total of 50,000 idle iterations will be used to detect convergence.

The experiments run to determine the parameter settings for the probability factor (p_a) and the PL size of the eBPA and the BPA can be seen in Figures 3.3 to 3.9 below. In Figure 3.3 and 3.7, the PL size remained fixed at 50, while the p_a values were randomly selected from within the range of $0 < p_a \leq 0.15$ for the eBPA, and $0 < p_a \leq 0.25$ for the BPA. This was per run for a total of 100 runs per experiment, in using the same initial solutions. Figure 3.4 is a zoomed in image of Figure 3.3, and Figure 3.8 is a zoomed in image of Figure 3.7. The zoomed in images show more clearly the best solutions determined.

Figure 3.3 and Figure 3.7 show that with probability factors below 0.0781 and 0.886 respectively, many solutions were determined which were found in regions that were far away from those of the best solutions found. However, it is seen that in both of these figures that there are no distinguished best values for the p_a values, as competitive solutions can be seen scattered throughout the probability ranges. This shows that irrespective of the values of the p_a's, the eBPA and the BPA would find good neighborhood regions with more consistency if the probability factors were to be greater than 0.077 and 0.885 respectively. The best solution determined for the eBPA, as seen in Figure 3.4, had a probability factor of 0.128 (truncated to three decimal places). The best solution determined for the BPA, as seen in Figure 3.8, had a probability factor of 0.121 (truncated to three decimal places). Therefore, for the rest of the experiments, the probability value of $p_a = 0.128$ will be used for the eBPA, and the probability value of $p_a = 0.121$ will be used for the BPA.

For the experiments run to determine the PL size's of the eBPA and the BPA, the probability value of $p_a = 0.128$ remained constant for the eBPA, and the probability value of $p_a = 0.121$ remainded constant for the BPA. The values of the PL size's were then randomly selected from within the range of $1 \leq PL\_size \leq 200$ for each algorithm per experiment. Again, this was per run for a total of 100 runs per experiment, in using the same initial solutions. For the eBPA, the results are seen in Figures 3.5 and 3.6. For the BPA, the results are seen in Figure 3.9. Figure 3.6 is a zoomed in image of Figure 3.5.

From Figures 3.5 and 3.6, it is seen that the most consistent performances were determined in using PL sizes within the range of 18 to 112 for the eBPA. From Figure 3.9, it is seen that the most consistent performances were determined using PL sizes greater than 132. However, it is again observed that the eBPA and the BPA determined competitive solutions throughout the PL size ranges. For the

eBPA, the best solution had a PL size of 69; this value will be used for the algorithmic performance comparison tests. For the BPA, the best solution had a PL size of 164; this value will be used for the algorithmic performance comparison tests.

With the termination criterion to be set at x (i.e. either 30,000 or 50,000) idle iterations, the strategy to be used to reduce of the PL size for the eBPA, until a size of 1 is reached, will be as follows: If half of the termination number of idle iterations have been reached (i.e. $minimum\_condition = termination\_criterion/2$), divide the remaining number of iterations by the current PL size (i.e. $reduction\_Criterion = (termination\_criterion - minimum\_condition)/PL\_size$). If the lower bound plus the reduction criterion (i.e. $minimum\_condition + reduction\_Criterion$) equates to the current number of idle iterations then reduce the PL size by 1. The reduction of the PL has the dual purpose of increasing exploitation, as well as eliminating the possibilities of cycling for PL sizes greater that one.

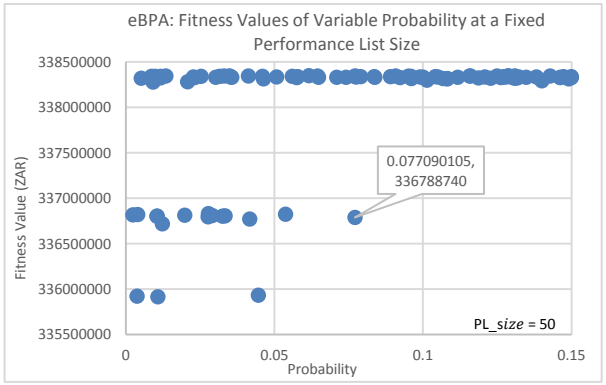


Figure 3.3: Fitness values determined using randomly selected probability factors at a fixed PL size of 50

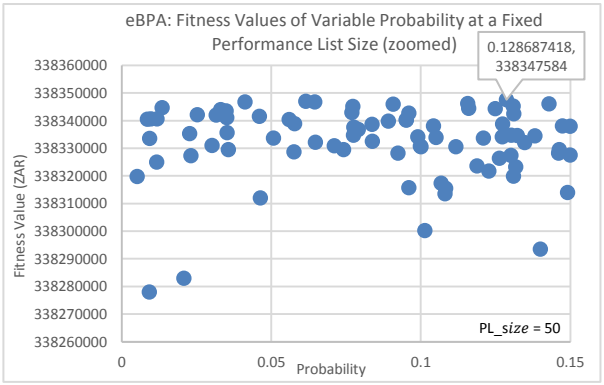
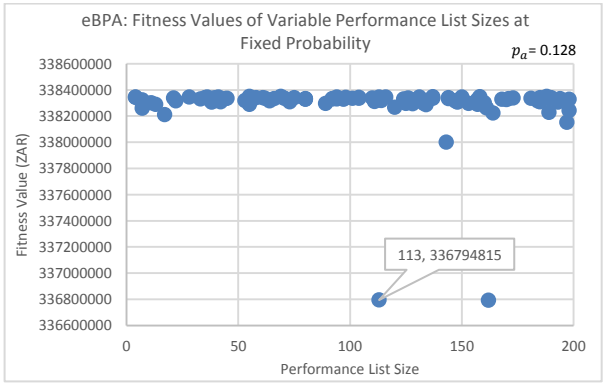


Figure 3.4: Zoomed in image of Figure 3.3



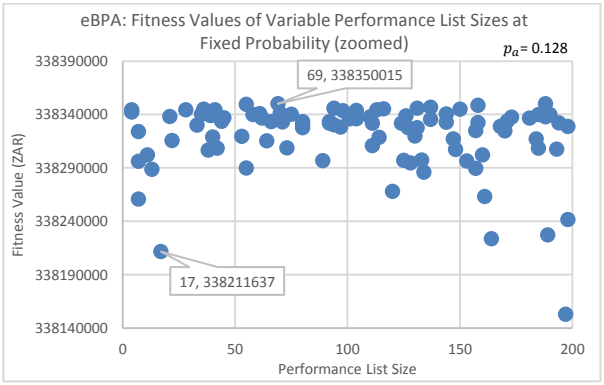


Figure 3.5: Fitness values determined using randomly selected *PL* sizes at a fixed probability factor of 0.128

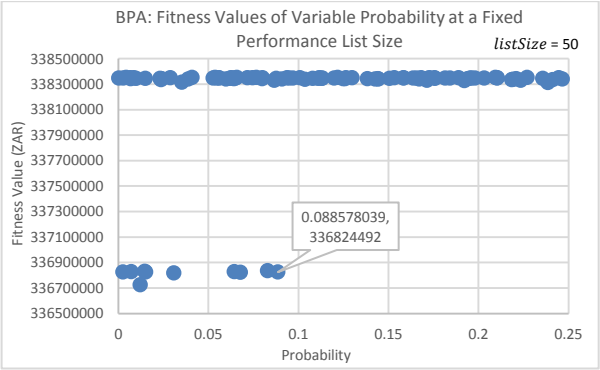


Figure 3.6: Zoomed in image of Figure 3.5

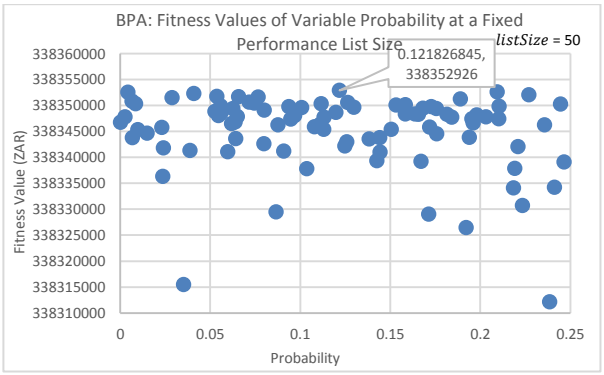


Figure 3.7: Fitness values determined using randomly selected probability factors at a fixed *PL* size of 50

Figure 3.8: Zoomed in image of Figure 3.7

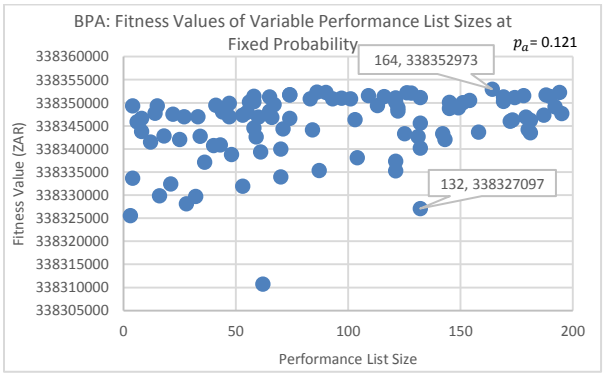


Figure 3.9: Fitness values determined using randomly selected *PL* sizes at a fixed probability factor of 0.121

The experiments run to determine the parameter settings for SA are seen in Figures 3.10 and 3.11 below. In Figure 3.10, the initial temperature T was fixed at 100, while the cooling factor α had been randomly selected from within the range of $0.95 \leq \alpha < 1$. This was done per run for a total of 100 runs in using the same initial solution. The cooling factor α controls the rate of convergence, and decreases T using the equation $T = T * \alpha$. Therefore, the higher the value of α, the slower the rate of convergence, and the more successful the annealing process will be. From Figure 3.10, it is observed that the fitness qualities of the solutions were similar in having found similar neighborhood regions. The best value of α seen is 0.96 (rounded off to two decimal places).

The value of $\alpha = 0.96$ remained fixed for the experiment related to Figure 3.11. In this experiment, the initial temperature T was randomly selected from within the range of $1 \leq T \leq 500$. This was done per run for a total of 250 runs in using the same initial solution. More runs were needed to determine T, as T importantly controls the transition from exploration to exploitation. The parameter settings for SA are more difficult to determine, and would explain the volume of research done on SA. From Figure 3.11, it is seen that the best solution for T was 226. Together with $\alpha = 0.96$, these will be the parameter settings to be used for SA in performing the algorithmic comparison tests.

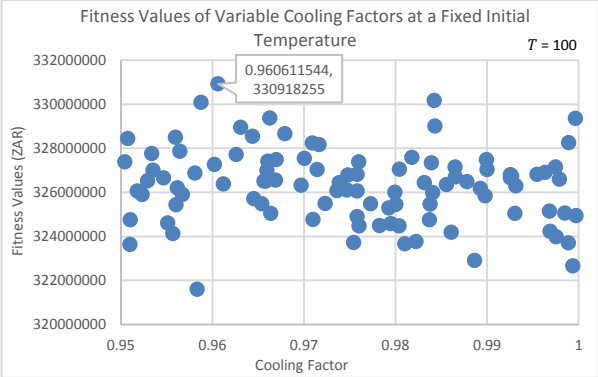


Figure 3.10: Fitness values determined using randomly selected cooling factors, at a fixed initial temperature of 50

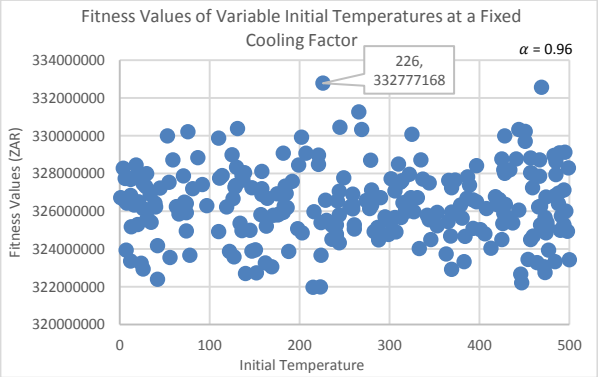


Figure 3.11: Fitness values determined using randomly selected initial temperature values, at a fixed cooling factor of 0.96

The experiments run to determine the CL size for TS is seen in Figures 3.12 and 3.13. Figure 3.13 is a zoomed in image of Figure 3.12. For this experiment, a recommended TL size of 7 was used (Glover, 1986). CL sizes were randomly selected from within the range of $1 \leq CL\_size \leq 500$. This was done per run for a total of 100 runs in using the same initial solution.

Figure 3.12 shows that CL sizes above 209 determined solutions that had fitness values which were far from the best solution found. The best solution found, as seen more closely in Figure 3.13, had a CL_size of 34. Figure 3.13 also shows a cluster of competitive solutions found around the CL_size of 34. This indicates that a size of 34 is a good value to choose. These values are the parameter settings that will be used for the TS in performing the algorithmic comparison tests.

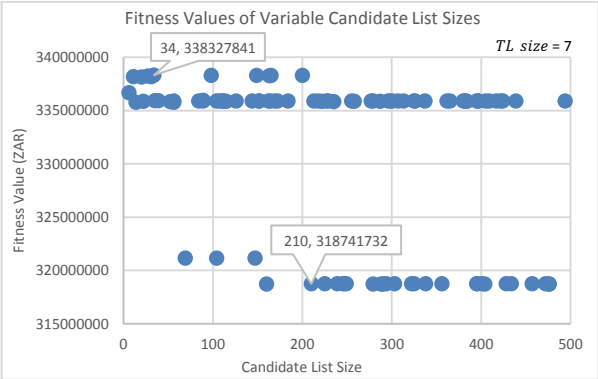


Figure 3.12: Fitness values determined by randomly selecting the *CL* size values

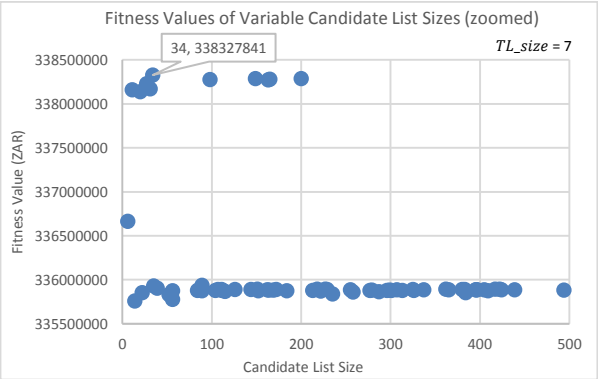


Figure 3.13: Zoomed in image of Figure 3.12

As can be seen from Figures 3.3, 3.5, 3.7 and 3.9 the parameter settings for the eBPA and the BPA did not significantly hinder its performances. This is an interesting observation in being compared to an algorithm such as SA which requires more effort to set its parameter values.

For the second experiment, in comparing the algorithmic performances, the parameter settings determined from the first set of experiments were used. For this experiment, a total of 50 runs per metaheuristic algorithm were executed. The termination criterion was 50,000 idle iterations. For each of the 50 runs, per algorithm, the same initial randomly generated solution was passed in as an input parameter to each algorithm. The experiments performed, together with these test criterion, were sufficient to ensure fair algorithmic comparison tests. From the 50 solutions determined by each algorithm, their overall best and average solutions are documented. Their 95% Confidence Interval[4] values are also documented for their fitness values.

Table 3.5: Average execution time performances (AVG) in milliseconds (ms)

Methods	**AVG (ms)**
BPA	218,093
eBPA	148,178
TS	52,367
SA	33,029

[4] The Confidence Interval (CI) indicates the reliability of an interval estimate of population parameters. 95% CI means to be 95% certain that the population parameters will lie within the interval estimate range.

In Table 3.5, the average execution times reflect on the number of best solutions found by each metaheuristic algorithm. Reason being, each time the best solution had been improved upon, the counter for the idle number of iterations had been reset. As can be observed, the BPA and the eBPA best solutions were improved upon significantly more times than TS and SA. However, the BPA did find more improved solutions over that of the eBPA. The BPA and the eBPA were thus intelligent in finding more promising neighborhood regions within the confines of the solution space. This was followed by TS and then SA.

Table 3.6 gives the statistical values of the overall best and average fitness value solutions (i.e. BFV and AFV respectively). The 95% CI values are also given, along with the initial solution (IS). The fitness value refers to the total gross profit earned.

Table 3.6: Statistics of the best and average fitness values solutions, along with the 95% CI values

Methods	**BFV (ZAR)**	**AFV (ZAR)**	**95% CI**
IS	290,775,157	N/A	N/A
BPA	338,353,400	338,349,798	AFV $\pm$ 725
eBPA	338,351,684	338,345,193	AFV $\pm$ 1,203
TS	338,340,881	337,493,100	AFV $\pm$ 261,742
SA	330,721,884	327,791,514	AFV $\pm$ 425,002

It is observed that each algorithm determined best solutions that improved upon the initial solution (IS). The BPA marginally determined the best BFV and AFV solutions over the eBPA, and had the lowest 95% CI value. This was then followed by the TS and SA algorithms. The BPA BFV solution determined a gross profit of ZAR 1,716, ZAR 10,803, ZAR 7,629,800 and ZAR 47,576,527 more than that of the eBPA, TS, SA and the IS respectively. Graphical comparisons of the metaheuristic statistics as given in Table 3.6 is seen in Figure 3.14 below. The 95% CI values are represented as the black interval estimates over the average fitness value towers.

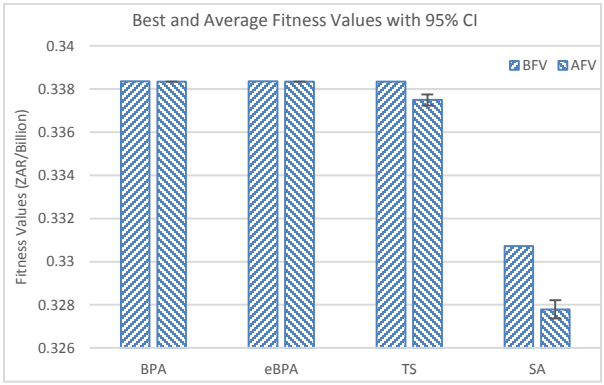


Figure 3.14: The best and average fitness values, along with their 95% CI estimates

Visually, it is seen that the differences between the best fitness value performances of the BPA, eBPA and TS were minimal. Yet on average, the BPA and the eBPA performed significantly better than TS. The BPA has also shown more consistency in having determined the lowest 95% CI estimate. This was only a marginal improvement over that of the eBPA. Having determined the best BFV and AFV solutions, along with the lowest 95% CI value, concludes that the BPA was the strongest and most consistent metaheuristic algorithm for this problem instance. However, the BPA overall performance was only marginally better than that of the eBPA for this continuous optimization problem.

The strengths of the BPA and the eBPA are attributed to their techniques employed in maintaining the solutions registered in their memory structures. The PL structures of both algorithms maintain a limited number of the best solutions found, at any given time, while traversing throughout the solution space. This maintenance is based on the idea of allowing solutions that meet the minimum criterion to be allowed admittance into the PL memory structures. The minimum criterion is that the fitness value of the worst solution must at least be improved upon with regards to the BPA, or at least be met with regards to the eBPA. If the admittance criterion of each algorithm were to be satisfied, then the design variables of the new solutions must be unique to be allowed admittance. Updates of the PL's are then performed by replacing the worst solution in the memory structures with that of the new. Thereafter, for the BPA, the sorted order of the memory structure must be maintained. For the eBPA, the indices referencing the $best$, $working$ and $worst$ solutions would need to be re-determined. These techniques, along with the strategy of their probability factors in attempting to escape local entrapment, and the strategic reduction of the PL size for the eBPA, have shown to be an effective blend in traversing the solution space effectively for this problem instance.

Table 3.7: Statistical values of the irrigated water requirements (IWR) and the costs of production (CP)

Methods	IWR (m^3)	CP (ZAR)
IS	244,491,000	156,924,202
BPA	241,997,367	154,799,322
eBPA	241,997,311	154,799,423
TS	241,998,185	154,799,348
SA	242,760,335	154,985,403

Table 3.7 gives the statistical values of the irrigated water requirements (IWR), and that of the costs of production (CP). As can be observed, each algorithm determined improved irrigated water allocation solutions over that of the IS. Interestingly, the CP values were also lower although the gross profit margins were higher.

From all algorithms, the eBPA determined a solution that required the least volume of irrigated water. The eBPA determined a solution that required a volume of 2,493,689 m^3 less than that of the IS. This was followed by the BPA, which required a volume of 2,493,633 m^3 less. Thereafter, TS required a volume of 2,492,815 m^3 less. Finally, SA required a volume of 1,730,665 m^3 less. These solutions conform to the objective of yielding higher returns per unit of irrigated water consumed. At the quota of 9,140 $m^3ha^{-1}annum^{-1}$, these savings would be able to supply irrigated water to an additional 272.83, 272.82, 272.7 and 189.3 hectares of agricultural land by the eBPA, BPA, TS and SA algorithms respectively. A visual representation of the irrigated water allocation solutions is seen in Figure 3.15 below.

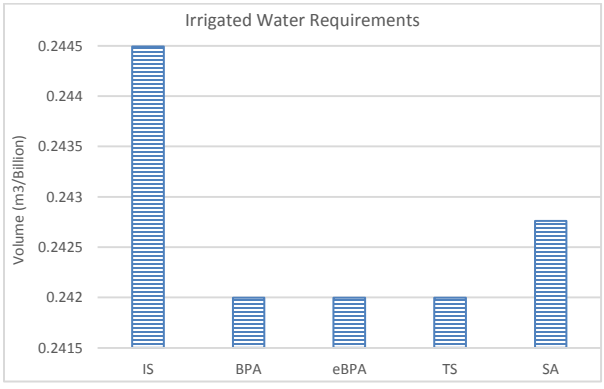


Figure 3.15: Irrigated water requirements (IWR) of the initial solution (IS) and that of the metaheuristic solutions

Figure 3.16 shows graphical comparisons of the hectare allocation solutions. The BPA, eBPA and the TS show to have determined similar solutions. The metaheuristic solutions are also seen to be comparable to that of the IS due to the constraints of the lower and upper bound settings.

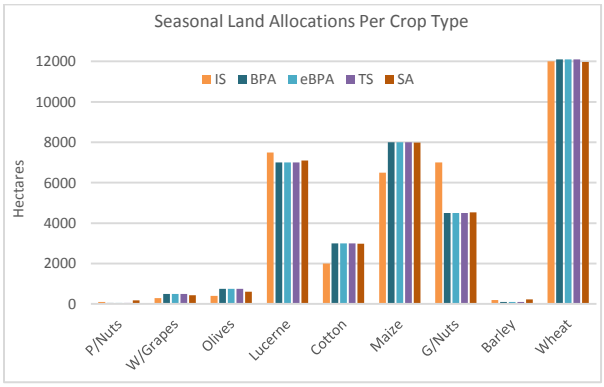


Figure 3.16: Comparison of the hectare allocation solutions per crop

The statistics of the hectare allocations (ha crop^{-1}), IWR, and the CP values of the initial and that of the best metaheuristic solutions are seen in Tables 3.8 and 3.9 below.

Table 3.8: Statistics of the initial (IS) and metaheuristic solutions per crop

Crops	**Methods**	**ha's crop^{-1}**	**IWR (m^3)**	**CP (ZAR)**
Pecan Nuts	IS	100	1,155,300	597,153.143
	BPA	50	577,650	254,826.6
	eBPA	50.003	577,685.304	254,847.493
	TS	50.001	577,662.84	254,834.181
	SA	174.722	2,018,562.036	1,108,738.936
Wine Grapes	IS	300	1,497,600	1,849,889.52
	BPA	499.971	2,495,856.1	3,210,253.1
	eBPA	499.995	2,495,977.51	3,210,418.552
	TS	499.751	2,494,757.158	3,208,755.529
	SA	430.796	2,150,534.609	2,739,669.671
Olives	IS	400	3,021,200	2,114,959.24
	BPA	750.029	5,664,967.7	4,096,961.8
	eBPA	749.99	5,664,672.134	4,096,740.2
	TS	750.215	5,666,375.011	4,098,016.827
	SA	604.264	4,564,003.826	3,271,581.702

Table 3.9: Statistics of the initial (IS) and metaheuristic solutions per crop

Crops	Methods	ha's crop^{-1}	IWR (m^3)	CP (ZAR)
Lucerne	IS	7,500	75,022,500	40,722,449.25
	BPA	7,000	70,021,000	37,122,431
	eBPA	7,000.012	70,021,117.62	37,122,515.7
	TS	7,000.033	70,021,327.18	37,122,666.53
	SA	7,090.218	70,923,452.63	37,772,005.34
Cotton	IS	2,000	6,272,000	9,475,054.4
	BPA	3,000	9,407,999.8	15,000,081
	eBPA	2,999.988	9,407,960.899	15,000,012.71
	TS	2,999.828	9,407,459.508	14,999,129.36
	SA	2,987.453	9,368,653.092	14,930,759.94
Maize	IS	6,500	45,500,000	23,809,100
	BPA	7,999.995	55,999,965	30,675,552
	eBPA	7,999.944	55,999,604.87	30,675,316.6
	TS	7,999.779	55,998,450.03	30,674,561.4
	SA	7,986.315	55,904,203.44	30,612,928.84
Ground Nuts	CP	7,000	40,075,000	32,193,977.5
	BPA	4,500.005	25,762,529	18,248,800
	eBPA	4,500.069	25,762,894.54	18,249,155.67
	TS	4,500.394	25,764,754.36	18,250,967.76
	SA	4,526.232	25,912,678.59	18,395,095.89
Barley	IS	200	943,400	791,047.98
	BPA	100	471,700.98	333,026.84
	eBPA	100.001	471,707.002	333,032.689
	TS	100.001	471,703.748	333,029.529
	SA	224.294	1,057,994.541	902,319.617
Wheat	IS	12,000	71,004,000	45,370,570.8
	BPA	12,100	71,595,699	45,857,390
	eBPA	12,099.999	71,595,691.22	45,857,383.66
	TS	12,099.999	71,595,695.3	45,857,387.02
	SA	11,975.706	70,860,252.72	45,252,302.99

3.10 Conclusion

This study further contributes to the recently introduced ACP problem in the literature. In this study, a new mathematical formulation for the ACP problem has been presented. It is based on the market economic factors of the economy of scale, and the demand and supply relations.

The objective of the ACP problem is to optimize resource allocation solutions in crop planning. The ACP problem was motivated due to the increased concerns of water scarcity, and that of the other limited resources available for crop production. In spite of the limited resources made available, more output is expected per unit due to increases in food demands. The ACP problem is a relevant problem in crop planning, within the agricultural sector.

In determining solutions, the BPA and the eBPA have been investigated. Their solutions were compared against those of the TS and SA algorithms. To ensure fairness in performing the algorithmic comparisons, experiments were run to determine the appropriate parameter settings for each of the metaheuristic algorithms. The termination criterion of each algorithm was a fixed number of idle iterations. This represented the point of convergence.

The results show that the techniques employed by the BPA and the eBPA were very effective, for this continuous optimization problem. The BPA marginally determine the overall best solutions over that of the eBPA.

Chapter Four:
The enhanced Best Performance Algorithm for the Travelling Salesman Problem

4.1 Introduction

The Travelling Salesman Problem (TSP) is defined as the problem of finding the minimal tour which traverses a list of n cities in a way in which every city is visited exactly once, except for the original city of departure where the salesman would start and finish. The problem is accounted to Euler in 1759, who presented a problem of trying to move a knight to every block on a chest board exactly once. The problem gained fame in a handbook written by B. F. Voigt in 1832 (Michalewicz, 1994). It was afterwards mathematically formulated by mathematicians W. R. Hamilton and Thomas Penyngton Kirkman. Detailed descriptions of the mathematical formulations are given in the book titled "Graph Theory 1736-1936" (Biggs *et al.*, 1986).

Several classifications of TSP's exist. The most notable are the symmetric Travelling Salesman Problem (sTSP), asymmetric Travelling Salesman Problem (aTSP), and multiple Travelling Salesman Problem (mTSP) (Matai *et al*., 2010). For sTSP's, the distance traveled from city i to city j is the same as the distance travelled from city j to city i. In graph theory, this constitutes a bi-directional graph. aTSP's on the other hand are characterized by directed graphs, i.e. the distance travelled from city i to city j will not necessarily be the same as the distance travelled from city j to city i. The practical significance of this problem include problems with one-way streets, traffic collisions, and the differences between the arrival and departure fees from airports, amongst others. mTSP's are the problems of finding the minimum tour of m travelling salesman, who start and finish at the same city in having every intermediate city visited exactly once.

This study implements instances of sTSP's, in investigating the abilities of the eBPA. The TSP is defined as follows (Lin, 1965);

Given a set of n cities, a maximum travelling distance D, and for every pair of adjacent cities v_i and v_j a travelling distance $d(v_i, v_j)$ ($\forall\, i, j = 1,2,\dots,n$). The objective is to find a permutation $P = v_{p_1}, v_{p_2}, \dots, v_{p_n}$ that minimizes the travelling distance in satisfying the constraint,

$$\sum_{1 \le i < n} d(v_{p_i}, v_{p_{i+1}}) + d(v_{p_n}, v_{p_1}) \le D \tag{4.1}$$

A straightforward approach to the sTSP is to compute all possible permutations in determining the optimal tour for an n city problem. However, the number of permutations is $n!$, with the running time being O($n!$). The problem is thus exponential, relative to n. Also, the total number of possible routes covering all cities is (n-1)!/2. As a startling example, it would take an estimated time of 5×10^{48} years to determine the optimal solution for a problem of size n = 50, if it were run on a mainframe computer executing 100 million instructions per second (Yan *et al.*, 2012).

The TSP is one of the most studied problems in discrete optimization. Mathematicians took particular interest in the 1930's; by the 1960's, the problem gained increased popularity. Due to the practical applicability of TSP's, and the complexity involved with determining optimal solutions, this problem has been significantly researched.

The complexity of the problem was proved by Richard. M. Karp (Karp, 1972). Karp proved the NP-Completeness of the Hamiltonian Cycle problem, implicitly proving the NP-Hardness of the TSP. This gave explanation to the apparent difficulty of determining optimal solutions.

Since then, large numbers of exact and approximate algorithms have been developed to determine solutions to TSP's. Nowadays, problem instances of up to 89,500 cities have been solved to optimality. Also, problems with cities into the millions have been solved to near-optimality using approximation techniques (Johnson and McGeoch, 1997). The TSP is also used as a standard benchmark problem in comparing the performances of optimization techniques, which is the purpose of this study.

The rest of this chapter is structured as follows. Section 4.2 describes examples of real-world applications of TSP's. Section 4.3 give descriptions of previous research work. Section 4.4 lists the

ten benchmark test instances to be implemented. Section 4.5 gives an overview of the investigation to follow. Section 4.6 presents and discusses the experimental results. Finally, section 4.7 draws conclusions and outlines possible future work.

4.2 TSP Applications

The TSP problem has practical significance in the real-world. This accounts for the interest in the problem. To-date, several commercial TSP solver applications have been developed. These solvers play significant roles in the industry, were time and cost factors are important. One of the biggest applications of the TSP is in transportation. A simple example of this is the scheduling of stacker cranes in warehouses. Brief descriptions of other real-world applications are given below.

4.2.1 NASA Starlight Space Interferometer Program

A team of engineers at Hernandez Engineering in Houston, and at Brigham Young University, studied the problem of trying to optimize the sequence of celestial objects to be imaged. The objective was to minimize the usage of fuel, in performing targeting and imaging maneuvers for the pair of satellites involved in the mission. In this problem, celestial objects were represented as cities, and distance was the quantity of fuel needed to reposition the two satellites from one image to the next (Bailey *et al.*, 2001).

4.2.2 Circuit Board Problems

In the circuit board problem, a machine operating on a circuit board would need to be programmed in a way to complete a set of tasks at different co-ordinate positions on the board. After completing the set of tasks, the machine would returned to its starting position before another board is set for another set of tasks to be completed.

One example of this is the circuit board construction and board cutting problem. Here, a set of cut and add operations are performed on a circuit board at different co-ordinate positions, before another board is set for a similar set of operations to be performed. The problem of optimizing the time taken to perform these operations, at different co-ordinate positions, is a TSP (Matai *et al.*, 2010).

Another related circuit board problem is the drilling of holes on a circuit board. On a circuit board, several holes would need to be drilled at different co-ordinate positions and of different diameters. Holes of the same diameter can be drilled together in one task. After the task is completed, the head of the machine is reset for drilling the next set of holes of a different diameter. This problem was modeled as a series of TSP's. Here, for each different diameter, cities represented the co-ordinate positions of each hole to be drilled, and the distance between these co-ordinate positions was the time taken to move from one hole to the next (Grötschel *et al.*, 1991).

4.2.3 Nozzle Guide Vane Placement Problem

When the turbine engine of an aircraft is overhauled, a detailed inspection can be performed upon dis-assembling the engine. Of particular interest is the turbine section of the engine which contains nozzle-guide vanes (or simply vanes) that accelerate, deflects and distributes the flow of gas that drives the turbine motor. The more efficient the distribution of gas about the circumference of the turbine motor, the more efficient will be its performance. The benefits of uniform gas distribution include reduced engine vibrations and reduced fuel consumption. The problem is that due to very high temperatures, and the velocities of the gas flow, the vanes would wear out and would need to be refurbished or replaced. Each vane has individual characteristics for the distribution of the gas, which is affixed about the circumference of the turbine engine. Therefore, upon replacing the damaged vanes (with either new or refurbished vanes, which may or may not be identical), the operator is faced with the challenge of sequencing the set of vanes about the circumference of the nozzle in an attempt to attain uniformity in the gas flow. The problem of the correct placements of the vanes have been modeled as a TSP (Plante *et al.*, 1987).

4.2.4 Order Picking Problem

This problem is associated with collecting a list of items which are stored at a warehouse. Upon receiving an order, the warehouse dispatches a vehicle to collect the list of stored items. The objective of the problem is to minimize the distance travelled by the vehicle in collecting all items. In relation to the TSP, the location of an item is represented as a city, and distance is the distance travelled between items. The problem of finding the shortest route is thus a TSP (Ratliff and Rosenthal, 1983).

4.3 Algorithmic Approaches and Previous Research

The two main factors in choosing an algorithm to solve a TSP is execution time and the quality of a tour. Due to the exponential time complexity involved with TSP's, exact algorithms are preferable for smaller instances which can be solved within polynomial time complexity (P). However, for larger instances, where the optimal solution cannot be determined within P, heuristic methods are preferable. Common heuristics for TSP's include the tour construction and tour improvement heuristics (Hjertenes, 2002).

Tour construction methods seek to construct a valid TSP tour from an unordered list of cities. The algorithm stops when a solution is found, and does not attempt to improve upon it. These algorithms run relatively fast and are believed to determine solutions within 10%-15% of the optimal solution. Ideally, they are used as inputs to local search heuristic algorithms. Popular tour construction algorithms include the Nearest Neighbor (NN), Greedy, Clarke Wright and Christofides algorithms (Davendra, 2010).

Tour improvement algorithms start off with a completed tour; it is preferable if the completed tour were to be generated by a tour construction heuristic. It then attempts to improve on this solution by searching the neighborhood regions of the solution space in trying to find improved solutions. It stops when the optimal solution has been found, or when the stopping criteria is satisfied. Several local search approximation algorithms have been studied for the TSP. The most successful include: 2-opt, 3-opt, λ-ops and Lin-Kernighan (LK) (Davendra, 2010). λ-opt algorithms involve iteratively removing λ edges, and replacing these with different edges in reconnecting the tour. The objective is to find shorter tours without cycles. LK is a λ-opt heuristic which dynamically determines suitable values for λ, per iteration. Most λ-opt moves can be performed as sequential moves. The simplest non-sequential move is the 4-opt move, which is called the double-bridge move (Lin and Kernighan, 1973).

Metaheuristic algorithms differ from pure heuristic algorithms by accepting dis-improved solutions in escaping local entrapment (Glover, 1990). The intelligence of accepting dis-improved solutions could lead the search to other neighboring regions, which may possibly contain higher quality

solutions. Several metaheuristic algorithms have been investigated for TSP's. Common examples include: the GA, Evolutionary Algorithms (EA's), TS, SA, ACO, PSO and the FA.

Dorigo and Gambardella (1997) investigated the effectiveness of applying an artificial Ant Colony System (ACS) to instances of sTSP's and aTSP's. Test instances ranged from 30 to 577 cities. The ACS results were compared against the results of SA, NN, Self-Organization Map (SOM), Evolutionary Programming (EP), GA and a hybridization of SA and GA which is called the Annealing-Genetic Algorithm (AG). Results showed that in using the 3-opt technique, ACS determined results as good as or even better than that of the other methods.

Tsai *et al.* (2004) presented an evolutionary algorithm called Heterogeneous selection Evolutionary Algorithm (HeSEA) for solving large instances of TSP's. HeSEA was developed integrating Edge Assembly Crossover (EAX) and LK through family competition and heterogeneous pairing selection. HeSEA was tested on 16 large instances of TSP's ranging from 318 to 13,509 cities. The results of HeSEA was compared against six other algorithms including SA, ACO, the Voronoi-crossover Genetic Algorithm (VGA), the Compact Genetic Algorithm (CGA), Iterated LK (ILK) and TS hybridized with LK called TLK. Results showed that HeSEA performed very competitively, and executed faster in being compared to the other algorithms.

Kumbharana and Pandey (2013) investigated the FA for six instances of TSP's ranging from 10 to 51 cities. FA was compared against ACO, GA and SA in determining solutions. Results showed that FA outperformed the other algorithms in determining the best solutions for all six test instances investigated.

Louis and Tang (1999) presented and interactive GA by implementing a divide and conquer technique for determining solutions to instances of TSP's. The divide and conquer technique had been investigated due to the standard GA being computationally expensive for this problem. The technique was used to divide the problem into smaller sub-problems, being solved separately, and then recombined later to determine a final solution. The study showed that this technique significantly reduced computation time compared to the standard GA. It also determined high quality solutions for TSP's ranging from 51 to 1084 cities.

Tasgetiren *et al.* (2007) presented a Discrete Particle Swarm Optimization (DPSO) algorithm for determining solutions to sTSP instances ranging from 51 to 442 cities. Results were compared against four heuristics, one exact and one metaheuristic algorithm. The heuristic algorithms included the GI heuristic, NN, FST-Lagrangian and FST-Root. The exact algorithm was the Branch and Cut procedure (B&C). The metaheuristic algorithm was GA. Results showed that DPSO, GA and FST-Root determined the best performances.

Yan *et al.* (2012) presented a new PSO algorithm for minimizing the possibility of local entrapment. The investigation was performed due to the weakness of population based metaheuristic algorithms such as PSO and GA getting stuck in local optima. Ten sTSP's were investigated. Results showed that PSO performed more efficiently compared to GA in determining the best results for all test instances.

Miki *et al.* (2003) presented a new SA algorithm which determined the maximum temperature setting dynamically rather than using a static maximum temperature setting. This study was motivated due to the difficulty of setting the maximum temperature parameter value for SA. This temperature, together with the minimum temperature parameter value, importantly controls the acceptance criterion in accepting dis-improved solutions. The algorithm presented, which was called the Adaptive Simulated Annealing (ASA) algorithm, was tested in performing investigations on ten TSP's which ranged from 59 to 280 cities. The results showed that ASA was an effective technique in considerably speeding up execution time performances without losing result quality in being compared to that of the standard SA algorithm.

Yao (1992) presented a SA algorithm which dynamically reduced the neighborhood sizes in relation to the temperature decreases, in determining solutions. The study was motivated due to SA generally consuming much computational time in determining good solutions for difficult optimization problems. The study was to investigate dynamic reduction of neighborhood sizes, in comparing computational time to that of the standard SA algorithm. The results showed that the SA with dynamic neighborhood size reductions outperformed the standard SA algorithm with fixed neighborhood sizes, in execution time performance and result qualities, for the test instances of TSP's.

Malek *et al.* (1989) investigated the abilities of SA and TS in serial and parallel simulation settings, for seven instances of TSP's. The problem instances ranged from 25 to 100 cities. The results showed

that TS consistently outperformed SA in the parallel environment in comparing execution time and tour-length solutions.

Tsubakitani and Evans (1998) researched determining appropriate Tabu List (TL) sizes for TSP instances. Test instances ranging from 20 to 100 cities were investigated. The conclusion was that TL sizes should be small enough to encourage exploitation, yet large enough to escape local entrapment. A comparison of different TL sizes were investigated. The results showed that smaller TL sizes have an advantage over larger TL sizes during earlier stages of the search.

4.4 Benchmark Test Instances

A popular library of TSP benchmark test instances is the TSPLIB collection which has been made available online by Gerhard Reinelt. This collection is freely accessible.

The TSPLIB collection consists of several classes of benchmark datasets. These include: sTSP's, Hamiltonian Cycle Problem's (HCP's), aTSP's, Sequential Ordering Problem's (SOP's) and Capacitated Vehicle Routing Problem's (CVRP's). Many of these problems are based on examples from printed circuit boards, VLSI applications, as well as the actual geographical locations of various cities. For majority of these test instances the optimal tour-length is known, and in some cases the optimal tour is also given. Therefore, based on different problem classifications, various levels of complexity per problem, and the fact that the optimal tour-lengths are given for many test instances, this collection has become popular amongst researchers in being used to compare performances of optimization techniques. The largest test instance in this collection consists of 85,900 cities.

This study investigates ten sTSP benchmark test instances from this collection. The problem instances, along with their characteristics, is given in Table 4.1. For each problem instance, the name, the number of vertices, the distance calculation type, and the optimal tour-lengths are given.

The distance between the adjacent vertices of these test instances are calculated on a Euclidean 2D-(EUC_2D) plane. The distance d_{ij} between adjacent vertices i and j is therefore computed to be;

$$d_{ij} = round\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{4.2}$$

In equation (4.2), $round$ is a function that rounds to the nearest integer.

Table 4.1: Symmetric Travelling Salesman Problem test instances, and their characteristics

No.	sTSP	No. of Vertices	Type	Optimal Tour-Length
1	ch130	130	EUC_2D	6,110
2	ch150	150	EUC_2D	6,528
3	rat195	195	EUC_2D	2,323
4	tsp225	225	EUC_2D	3,916
5	a280	280	EUC_2D	2,579
6	lin318	318	EUC_2D	42,029
7	pcb442	442	EUC_2D	50,778
8	d493	493	EUC_2D	35,002
9	rat575	575	EUC_2D	6,773
10	d657	657	EUC_2D	48,912

4.5 Discussion

The eBPA, TS and SA algorithms are all single-point metaheuristic algorithms. However, the differences in the fundamentals of their designs will cause each algorithm to traject differently throughout the solution spaces.

SA is a memory-less search technique; it stochastically moves throughout the solution space primarily based on randomization. TS is a memory-based search technique; it uses the advantage of memory to intelligently select the next solution from within a local neighborhood region in advancing the search. On the other hand, the eBPA takes advantage of the benefits of both randomization and memorization in proceeding with the search. Thus, the eBPA lay in-between the memory-less and stochastic search techniques such as the SA, and the memory-based search techniques such as the TS. Based on this truth, together with investigations into the potentials of the eBPA being yet in the initial stages, the eBPA will be compared against TS and SA for the test instances given in Table 4.1.

Generally, in employing metaheuristic algorithms to solve optimization problems, the parameter settings of the algorithm would need to be set appropriately. However, the versatility and strength of

an algorithm can also be seen in its ability to determine solutions for multiple problem instances in using the same parameter settings. Algorithms that are more versatile may be particularly beneficial in scenarios where the parameter settings of the algorithm cannot be tweaked; for example, the applications that run in a fully automated mode. The investigation done in this study is performed from this perspective: use the same parameter settings for each metaheuristic algorithm investigated, in solving multiple instances of the sTSP's.

The problem instances differ in complexity, and range from 130 to 657 cities/vertices. This will provide sufficient challenges to the algorithms for testing purposes. Each algorithm will be tested in their abilities to determine solutions in using the same parameter settings. The strength of the solutions determined by the eBPA will shed light on its abilities, in being compared to TS and SA. More importantly, insight will be given on the eBPA ability to balance exploration, during the initial phases of the search, and exploitation, during the final phases of the search.

4.6 Results and Discussion

The BPA will not be investigated in this chapter due to its weakness to discrete optimization problems; this has been demonstrated in chapter two. In chapter two, it was proved that the BPA performed very poor for a discrete optimization problem, even for a relatively simple instance of the sTSP. For the following investigation on the sTSP's, only the eBPA, TS and SA metaheuristic algorithms will be investigated.

To solve the problem instances, we first employ the Nearest Neighbor (NN) tour construction heuristic. This heuristic is used to provide the initial solution to each metaheuristic algorithm, per problem instance. The NN heuristic is straightforward: it is implemented by starting off at the first city, and thereafter it moves to the nearest adjacent unvisited city. The NN tour-length solutions are given in Table 4.2. A tour-length solution is also referred to as the fitness value.

Table 4.2: Nearest Neighbor tour-length solutions for each problem instance

No.	sTSP	Nearest Neighbor Tour-Length
1	ch130	7,579
2	ch150	8,191
3	rat195	2,752
4	tsp225	5,030
5	a280	3,157
6	lin318	54,019
7	pcb442	61,979
8	d493	41,665
9	rat575	8,605
10	d657	61,627

In executing the metaheuristic algorithms, the solution at each iteration is determined by selecting the best of six moves. The best move is the one that will result in the lowest fitness value. The six moves employed are as follows:

1. 2-opt – The 2-opt move removes two edges from a complete tour. It then reconnects the tour by introducing two new edges, which join the opposite ends of the removed edges. An illustration is given in Figure 4.1.

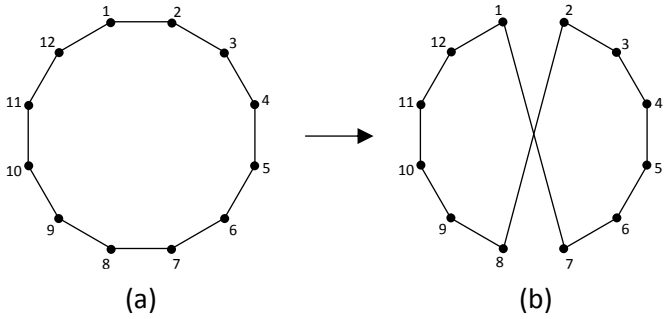


Figure 4.1: 2-opt: (a) shows the completed tour; (b) shows that edges (1, 2) and (7, 8) have been removed, while two new edges (1, 7) and (2, 8) have been introduced in reconnecting the tour

2. 3-opt – The 3-opt move is similar to the 2-opt, except that with 3-opt three edges are removed instead of two. It is implemented as two sequential 2-opt moves; this results in *two* solutions. An illustration is given in Figure 4.2.

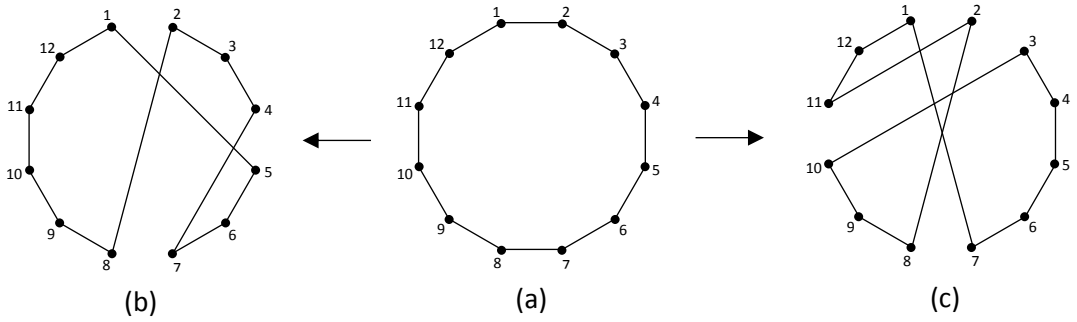


Figure 4.2: 3-opt: From initial solution (a), solutions (b) and (c) are determined by performing the first 2-opt move, in removing edges (1, 2) and (7, 8). Thereafter, to determine solution (b), edge (4, 5) is removed while keeping the removed edge (1, 2) constant in performing the second 2-opt move. Similarly, to determine solution (c), edge (10, 11) is removed while keeping the removed edge (7, 8) constant in performing the second 2-opt move

3. Double-bridge – The double-bridge move is a non-sequentially move (unlike 3-opt), which is implemented by randomly dividing the completed tour into four segments; the tour is then reconnected in the reverse order. An illustration is given in Figure 4.3.

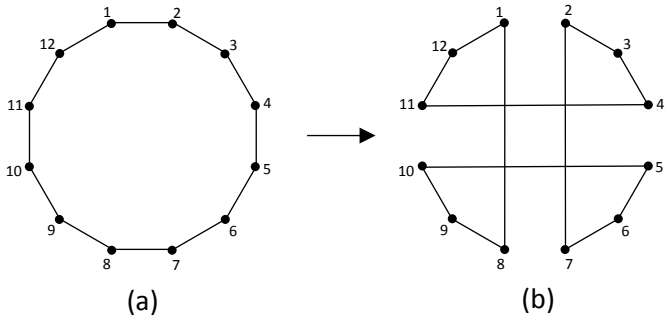


Figure 4.3: Double-bridge move: (a) shows the completed tour; (b) shows that edges (1, 2), (4, 5), (7, 8) and (10, 11) have been removed. The tour is then reconnected by introducing edges (1, 8), (10, 5), (7, 2) and (4, 11)

4. Random swap – This move is implemented by randomly selecting two vertices from a complete tour, and then swapping them. An illustration is given in Figure 4.4.

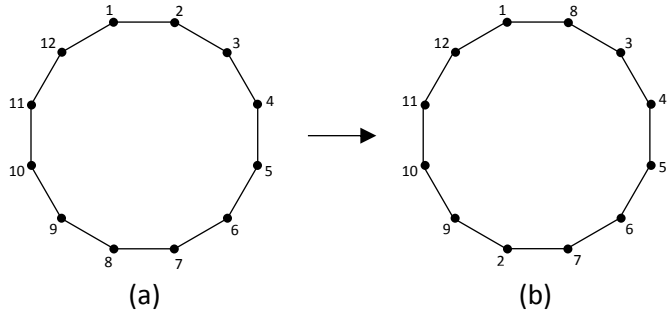


Figure 4.4: Random swap move: (a) shows the completed tour; (b) shows that vertices 2 and 8 have been swapped

5. Vertex reposition – This move is implemented by repositioning a randomly selected vertex at a randomly selected position in the tour. An illustration is given in Figure 4.5.

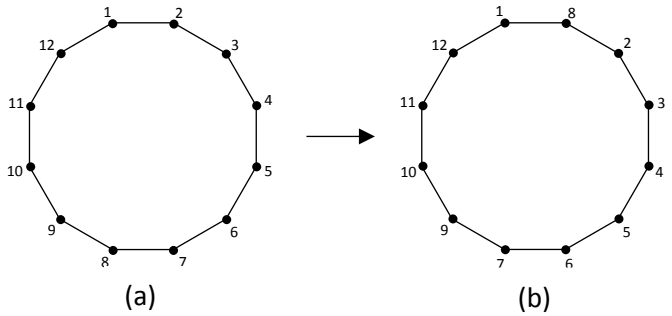


Figure 4.5: Vertex reposition move: (a) shows the completed tour; (b) shows that vertex 8 has been repositioned at location 2

The parameter settings for the algorithms will be as follows:

a) The parameter settings of SA will be according to the recommendations from the literature (Soubeiga, 2003): The initial temperate (T) will be set at 50% of the fitness of the initial solution, while the cooling rate alpha (α) will be set at 85%.

b) Likewise, the TL size of TS will be set at 7 (Glover, 1986; Malek *et al.*, 1989). To determine the CL size for TS, we make use of the test instance pr439 from the TSPLIB collection; pr439 is also a sTSP which has its distance calculated on the Euclidean 2D-plane. Determining the CL size for TS, and the parameter settings for the eBPA will be the first set of experiments.

c) Since this is the first research on the eBPA for multiple problem instances on the TSP problem, we make use of the pr439 problem to determine its parameter settings. The parameter settings are for the probability factor (p_a) and the PL size.

Once the set of experiments are run to determine the remaining parameter settings, all parameter settings will remain constant for the second set of experiments. The second set of experiments will be to compare the performances of the algorithms in using the same parameter settings, for the multiple sTSP instances to be investigated.

For the first and second sets of experiments, the stopping criterion will be to terminate the execution at the point of convergence. In this study, convergence will be detected when no further improvements have been made to be $best$ solution for a large number of iterations. For the first set of experiments (i.e. to determine the parameter settings) convergence will be set at 3% of idle iterations. For the second set of experiments (i.e. in making algorithmic comparisons) convergence will be set at 5% of idle iterations. The termination criterion will apply provided that a minimum of 10^6 iterations have been executed. For example, if 10^6 iterations have executed, and the total number of consecutive idle iterations is 50,000 (assuming we are referring to the second set of experiments), then the algorithms will detect convergence and will terminate.

For the first set of experiments, each algorithm will be run 50 times to determine each parameter value in using the pr439 problem. For the second set of experiments, each algorithm will be run 30 times per problem instance. 30 runs are sufficient in considering the large computational times required; for example, for the u724 vertices problem the estimated execution time would have been around 30 hours.

In using this termination criterion, the strategy to be used to reduce the PL size of the eBPA, until a size of 1 is reached, is as follows: Calculate the total number of idle iterations required to detect termination (i.e. $termination\_criterion = x\%\ (where\ x = 3\%\ or\ 5\%) * total\_number\_of\_iterations$). If half of the termination criterion has been reached (i.e. $minimum\_condition = termination\_criterion/2$), divide the remaining number of iterations (i.e. $termination\_criterion - minimum\_condition$) by the current PL size (i.e. $reduction\_criterion = (termination\_criterion - minimum\_condition)/PL\_size$). If

the minimum condition plus the reduction criterion equates to the current number of idle iterations then reduce the PL size by 1.

The experiment run to determine the CL size for TS is seen in Figure 4.6. The CL size's were randomly selected from within the range of $1 \leq CL\_size \leq 1{,}000$. This CL size range was considered due to the NN solutions being used as the input to each algorithm per problem instance. For this reason, greater levels of exploitation were required and thus a larger CL size range was considered. For the 50 runs, the TL size remained constant at size 7. Figure 4.6 shows that CL size's below 200 determined weaker solutions, and that the most competitive solutions fell within the range of 400 to 1000. The best solution seen had a CL size value of 723. The CL size value of 723 will be the parameter value to be used for the second set of experiments.

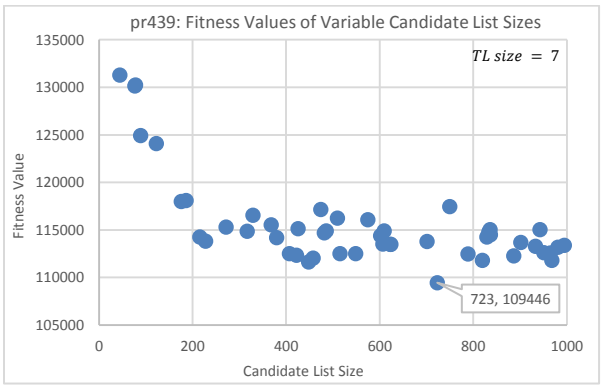


Figure 4.6: Fitness values determined by randomly selecting the *CL* size values

The experiments run to determine the probability factor (p_a) and the PL size values for the eBPA is seen in Figures 4.7 and 4.8. In Figure 4.7, the PL size remained fixed at 50, while p_a was randomly selected from within the range of $0 < p_a \leq 0.15$. From Figure 4.7 it can be seen that, for the 50 runs, the solutions are scattered throughout the entire probability range and roughly within the same height of the fitness range. There is no specific value for the p_a that is best suited for this problem instance in favoring more competitive solutions. The best solution seen had a p_a value of 0.045 (truncated to three decimal places). This will be the value used for the rest of the experiments.

For the experiment run to determine the PL size (as seen in Figure 4.8), the value of $p_a = 0.045$ remained constant, while the value of the PL size was randomly selected from within the range of $1 \leq PL\_size \leq 200$. From the 50 runs, it can be seen that values greater than 130 determined the

poorest solutions. The solutions determined in having used the values between that ranges of 50 to 130 determined competitive solutions; however, these solutions also show evidence of having determined slightly weaker solutions. The most consistent and competitive cluster of solutions can be seen within the value range of 4 to 31. The best solution determined had a *PL* size value of 10. This value, together with $p_a = 0.045$, are the parameter values that will be used for the eBPA in the second set of experiments.

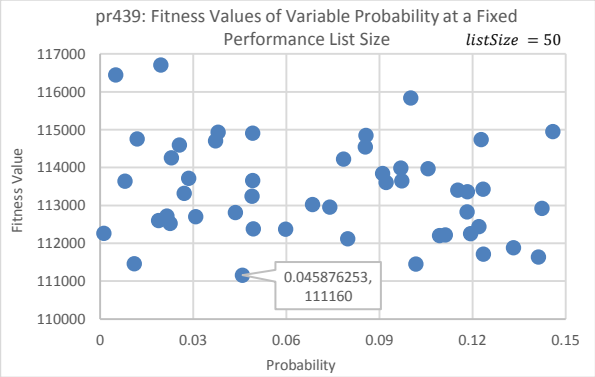


Figure 4.7: Fitness values determined using randomly selected probability factors, at a fixed *PL* size of 50

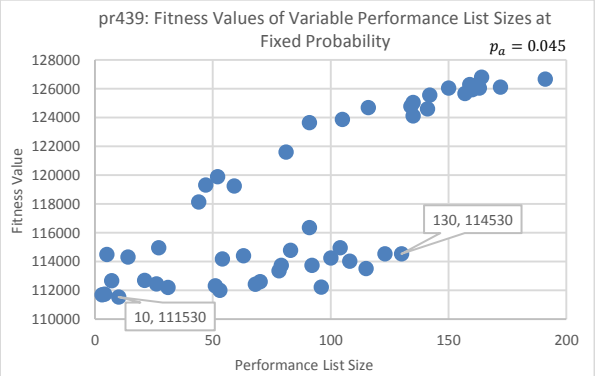


Figure 4.8: Fitness values determined using randomly selected *PL* sizes, at a fixed probability factor of 0.045

For the second set of experiments, the parameter values of all algorithms remained constant for all the problem instances implemented. For each problem instance, each algorithm was run 30 times. As mentioned previously, 30 runs per algorithm was sufficient due to the large computational times consumed per run. For each algorithm per problem instance, their best and average fitness value solutions (i.e. BFV and AFV respectively) have been documented. For the average fitness values, their 95% CI values are also given. The statistical results of the runs are given in Table 4.3.

Table 4.3: Best, average and 95% Confidence Interval fitness values, for each algorithm per problem instance

sTSP	eBPA			TS			SA		
	BFV	**AFV**	**95% CI**	**BFV**	**AFV**	**95% CI**	**BFV**	**AFV**	**95% CI**
ch130	6,144	6,261	AVG ± 26	6,208	6,305	AVG ± 57	6,124	6,228	AVG ± 24
ch150	6,563	6,643	AVG ± 23	6,563	6,664	AVG ± 23	6,543	6,683	AVG ± 30
rat195	2,330	2,359	AVG ± 2	2,356	2,375	AVG ± 4	2,379	2,429	AVG ± 10
tsp225	3,971	4,011	AVG ± 8	3,988	4,034	AVG ± 8	3,993	4,064	AVG ± 14
a280	2,637	2,677	AVG ± 7	2,654	2,705	AVG ± 7	2,638	2,726	AVG ± 13
lin318	43,233	43,685	AVG ± 126	43,492	44,310	AVG ± 156	43,485	44,340	AVG ± 197
pcb442	51,519	52,400	AVG ± 146	52,257	53,071	AVG ± 161	52,584	54,148	AVG ± 263
d493	35,862	36,235	AVG ± 78	36,565	35,977	AVG ± 95	36,422	37,136	AVG ± 145
rat575	6,955	7,062	AVG ± 18	7,024	7,093	AVG ± 14	7,206	7,290	AVG ± 20
d657	50,475	51,048	AVG ± 107	50,564	51,492	AVG ± 141	51,699	53,076	AVG ± 189

From Table 4.3, it is observed that the eBPA determined the best BFV solutions for all problem instances, except for ch130 and ch150. For ch130, SA performed the best overall. For ch150, SA determined the best BFV solution, yet the eBPA determined the best AFV solution. For all other problem instances, the eBPA determined the best BFV and AFV solutions, together with the lowest 95% CI values. Visual representations of the statistics given in Table 4.3 are seen in Figures 4.9 to 4.18. For the AFV solution towers, the 95% CI values are represented as the black interval estimates at the top.

For convenience, these figures also display the optimal solutions. The purpose of these experiments were to test the sequences of instructions constituting the algorithmic designs of each metaheuristic algorithm. It was also to test the abilities of each metaheuristic in using the same parameter settings across multiple problem instances. Therefore, the executions were not to explicitly seek out the optimal solution, yet to terminate at the point of convergence in monitoring the algorithmic performances.

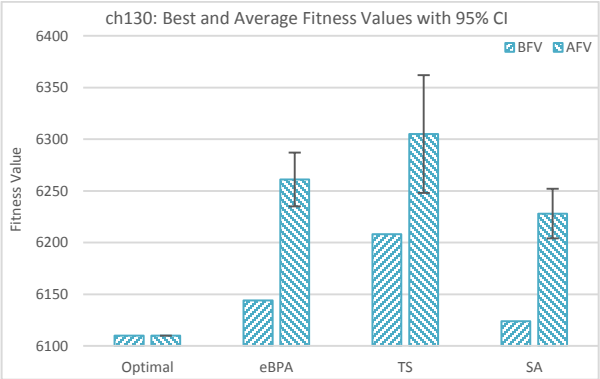


Figure 4.9: The best and average fitness values, along with their 95% CI estimates for ch130

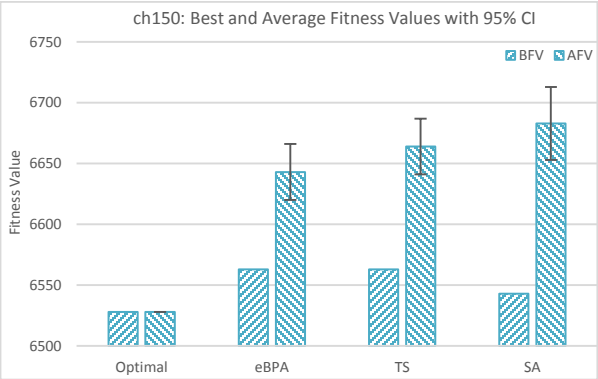


Figure 4.10: The best and average fitness values, along with their 95% CI estimates for ch150

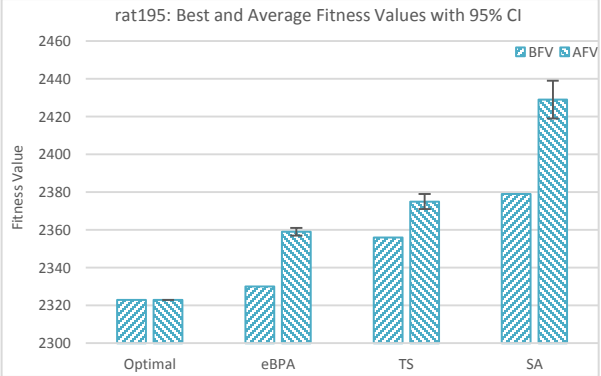


Figure 4.11: The best and average fitness values, along with their 95% CI estimates for rat195

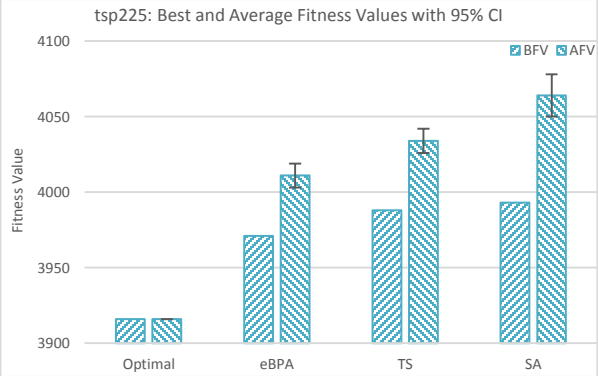


Figure 4.12: The best and average fitness values, along with their 95% CI estimates for tsp225

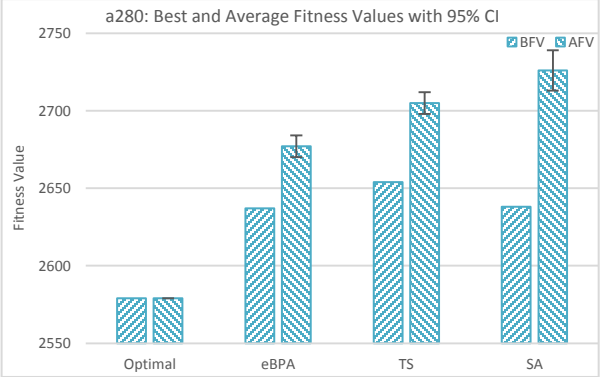


Figure 4.13: The best and average fitness values, along with their 95% CI estimates for a280

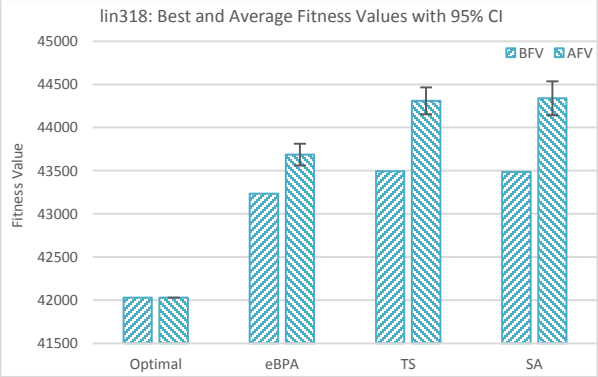


Figure 4.14: The best and average fitness values, along with their 95% CI estimates for lin318

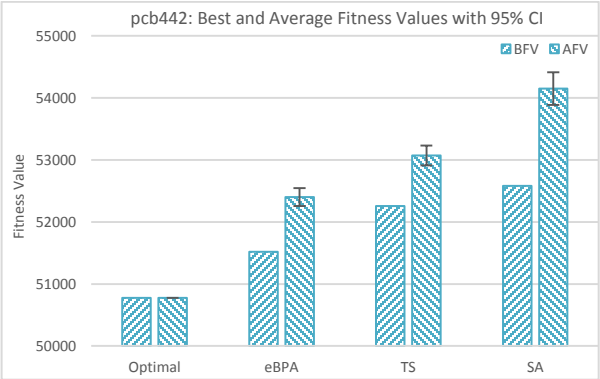


Figure 4.15: The best and average fitness values, along with their 95% CI estimates for pcb442

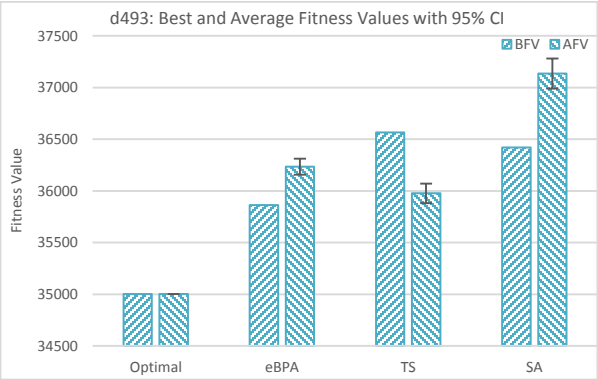


Figure 4.16: The best and average fitness values, along with their 95% CI estimates for d493

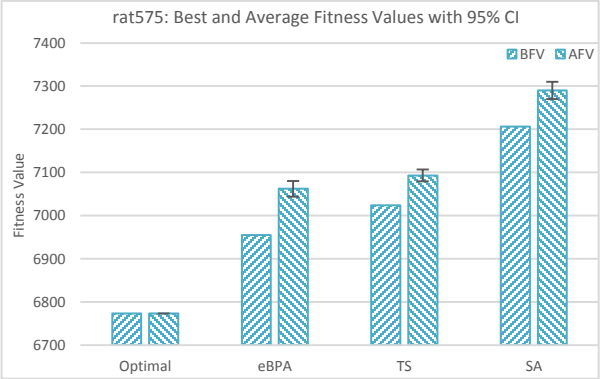


Figure 4.17: The best and average fitness values, along with their 95% CI estimates for rat575

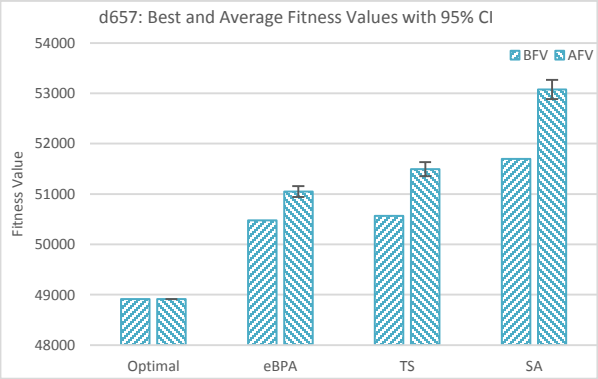


Figure 4.18: The best and average fitness values, along with their 95% CI estimates for d657

The experiments show the ability of the eBPA in determining competitive solutions across multiple problem instances, in using the same parameter settings. Each problem instance differed in complexity and challenged the algorithms in their abilities to balance their transitions from exploration to exploitation, per problem instance. Finding this balance between exploration and exploitation is critical in determining quality solutions. In having determined the best solutions, except for ch130 and the best BFV solution for ch150, it is noted that the eBPA intelligently found promising neighborhood regions more consistently and sifted out higher quality solutions from within those neighborhood regions.

The strength of the eBPA is attributed to its ability to influence the trajectory of the search stochastically, and by way of adaptive memory. The admittance criterion of the eBPA memory structure allows for a $working$ solution to get inserted into the PL memory structure provided that it

meets the minimum requirements (please refer to section 2.3.1). The newly inserted solution will then become the next solution to be used to direct the search. Implicitly, any solution inserted into the PL could possibly be the next $best$ solution. With every insert into the PL, the admissible criterion will constrain further as the quality of the $worst$ solution will improve. This makes admittance into the memory structure more difficult, and this controls the transition from exploration to exploitation. Exploitation is further enhanced by reducing the PL size. This also serves as a purpose for eliminating cycling for PL sizes greater than one.

4.7 Conclusion

The TSP is largely studied in discrete optimization. Its complexity is NP-Hard. This study investigates the abilities of the eBPA, TS and SA in determining solutions to ten sTSP test instances.

The metaheuristic algorithms were compared in their abilities to determine their best and average tour-length solutions, along with their 95% CI solutions per problem instance. The complexities of the problem instances differed in ranging from 130 to 657 vertices. These problem instances provided sufficient challenges to the algorithms for testing purposes.

The results show the competitiveness of the eBPA in determining solutions across multiple problem instances in using the same parameter settings, along with eBPA's competitiveness for discrete optimization problem.

Chapter Five:
The enhanced Best Performance Algorithm on the Just-in-Time Scheduling Problem

5.1 Introduction

Scheduling problems altogether constitute a large and important field of study. It involves the allocation of production (or operational) resources with the intent of optimizing business objectives. Business objectives may include reduced operational costs, reduced production times, increased customer satisfaction, increased profits, etc., in optimizing production processes or service delivery.

Several categorizations of scheduling problems are found in the literature (Brucker, 2007; Adewumi et al., 2009). However, of particular interest in this study is the problem of Just-in-Time (JIT) scheduling (Adamu and Adewumi, 2013a & 2014).

JIT scheduling, as described by Taiichi Ohno (commonly referred to as the father of JIT) is when, "*in a flow process, the right parts needed in assembly reach the assembly line at the time they are needed and only in the amount needed*," (Fateha *et al.*, 2012). Ohno perfected JIT principles at the Toyota manufacturing plants in Japan while being vice-president of manufacturing. At the time, Toyota created high quality vehicles at relatively low costs, compared to its competitors. This was in spite of the disadvantage of having a lack of natural resources in the country. The success of implementing JIT techniques in manufacturing gave Toyota a prominent position within the automobile sector.

Observing Toyota's success, many organizations on a global scale have adopted and implemented JIT techniques with relative successes. Proper implementations of JIT techniques have resulted in documentation emphasizing improved product qualities, improved service deliveries, improved customer satisfaction, improved employer and employee relations, decreased production costs, reduced levels of inventory, and increased profit turnover (Kootanaee, *et al.*, 2013).

Organizations have further benefited in remaining competitive within an industry by offering products and/or services without negotiating on quality at competitive costs. These factors constitute

important business objectives, as organizations remain competitive on the basis of cost, quality and service delivery (Kumar, 2008).

The JIT scheduling problem is largely studied in the sectors of engineering, manufacturing and service delivery (Brucker, 2007). The objective is the optimized delivery of business resources that meet demand, rather than manufacturing or supplying less, or in surplus. JIT scheduling objectives are summarized as follows (Singh and Gard, 2011);

1. Competitiveness – Companies strive to remain competitive in offering products and services at relatively low costs.
2. Efficient processes of production – The more efficient the production processes, the more successful the company.
3. Improved quality of products – Production of smaller quantities allow for better assessment checks. This results in improved product quality.
4. Minimal wastage – This will reduce costs. It will also save time and effort.
5. Reduced inventory – This will minimize investments, as excess inventory will not be held.
6. Efficient space utilization – Fewer inventories means more space available.
7. Improved customer satisfaction – The on-time delivery of quality products and services at competitive rates earn customer satisfaction.
8. Improved supplier relations – Supplier relations get strengthened in having organized delivery of goods and services as required.

The JIT scheduling problems are NP-Hard (Adamu and Adewumi, 2012; Adamu and Adewumi, 2013a&b). This study investigates a JIT scheduling problem and determines solutions using the eBPA, TS and SA. The objective of this study is to test the abilities of the eBPA in determining solutions. The eBPA solutions will be compared against those of the TS and SA algorithms. Once again, this study is the first on the eBPA for a scheduling problem.

The rest of this chapter is structured as follows. Section 5.2 gives descriptions of previous research work. Section 5.3 briefly discusses the investigation to follow. Section 5.4 describes and presents the JIT scheduling problem. Section 5.5 presents and discusses the experimental results obtained. Finally, section 5.6 draws on conclusions.

5.2 Related works

Previous studies on JIT scheduling problems have investigated both the single and multiple machine scenarios. Many optimization techniques have been investigated in determining solutions. They include both the exact and heuristic algorithms.

Ronconi and Kawamura (2010) investigated a single machine JIT scheduling problem with restrictive common due dates. The objective was the minimization of the earliness and tardiness penalties. The study proposed a Branch and Bound algorithm which used lower bounds and pruning rules in exploiting properties of the problem in determining solutions. The algorithm was investigated using 280 jobs. These jobs were characterized by different due dates. The proposed algorithm showed to be effective in outperforming the CPLEX optimization software.

Monette *et al.* (2009) studied a JIT Job-Shop scheduling problem. Jobs were characterized by earliness and tardiness penalties with respect to their due dates. The objective was the minimization of the earliness and tardiness penalties. The study presented a Constrained Programming algorithm. This was a filtering algorithm based on machine relaxation. The study investigated a large range of benchmark test instances. 72 problems were studied in total. The algorithm showed to be very effective in determining 29 of the best-known solutions from the problems studied.

Dereniowski and Kubiak (2010) studied a JIT multi-slot scheduling problem. In this problem, processing time was divided into time slots rather than a single due date for the jobs. The intent of the study was to determine a minimization for the schedule makespan. The study presented algorithms for both the single and parallel machine problem instances.

Suer *et al.* (2012) studied a single machine scheduling problem with non-zero ready times. Jobs were assumed to have arrived at different times, with the arrival times being known in advance. The objective was determining the job sequences in minimizing tardiness. For the problem setting, preemption was not allowed. The study investigated the GA, and compared its solutions to known optimal solutions for small to large size problems. Results showed that GA determined optimal solutions for smaller instances, and near-optimal solutions for larger instances.

Laarhoven *et al.* (1992) investigated the SA algorithm in finding the minimum makespan in large instances of job-shop scheduling problems. The results showed that SA found shorter makespan than tailored deterministic algorithms at the expense of greater execution times. The conclusion was that the disadvantage of expensive computation times was compensated by the simplicity of the algorithm and the higher quality solutions determined.

Sidhoum *et al*. (2004) studied a JIT scheduling problem in a parallel machine environment. Jobs were characterized by distinct due dates, and earliness and tardiness penalties. The research was motivated due to the difficulty of determining lower bounds for JIT scheduling problems in the single and parallel machine environments. A simple heuristic algorithm was presented. Results showed that the differences between the lower and upper bound values for the single and parallel machine environments were around 1% for the problem instances investigated.

McMullen (1998) investigated the use of TS to a mix-model production scheduling problem at an assembly line. The objective of the algorithm was to best determine an assembly schedule based on the part-usage rates and the number of setups involved in the process. The problem objective was to determine an assembly sequence that optimized the assembly process. Results showed that the multiple-objective problem of minimizing part-usage and setup time could be valuable from a managerial perspective.

Naso *et al*. (2007) investigated a hybridized algorithm constructed using GA and a constructive heuristic for a JIT delivery problem in supply chain management. The problem setting is that of a ready-mixed concrete delivery service, in trying to best coordinate the supply of concrete from producers to customer's on-time. Apart from problem complexity, strict time constraints had forbid the early or tardy delivery of ready-mixed concrete. The problems objective was scheduled delivery that maximized profit, in minimizing risk. The case study presented used actual industrial data. The hybridized algorithm was compared to that of four other constructive heuristics. Results showed that the hybridized algorithm determined superior solutions to that of the constructive heuristics.

5.3 Problem Description and Mathematical Formulation

The allocation of company resources to meet business demands are critical to the success of an organization. Therefore, in JIT problem formulation, the untimely scheduling of business resources that miss expected due dates are accompanied by penalty factors called earliness and tardiness penalties.

An earliness penalty is incurred when a job (which implies a service rendered or an item being produced) is scheduled in business before its expected time. As an example, the implication of an earliness penalty relates to the cost of holding inventory before its expected time. Also, a tardiness penalty is incurred when a job is expected to complete after its expected due-date. As an example, this could imply customer dissatisfaction.

The due date of a job refers to either a specific point in time or an interval specified by a window frame of time. The jobs due date is important. It relates to the demand of products or services at predetermined times. The inability of organizations to provide on-time delivery of products and/or services sets the stage for competitiveness in industry.

In a perfect scheduling environment, resources will be made available as required. Realistically however, the limited availability of resources and the differences in demands result in resources becoming available before or after expected due dates. Hence, the problem with JIT scheduling relates to either minimizing the earliness penalty, minimizing the tardiness penalty, or both in scheduling resources (Brucker, 2007). Optimizing a JIT schedule is difficult due to the conflicting objectives.

Most JIT investigations have studied the scheduling of n jobs on a single machine where the due dates are specific points in time. This research studies a JIT problem of scheduling n jobs on m parallel machines where the due dates are window frames of time. The single machine scenario is easier to model and solve, although in industry the possibility of bottlenecking exists. Surprisingly, far fewer papers have surfaced on JIT problems for scheduling jobs on multiple and parallel machines.

The mathematical model presented in this study is that given in Adamu and Abass (2010). This study takes the opportunity of correcting the original mathematical formulation by removing irrelevant

constraints and reformulating the objective function in terms of the schedule. Also, although the formulation is a maximization model, the original study presented solutions for a minimization model. These inconsistencies present the opportunity for this problem to be restudied.

In the mathematical formulation given below, the left and right hand sides of a window interval of time represents the earliest start time a_j (were the job becomes available for processing) and the latest due date d_j (were the job must be completed). The jobs are scheduled starting from time zero. The problems objective is the maximization of the total weight of all on-time jobs. w_j is the weight of a job. This relates to the importance of job x_{ij} being delivered on-time. This problem assumes equivalent earliness and tardiness penalties. These penalty factors are not considered in the objective function. The mathematical formulation is as follows.

Indices:

- i – Indicative of each machine, i.e., $i = 1,..,m$.
- j – Indicative of each job, i.e., $j = 1,..,n$.

Parameters:

- a_j – Represents the left hand side of the due window of job j. This is the earliest start time of job j.
- d_j – Represents the right hand side of the due window of job j. This is the expected completion time of job j.
- p_j – Represents the processing time of each job j.
- t_{ij} – Represents the actual start time of job j on machine i.
- $C_{ij}(S)$ – Given a schedule S, $C_{ij}(S)$ represents the completion time of job j on machine i, i.e., $C_{ij}(S) = t_{ij} + p_j$. Hence, job j is said to be early if $C_{ij}(S) < a_j$, tardy if $C_{ij}(S) > d_j$ else on-time if $a_j \leq C_{ij}(S) \leq d_j$.
- w_j – Weight of job j.

Variables:

- $x_{ij}(S)$ – Representative if job j is allocated on machine i, in schedule S.

Objective Function:

$$\text{Maximize } f = \sum_{i=1}^{m}\sum_{j=1}^{n} w_j x_{ij}(S) \tag{5.1}$$

Subject to constraints;

$$a_j \leq \left\{max_{k=1}^{j}\left\{C_{i(k-1)}(S), a_j - p_j\right\} + p_j\right\}x_{ij} \leq d_j, \qquad \forall i = 1,..,m;\ \forall j = 1,..,n \tag{5.2}$$

$$\sum\nolimits_{i=1}^{m} x_{ij} \leq 1, \qquad \forall j = 1,..,n \tag{5.3}$$

$$x_{ij} = \begin{cases} 1, & iff\ a_j \leq C_{ij}(S) \leq d_j \\ 0, & otherwise \end{cases}, \qquad \forall i = 1,..,m;\ \forall j = 1,..,n \tag{5.4}$$

Equation 5.1 represents the total weight of all on-time jobs. Equation 5.2 ensures that if job j is scheduled on machine i, it will start and complete processing between its earliest start time a_j and latest finishing time d_j. Equation 5.3 ensures that job j will be assigned to at most one machine i. Equation 5.4 represents a job being either on-time, early or tardy, with 1 representing on-time and 0 otherwise.

The problem assumptions are as follows;

1. Setup time is included in processing time. Hence, preemption is not allowed. When job $j-1$ is completed, there is no delay in starting job j on machine i.
2. There is no delay in machine processing. When job j starts, it is expected to be completed as represented by processing time p_j.
3. Only one job can be processed at any given time on machine i.

5.4 Results and Discussion

The JIT scheduling problem is a discrete optimization problem. For this reason, only the eBPA, TS and SA metaheuristics are investigated (again, the eBPA had been developed due to the weakness of the BPA for discrete optimization problems, as was demonstrated in chapter two).

In chapter's three and four, termination of the algorithms occurred at a point of convergence. Convergence had been detected when a specific number of idle iterations had been reached. However, in this investigation, we implement termination after a fixed number of iterations. With this approach, the parameter settings of the algorithms need to be set appropriately to make the most effective use of the limited computational time available. For this reason, the parameter settings of the algorithms have been set to exercise greater levels of exploitation, as was determined after a number of experimental tests.

Simulations were run using sets of jobs $n \in \{500, 1500, 2500\}$, tested on sets of machines $m \in \{2, 5, 10, 15, 20\}$. For each job $j = 1, .., n$, its processing time p_j was randomly determined to fall within the interval (1,99). To set the starting and completion times a_j and d_j for job j two "Traffic Congestion Ratio" variables k_1 and k_2 was randomly selected from set $V \in \{1, 5, 10, 20\}$. Using k_1, a_j was randomly generated to fall within the interval $(0, n\ /\ mk_1)$. Using k_2, d_j was randomly generated to fall within the interval $(a_j + p_j, a_j + p_j + n\ /\ mk_2)$.

To test the algorithms fairly, a set of n jobs was initially generated and passed in as the input parameter to each of the algorithms. This was then used to test the algorithms on a particular machine m. Therefore, each algorithm used the same job set in testing on a particular machine. This ensured the results were determined fairly for comparative purposes. To determine average performance results, each algorithm was run 30 times for each pair of job-machine combination. 30 runs were sufficient considering the expensive computational times of the metaheuristic algorithms. From the 30 runs, per job-machine combination, the best solution (BFV) of each algorithm will be compared. The BFV is the highest total weight of all on-time jobs from the 30 runs, per job-machine combination per algorithm. Comparisons of the average solution performances will also be documented. This is for their average fitness value (AFV) solutions and their average execution time (AVG) performances.

To further test the algorithms fairly, their parameter settings were set such that each metaheuristic algorithm executed for exactly 10^6 objective function evaluations, per run. The parameter settings were set as follows;

- eBPA – The PL size was set at 5. The $noOfIterations$ was set at 10^6. p_a was set at 0.005.
- TS – The TL size was set at 7. The CL size was set at 10^2. The $noOfIterations$ was set at 10^4.

- SA – The $stepsPerChange$ was set at 10^3. T was set at 115. F was set at 0.005. α was set at 0.99.

The findings of the simulations are documented below.

Table 5.1: Statistics of the Best Fitness Values (BFV) and Average Fitness Values (AFV) for the class of 500 jobs

No. of Jobs	**Methods**	**Fitness Values**	**No. of Machines**				
			2	**5**	**10**	**15**	**20**
500	eBPA	BFV/ AFV	**391.34**/ **379.34**	**441.74**/ 427.02	**546.84**/ **526.58**	**601.23**/ 581.25	**683.79**/ 665.76
	TS	BFV/ AFV	325.91/ 317.44	382.02/ 371.01	462.70/ 453.36	519.78/ 511.09	584.56/ 575.07
	SA	BFV/ AFV	388.70/ 370.75	441.10/ **431.32**	532.90/ 519.58	593.44/ **583.35**	680.78/ **667.15**

Table 5.1 gives the statistical values of the BFV and AFV fitness values of each algorithm, per machine set, for the class of 500 jobs. The best BFV and AFV solutions, per machine set, is highlighted in bold font for clarity purposes.

From Table 5.1 it is seen that the eBPA determined the overall BFV solutions for all machine sets. On average, the eBPA determined the overall AFV solutions for machine sets 2 and 10. SA determined the overall AFV solutions for machine sets 5, 15 and 20. However, it is seen that these solutions are only marginally superior to the eBPA solutions. TS has shown to be the weakest of the algorithms.

Graphical comparisons of the algorithms best and average fitness value solutions, as determined from Table 5.1, are seen in Figures 5.1 and 5.2 below.

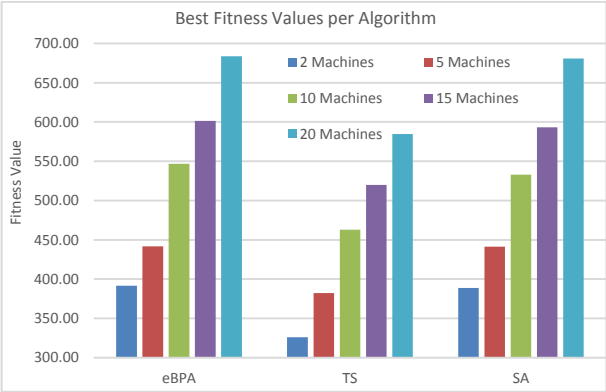


Figure 5.1: BFV comparisons for the class of 500 jobs

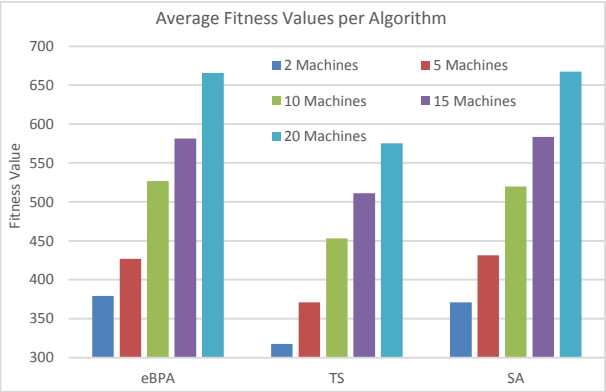


Figure 5.2: AFV comparisons for the class of 500 jobs

Table 5.2: The average execution times in milliseconds per machine set, for the class of 500 jobs

No. of Jobs	**Methods**	**Average Execution Time (ms) for each Machine Set**				
		2	**5**	**10**	**15**	**20**
500	eBPA	**8,394**	**15,637**	**26,763**	38,338	**47,594**
	TS	8,568	15,667	27,622	**37,804**	47,773
	SA	8,829	15,710	27,502	38,756	49,541

Table 5.2 gives the statistical values of the average execution times in milliseconds (ms) for the algorithms, per machine set, for the class of 500 jobs. Although it is observed that the average execution times of the algorithms are fairly similar, the eBPA executed the fastest for machine sets 2, 5, 10 and 20. TS executed the fastest for machine set 15.

The relatively fast execution times of the eBPA relate to its small *PL* size, which strategically decreased as the algorithm iterated. This caused the admittance criterion to become increasingly restrictive in allowing for greater exploitation by accepting fewer solutions to update the memory structure. This allowed the eBPA to identify stronger solutions and explains its relatively fast execution times. A graphical comparison of the statistics given in Table 5.2 is seen in Figure 5.3.

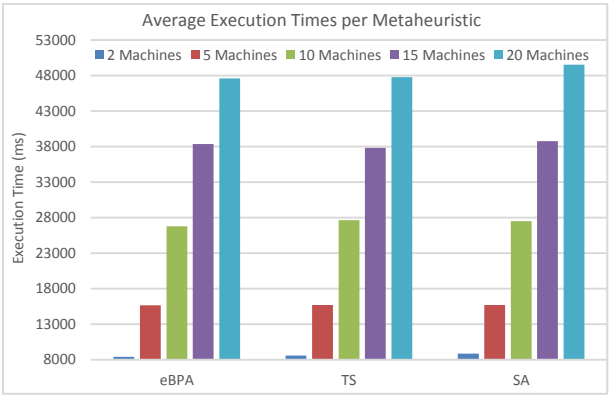


Figure 5.3: Average execution times per metaheuristic per machine set, for the class of 500 jobs

For the class of 500 jobs, it is concluded that the eBPA was the strongest algorithm.

Table 5.3 gives the statistical values for the overall BFV and AFV solutions per machine set, for the class of 1,500 jobs. From Table 5.3 it is observed that the eBPA determined the overall BFV solutions for all machine sets, except machine set 10. It also determined the overall AFV solutions for all machine sets. SA determined the overall BFV solutions for machine set 10. SA again determined superior solutions over TS.

Table 5.3: Statistics of the Best Fitness Values (BFV) and Average Fitness Values (AFV) for the class of 1,500 jobs

No. of Jobs	**Methods**	**Fitness Values**	**No. of Machines**				
			2	**5**	**10**	**15**	**20**
1,500	eBPA	BFV/ AFV	**696.21**/ **653.63**	**841.96**/ **778.05**	896.90/ **877.80**	**952.07**/ **905.63**	**1,066.63**/ **1,022.07**
	TS	BFV/ AFV	549.57/ 531.71	679.26/ 664.42	777.44/ 756.31	793.86/ 782.26	908.43/ 884.27
	SA	BFV/ AFV	654.36/ 632.47	811.42/ 776.90	**909.04**/ 867.83	941.62/ 898.14	1,041.08/ 999.64

Graphical comparisons of the algorithms best and average fitness value solutions, as determined from Table 5.3, are seen in Figures 5.4 and 5.5 below.

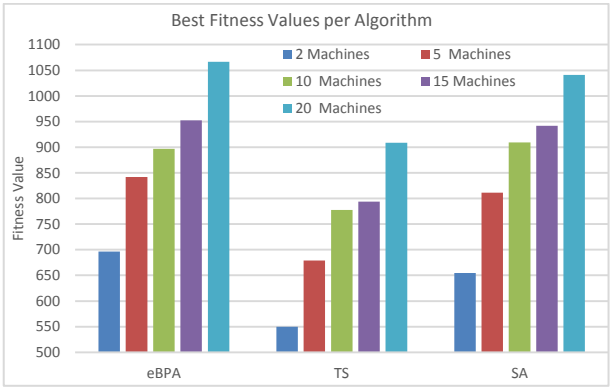


Figure 5.4: BFV comparisons for the class of 1,500 jobs

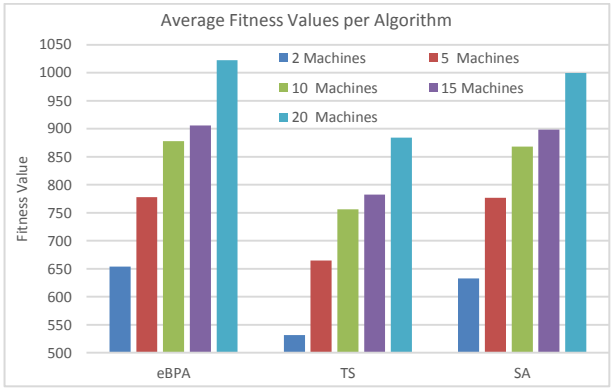


Figure 5.5: AFV comparisons for the class of 1,500 jobs

Table 5.4 below gives the statistics of the average execution times for the metaheuristic algorithms, per machine set, for the class of 1,500 jobs. It is observed that the average execution times were much more competitive for this class of jobs. The eBPA performed faster on average for machine sets 2, 10 and 20. TS performed the fastest for machine set 5, and SA performed the fastest for machine set 15. Graphical comparisons of the execution time performances are seen in Figure 5.6.

Table 5.4: The average execution times in milliseconds per machine set, for the class of 1,500 jobs

No. of Jobs	**Methods**	**Average Execution Time (ms) for each Machine Set**				
		2	**5**	**10**	**15**	**20**
1,500	eBPA	**27,180**	49,508	**87,321**	117,160	**149,333**
	TS	27,964	**49,216**	88,229	117,477	150,678
	SA	28,184	49,281	88,037	**116,585**	150,116

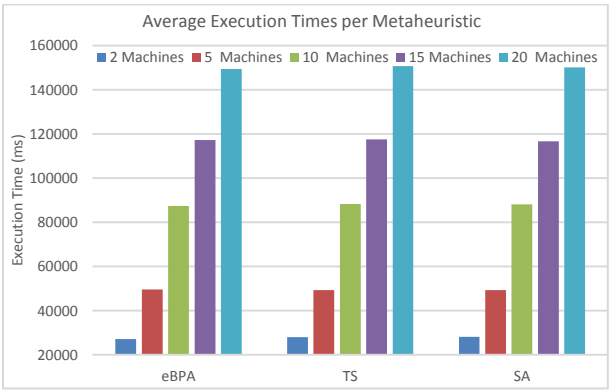


Figure 5.6: Average execution times per metaheuristic per machine set, for the class of 1,500 jobs

For the class of 1,500 jobs it is also concluded that the eBPA was the strongest algorithm.

Table 5.5: Statistics of the Best Fitness Values (BFV) and Average Fitness Values (AFV) for the class of 2,500 jobs

No. of Jobs	Methods	Fitness Values	No. of Machines				
			2	5	10	15	20
2,500	eBPA	BFV/ AFV	993.52/ 960.94	1,100.46/ **1,066.45**	**1,109.63**/ **1,054.93**	**1,249.22**/ **1,197.17**	1,307.02/ 1,267.87
	TS	BFV/ AFV	799.60/ 789.21	948.02/ 919.61	951.80/ 930.43	1,071.70/ 1,057.45	1,150.72/ 1,130.19
	SA	BFV/ AFV	**1,003.11**/ **962.44**	**1,105.59**/ 1,039.86	1,085.21/ 1,039.86	1,234.31/ 1,182.19	**1,309.06**/ **1,267.89**

Table 5.5 gives the statistical values of the BFV and AFV solutions for each algorithm per machine set for the class of 2,500 jobs. From Table 5.5 it is seen that the eBPA determine better BFV and AFV solutions for machine sets 10 and 15, while SA determined better BFV and AFV solutions for machine sets 2 and 20. For machine set 5, the eBPA determined a better AFV solution and SA determined a better BFV solution.

Graphical comparisons of the algorithms best and average fitness value solutions, as determined from Table 5.5, are seen in Figures 5.7 and 5.8 below.

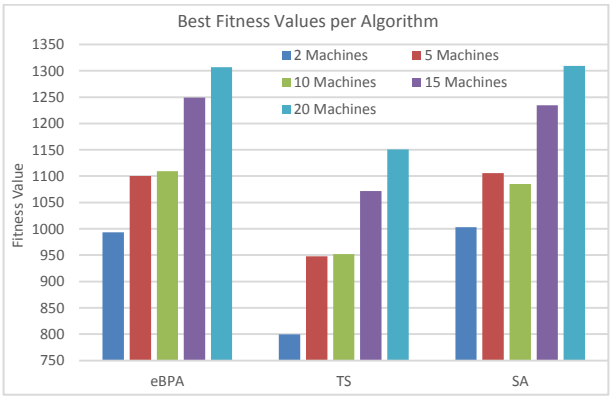


Figure 5.7: BFV comparisons for the class of 2,500 jobs

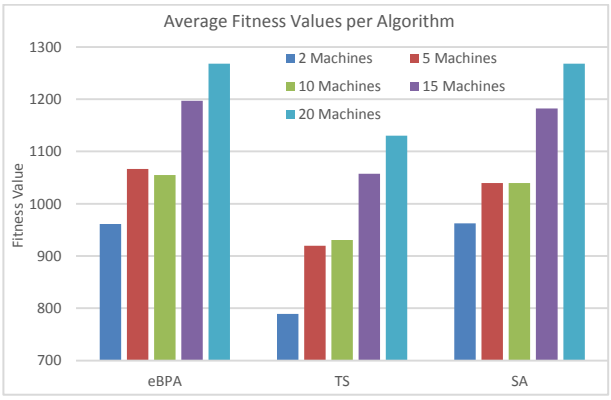


Figure 5.8: AFV comparisons for the class of 2,500 jobs

Table 5.6 below gives the statistics of the average execution times for the metaheuristic algorithms, per machine, set for the class of 2,500 jobs. It is observed that for this class, TS executed the fastest for machine set 2, SA executed the fastest for machine set 5, and the eBPA executed the fastest for machine sets 10, 15 and 20. Graphical comparisons of the execution time performances are seen in Figure 5.9.

Table 5.6: The average execution times in milliseconds per machine set, for the class of 2,500 jobs

No. of Jobs	Methods	Average Execution Time (ms) for each Machine Set				
		2	5	10	15	20
2,500	eBPA	44,534.00	80,756.00	**139,053.00**	**195,479.00**	**260,926.00**
	TS	**44,461.00**	81,128.00	141,250.00	209,198.00	285,553.00
	SA	45,055.00	**80,646.00**	139,213.00	196,093.00	272,481.00

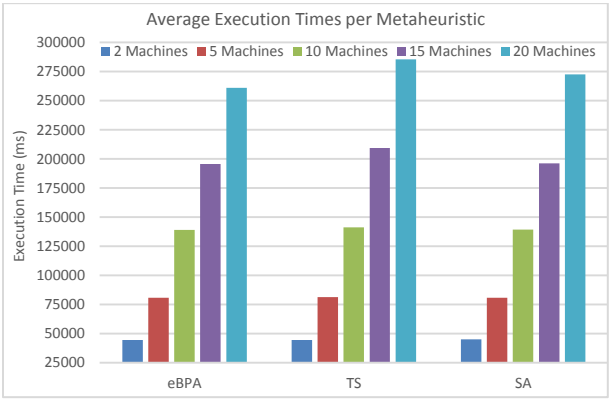


Figure 5.9: Average execution times per metaheuristic per machine set, for the class of 2,500 jobs

For the class of 2,500 jobs, both the eBPA and SA performed similarly in determining an equivalent number of best solutions. However, the eBPA executed the fastest for most machine sets.

Although standard implementations of the algorithms were compared, the results documented are significant in that the techniques employed by the eBPA have shown to be very competitive compared to that of TS and SA for this discrete optimization problem.

The strength of the eBPA lay in its memory structure and the techniques used in allowing the solutions contained within to direct the search. Solutions registered in the PL would have identified attractive points within the neighborhood regions of the solution space. However, it uses the information of the worst solution in the list as a strategic point to move the search forward. The memory structure adapts dynamically in accepting solutions that satisfy the admittance criterion. It uses each solution inserted into the PL as the next $working$ solution. This strategy allows the eBPA to use a population of solutions to direct the search rather than using the population as a network to exploit a neighborhood region.

As the search iterates, and the worst solution in the PL is improved upon, the admittance criterion becomes more restrictive in allowing for greater levels of exploitation. Exploitation is further increased with the PL dynamically reducing in size by cutting away the worst solutions in a strategic manner. This constrains the admittance criterion further, and allows the eBPA to exploit quality solutions as the PL narrows in size. The solutions accepted into the PL does not need to be the best overall. However, along the way the best solution will be found. An added advantage of the eBPA is its simplistic design and the few parameter settings required.

5.5 Conclusion

The problem of JIT scheduling is an important study. The objective is to determine operational processes that would allocate limited business resources efficiently in optimizing business objectives. Business objectives may include the optimization of operational costs, operational times, inventory storage, customer and supplier relations, profits margins, etc.

In this study, the JIT problem of allocating a large number of jobs required to be processed on m parallel machines was investigated. A job represents a business resource required to be made available during a specific window interval of time. An example may be the delivery of vehicles to customers that require rented vehicles within a specific time frame. The objective was therefore to determine a schedule that would maximize the total weighed number of all on-time jobs that could be scheduled. The JIT problem is NP-Hard.

To determine solutions, the eBPA, TS and SA algorithms were investigated. The algorithms were compared in their abilities to determine their best and average fitness value solutions. The fitness value referred to the weight of all on-time jobs scheduled per job-machine pair. The algorithms were also compared in terms of their average execution times performances. The results showed that the eBPA performed very competitively in being compared to both TS and SA.

Chapter Six:
Conclusions and Future Research

6.1. Summary of Research Work

Faced with the challenges of limited availabilities of natural resources such as land, irrigated water supplies and financial investments, in crop production, the ACP problem had been developed. The ACP problem seeks to determine optimized solutions for these limited resources.

The first ACP mathematical model was introduced in Chetty and Adewumi (2013b). The mathematical model related to that of an existing irrigation scheme. In this study, SI techniques were investigated. These included the CS, FA, GSO and the GA algorithms. This study was significant in that it was the first on the CS, FA and GSO for a crop planning problem. Thereafter, the ACP problem was further evolved in considering new irrigation schemes (Chetty and Adewumi, 2013c). In that study, the same SI metaheuristic techniques were investigated.

Thereafter, the research efforts took a turn in having investigated LS metaheuristic algorithms. The algorithms investigated were that of TS and SA. It was at this point that it was realized that there were apparent weaknesses in the strategic designs of both of these metaheuristics.

With TS, it was realized that although it employed the benefits of memory strategies, it lacked slightly in its stochastic ability. On the other hand, although SA is pure stochastic, its disadvantage is that it does not employ memory strategies. Hence, SA loses valuable solutions found during its search. These realizations motivated for the development of a new metaheuristic, which was the BPA (Chetty and Adewumi, 2013a). Hence, the BPA was an attempt to bridge the strengths of both the memory ability of TS and the stochastic ability of SA. In Chetty and Adewumi (2013a), a large benchmark collection of unconstrained continuous optimization functions were investigated for comparative study.

To further test the abilities of the BPA, both ACP problems at existing and new irrigation schemes were investigated in Chetty and Adewumi (2014) and in Chetty and Adewumi (2013d) respectively. Yet again, there problems were instances of continuous optimization. To test the completeness of the algorithm, discrete optimization problems needed to be investigated.

The discrete optimization problem chosen to be investigated was that of the sTSP; herein lay the stumbling block. In BPA's applications to instances of sTSP's, it was realized that the BPA had performed very poorly relative to both TS and SA. Further analysis revealed that the reason for the poor performances were due to stronger levels of explorative abilities, and weaknesses in exploitation. The imbalance of these contrasting objectives had proven to be very costly for this discrete optimization problem (and for discrete optimization in general).

Therefore, to try and correct this performance aspect, the eBPA had been developed. As mentioned in chapter two, although these algorithms are modeled on similar analogical principles, their algorithmic designs are different. The eBPA is now a truer representation of the merge between the strengths of both SA and TS, in that, it is a single-point metaheuristic algorithm, which is similar to that of SA, and it directs the trajectory of the search by intelligently employing memory strategies, similar to that of TS. The strategies implemented by the eBPA has proven to be very effective in determining competitive solutions to both continuous and discrete optimization problems. Additional advantages of the eBPA include its simplistic design, its flexibility in being non-problem dependant, its consistency in balancing its transition from exploration to exploitation, and its effectiveness in determining high quality solutions. Overall, the eBPA is very robust.

6.2 Conclusions

This thesis has presented the eBPA, which is a new Monte Carlo LS metaheuristic algorithm. The eBPA trajects through a solution space stochastically, yet uses intelligence by way of its implementation of adaptive memory.

The strategies implemented by the eBPA are intended to penetrate complexed regions of the solution space. It determines high quality solutions to difficult optimization problems, within polynomial time complexity, and at low computational costs. The strength of the eBPA is reflected in its ability to balance its transition from exploration to exploitation.

Exploration is a global search strategy. It attempts to locate promising neighborhood regions within the domains of the solution space. Exploration is primarily influential during the initial phases of the search. On the other hand, exploitation is a local search strategy. It attempts to identify the local optimum point from within a local neighborhood region. Exploitation is primarily influential during the latter stages of the search. A fine balance between these two contrasting objectives is critical to the success of any metaheuristic algorithm.

The eBPA stems from its predecessor–the BPA. Further research was undergone to try and improve upon the efficiency aspects of the BPA (specifically for discrete optimization). In this thesis, a comprehensive analysis has been performed in discussing the conceptual differences in the technical and strategic designs of both the BPA and the eBPA. The results, in having performed the investigations, have shown the superiority of the eBPA over the BPA for discrete optimization problems.

Yet, in their applications to the ACP problem, which is a continuous optimization problem, both algorithms had performed very competitively. To further test the efficiency aspects of the eBPA, this algorithm had been implemented in determining solutions to ten benchmark instances of the sTSP's, and to an instance of the JIT scheduling problem. Both these problems types were discrete optimization problems. The ACP problem, sTSP, and the JIT scheduling problem are all NP-Hard optimization problems.

For the ACP problem, sTSP, and the JIT scheduling problem, the comparisons were made against that of TS and SA. TS and SA were the algorithms investigated due to both the eBPA and the BPA embedding characteristics of both of these metaheuristics. However, the eBPA is a truer blend of the embedded strengths, which has been proven by way of its excellent balance in its transition from exploration to exploitation, which is critical to the success of any metaheuristic algorithm. Formally, the eBPA differs from memory-less search algorithms, which are modelled primarily on randomization, and memory-based search algorithms, which are modelled primarily on determinism.

In this thesis a new mathematical formulation for the ACP problem has also been presented. The mathematical formulation considers for the market economic factors of the economy of scale, and the demand and supply relations.

Market economics have always had a noticeable presence in crop production. With the economy of scale influence, crop production on a larger scale has always been more profitable, as unit costs are lower. However, since the sale of the finished products are sold within deregulated marketing environments, the demand and supply relational factors also needed to be incorporated. The inclusion of these market economic factors make for a more interesting problem, yet is necessary for realistic solutions.

6.3. Future Research

The eBPA has been developed as an AI algorithm, in having modeled a competitive element of an individual. Although metaheuristic algorithms in AI have been designed in primarily modelling biological agents (or occurrences) in nature, scarce research have surfaced on modelling human cognitive behaviors and thinking within the AI framework. This thesis has opened doors in realizing the potentials of modeling human characteristics in metaheuristic design, within the AI framework.

The possibilities are numerous in investigating human behaviors and thinking at both personal and group levels, especially in trying to capture the competitive nature of individuals in their attempts to achieve maximum successes. Intelligence is also related to the way human's reason in decision making; this is another important reason for modelling human beings in the design of metaheuristic algorithms.

The eBPA should also be applied in investigating other types of optimization problems. Hybridizations of this algorithm is also possible. Another alternative is to research the possibility of temporarily increasing the memory structure size of the eBPA, in attempting to escape from local entrapments.

REFERENCES

Acquaah, G. (2004). "Principles of crop production: theory, techniques, and technology," 2nd Edition, Prentice Hall.

Adamu, M. O. and Abass, O. (2010). "Parallel Machine Scheduling To Maximize The Weighted Number Of Just-In Time Jobs," Journal of Applied Science and Technology (JAST), Vol. 15 (1&2), pp. 27-34.

Adamu, M. O. and Adewumi, O. A. (2012). "Metaheuristics for scheduling on parallel machine to minimize weighted number of early and tardy jobs," International Journal of Physical Sciences, Vol. 7(10), pp. 1641-1652.

Adamu, M. O. and Adewumi, A. O. (2013a). "A comparative study of meta-heuristics for identical parallel machines", Journal of Engineering and Technology Research, Vol. 5(7), pp. 207-216.

Adamu, M. O. and Adewumi, A. O. (2013b). "Unweighted Parallel Machine Scheduling: A Meta-Heuristic Approach", Proceedings of International Conference in Electrical and Electronics Engineering, Istanbul, Turkey, 65-72.

Adamu, M. O. and Adewumi, A. O. (2014). "A survey of single machine scheduling to minimize weighted number of tardy jobs", Journal of Industrial and Management Optimization, 10(1), 219-241.

Adewumi, A. O. (2010). "Some improved genetic-algorithms based heuristics for global optimization with innovative applications", Ph.D. thesis, School of Computational and Applied Mathematics, University of Witwatersrand, Johannesburg, South Africa.

Adewumi, A. O. and Ali, M. M. (2010). "A multi-level genetic algorithm for a multi-stage space allocation problem". Mathematical and Computer Modelling, Vol. 51(1–2), pp. 109-126.

Adewumi, A. O., Sawyerr, B. A. and Ali, M. M. (2009) "A heuristic solution to the university timetabling problem", Engineering Computations, Vol. 26(8), pp.972 – 984

Adeyemo, J. and Otieno, F. (2010a). "Maximum irrigation benefit using multi-objective differential evolution algorithm (MDEA)," Int. J. of Sustainable Development, Vol. 1(2), pp. 39-44.

Adeyemo, J., Bux, F. and Otieno, F. (2010b). "Differential evolution algorithm for crop planning: Single and multi-objective optimization model," Int. J. of the Physical Sciences, Vol. 5(10), pp. 1592-1599.

Akinyelu, A. A. and Adewumi, A. O. (2014). "Classification of Phishing Email Using Random Forest Machine Learning Technique," Journal of Applied Mathematics, vol. 2014, Article ID 425731, 6 pages.

Arasomwan, M. A. and Adewumi, A. O (2013). "On adaptive chaotic inertia weights in particle swarm optimization", Proceedings of the 4th IEEE Symposium Series on Computational Intelligence (SSCI '13), Singapore, pp. 72–79.

Arasomwan, M. A. and Adewumi, A. O. (2014). "Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems," The Scientific World Journal, vol. 2014, Article ID 798129, 23 pages.

Arasomwan, M. A. and Adewumi, A. O. (2014). "An Investigation into the Performance of Particle Swarm Optimization with Various Chaotic Maps", Mathematical Problems in Engineering, Vol. 2014, Article ID 178959, 17 pages.

Astera, M. (2010). "The Ideal Soil: A Handbook for the New Agriculture," 1st edition, ISBN # 978-0-9844876-3-9.

Aspremont, A. and Boyd, S. (2003). "Relaxations and Randomized Methods for Nonconvex QCQPs," EE392o Class Notes, Stanford University.

Bai, R. (2005). "An investigation of novel approaches for optimising retail shelf space allocation," Ph.D. thesis, University of Nottingham.

Bailey, A. C., McLain, W. T. and Beard, W. R. (2001). "Fuel-Saving Strategies for Dual Spacecraft Interferometry Missions," Journal of the Astronatical Sciences, Vol. 49(3), pp. 469-488.

Biggs, N. L., Lloyd, E., Keith, W. and Robin, W. (1986). "Graph Theory 1736-1936", Clarendon Press, Oxford,ISBN 978-0-19-853916-2.

Blum, C. and Merkle, D. (2008). "Swarm Intelligence: Introduction and Applications (Natural Computing Series)," Springer-Verlag. ISBN: 978-3-540-74088-9.

Blum, C and Roli, A. (2003). "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," ACM Computing Surveys, Vol. 35(3), pp. 268-308.

Boyd, S and Vandenberghe, L. (2004). "Convex Optimization," Cambridge University Press, UK.

Brouwer, C. and Heibloem, M. (1986). "Irrigation Water Management: Irrigation Water Needs," FAO Land and Water Development Division, [Online] Available: http://www.fao.org/docrep/s2022e/s2022e00.htm.

Brucker, P. (2007), "Scheduling Algorithms," 5th ed., Springer, ISBN 978-3-540-69515-8.

Charles, A. F. (1986). "Multiple cropping systems," Technology and Engineering, Macmillan Pub. Co.

Chetty, S. and Adewumi, A. O. (2013a), "Three New Stochastic Local Search Algorithms for Continuous Optimization Problems," Computational Optimization and Applications, Vol. 56(3), pp. 675-721.

Chetty, S. and Adewumi, A. O. (2013b). "Comparison Study of Swarm Intelligence Techniques for the Annual Crop Planning Problem," IEEE Transactions on Evolutionary Computing, Vol. 18(2), pp. 258-268.

Chetty, S. and Adewumi, A. O. (2013c). "Studies in Swarm Intelligence Techniques for AnnualCrop Planning Problem in a New Irrigation Scheme," South African Journal

Of Industrial Engineering, Vol. 24(3), pp. 205–226.

Chetty, S. and Adewumi, A. O. (2013d). "Three New Stochastic Local Search Metaheuristics for the Annual Crop Planning Problem Based on a New Irrigation Scheme," Journal of Applied Mathematics, Vol. 2013.

Chetty, S. (2013e). "Studies in Heuristics for the Annual Crop Planning Problem," MSc Thesis, University of KwaZulu-Natal.

Chetty, S. and Adewumi, A. O. (2014). "On the performance of new local search heuristics for annual crop planning: case study of the Vaalharts irrigation scheme," Journal Of Experimental and Theoretical Artificial Intelligence, Vol. 27(2), pp. 159–179.

Chetty, S. and Adewumi, A. O. (2015). "A Study on the Enhanced Best Performance Algorithm for the Just-In-Time Scheduling Problem," Discrete Dynamics in Nature and Society, Vol. 2015.

Davendra, D. (2010). "Traveling Salesman Problem, Theory and Applications," InTech, ISBN 978-953-307-426-9.

Dereniowski, D. and Kubiak, W. (2010), "Makespan Minimization of Multi-Slot Just-In Time Scheduling on Single and Parallel Machines," Journal of Scheduling, Vol. 13(5), pp. 479-492.

Domshlak, C. Prestwich, S. Rossi, F. Venable, K. B. and Walsh, T. (2006). "Hard and soft constraints for reasoning about qualitative conditional preferences," J. of Heuristics, Vol. 12(4-5), pp. 263-285.

Dorigo, M. (1992). "Optimization, Learning and Natural Algorithms," Ph.D. thesis, Politecnico di Milano, Italie.

Dorigo, M. and Gambardella, L. M. (1997). "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Transactions on Evolutionary Computation, Vol. 1(1), pp. 53–66.

Dukes, M. D. Zotarelli, L. Liu, G. D. and Simonne, E. H. (2012). "Chapter **3.** Principles and Practices of Irrigation Management for Vegetables," Univ. of Florida.

Eiben, A. E. and Smith, J. E. (2003). "Introduction to evolutionary computing," 1st Edition, Springer, Natural Computing Series.

Faris, J. E. (1961). "Economies of Scale in Crop Production," American Journal of Agricultural Economics, Vol. 43(5), pp. 1219-1226.

Fateha, A. A. N., Nafrizuan, M. Y. and Razlan, A. (2012). "Review on Elements of JIT Implementation," In International Conference on Automotive, Mechanical and Materials Engineering (ICAMME'2012) Penang, Malaysia.

Hancock, H. (2005). "Theory of Maxima and Minima," Scholarly Publishing Office, University of Michigan Library.

Holland, J. H. (1975). "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, MI.

Georgiou, P. E., Papamichail, D. M. (2008). "Optimization model of an irrigation reservoir for water allocation and crop planning under various weather conditions," Irrigation Science, Vol. 26(6), pp. 487–504.

Glover, F. (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence," Computers and Operations Research, Vol.13, pp. 533–549.

Glover, F. (1989). "Tabu Search - Part 1," ORSA Journal on Computing, Vol. 1(2), pp. 190 206.

Glover, F. (1990). "Tabu Search - Part 2," ORSA Journal on Computing, Vol. 2(1), pp. 4 32.

Glover, F. (1993). "A User's Guide to Tabu Search*," Annals of Operations Research, Vol 41, pp. 3-28.

Glover, F. (1995), "Tabu Search Fundamentals and Uses," Graduate School of Business, University of Colorado, Condensed version published in; Mathematical Programming: State of the Art, pp. 64-92.

Grove, B. (2008). "Stochastic efficiency optimisation analysis of alternative agricultural water use strategies in Vaalharts over the long- and short-run," Ph.D. thesis, Univ. of the Free State, Bloemfontein, South Africa.

Grötschel, M., Jünger, M. and Reinelt, G. (1991). "Optimal Control of Plotting and Drilling Machines: A Case Study," Mathematical Methods of Operations Research, Vol. 35(1), pp. 61-84.

Hjertenes, M. O. (2002). "A Multilevel Scheme for the Travelling Salesman Problem," University of Bergen.

Johnson, S. D. and McGeoch, A. L. (1997). "The Traveling Salesman Problem: A Case Study in Local Optimization," Local Search in Combinatorial Optimization, Vol. 1, pp. 215 310.

Kantanantha, N. (2007). "Crop decision planning under yield and price uncertainties," Ph.D. thesis, Georgia Institute of Technology.

Karp, R. M. (1972). "Reducibility among Combinatorial Problems," Complexity of Computer Computations: Proc. of a Symp. on the Complexity of Computer Computations, The IBM Research Symposia Series, New York: Plenum Press, pp. 85-103.

Kennedy, J. and Eberhart, R. (1995). "Particle Swarm Optimization," Proceedings of IEEE International Conference On Neural Networks, Vol. 4. pp. 1942–1948.

Kirkpatrick, S. Gelatt, C. D. and Vecchi, M. P. (1983). "Optimization by Simulated Annealing," Science, Vol. 220, pp.671–680.

Kootanaee, A. J., Babu, K. N. and Talari, H. F. (2013). "Just-in-Time Manufacturing System: From Introduction to Implement," International Journal of Economics, Business and Finance, Vol. 1(2), pp. 7–25.

Kougias, I. and Theodosiou, N. (2010). "A New Music-Inspired Harmony Based Optimization Algorithm. Theory and Applications," International Conference on Protection and Restoration of the Environment, X Corfu, Greece.

Krauth, W. (1998). "Introduction to Monte Carlo Algorithms," Advances in Computer Simulation, Lecture Notes in Physics, Springer Verlag.

Krishnand, K. N. and Ghose, D. (2009a). "Glowworm Swarm Optimisation for Simultaneous Capture of Multiple Local Optima of Multimodal Functions," Swarm Intelligence, Vol. 3, pp. 87-124.

Krishnand, K. N. and Ghose, D. (2009b). "Glowworm Swarm Optimisation: A New Method for Optimizing Multimodal Functions," International J. of Computational Intelligence Studies, Vol. 1(1), pp. 93-119.

Krugman, P. (1980), "Scale Economies, Product Differentiation, and the Pattern of Trade", American Economic Review, Vol. 70, pp. 950-959.

Kumar, S., Ranga, V., Chopra, R. and Sehrawat, M. S. (2008). "Scope of JIT Management in Indian Service Industries," International Conference on Intelligent Systems and Networks (ISN-2008).

Kumbharana, S. N. and Pandey, G. M. (2013). "Solving Travelling Salesman Problem using Firefly Algorithm," International Journal for Research in Science & Advanced Technologies, Vol. 2(2), pp. 53-57.

Laarhoven, P. J. M., Aarts, E. H. L. and Lenstra, J. K. (1992). "Job Shop Scheduling by Simulated Annealing," Operations Research, Vol. 40(1), pp. 113-125.

Liberti, L. (2008). "Introduction to Global Optimization," Technical Report, LIX, ´Ecole Polytechnique, Palaiseau F-91128, France.

Lin, S. (1965). “Computer Solutions of the Traveling Salesman Problem,” Bell System Technical Journal, Vol. 44, pp. 2245-2269.

Lin, S. and Kernighan, B. W. (1973). “An effective heuristic algorithm for the travelling salesman problem,” Operations Research, Vol. 21(2), pp. 498-516.

Louis, S. J. and Tang, R. (1999). “Interactive Genetic Algorithms for the Travelling Salesman Problem,” Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, Vol. 1, pp. 1043-1048.

Maisela, R. J. (2007). “Realizing agricultural potential in land reform: the case of Vaalharts Irrigation Scheme in the Northern Cape Province,” M. Phil mini-thesis, Univ. of the Western Cape, Cape Town, South Africa.

Malek, M., Guruswamy, M. and Pandya, M. (1989). “Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem,” Annals of Operations Research, Vol. 21, pp. 59-84.

Matai, R., Singh, S. and Mittal, L. M. (2010). “Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches,” Traveling Salesman Problem, Theory and Applications, InTech, ISBN 978-953-307-426-9.

McMullen, P. R. (1998), “JIT Sequencing for Mixed-Model Assembly Lines with Setups using Tabu Search,” Production Planning & Control (1998), Vol. 9(5), pp. 504-510.

Michalewicz, Z. (1994). “Genetic Algorithms + Data Structures = Evolution Programs,” Springer-Verlag, 2nd Edition, ISBN 3-540-58090-5.

Miki, M. Hiroyasu, T. and Jitta, T. (2003). “Adaptive Simulated Annealing for Maximum Temperature,” Proc. of the 2003 IEEE International Conference on Systems, Man & Cybernetics (SMC 2003), pp. 20-25.

Mohamad, N. H., Said, F. (2011). “A mathematical programming approach to crop mix problem,” African J. of Agricultural Research, Vol. 6(1), pp. 191-197.

Monette, J. N., Deville, Y. and Hentenryck, P. V (2009), "Just-In-Time Scheduling with Constraint Programming," in Proceedings of 19th International Conference on Automated Planning and Scheduling (ICAPS 2009), September 19-23, 2009, Thessaloniki, Greece.

Monyei, C. G., Adewumi, A. O., and Obolo, M. O. (2014). "Oil Well Characterization and Artificial Gas Lift Optimization Using Neural Networks Combined with Genetic Algorithm," Discrete Dynamics in Nature and Society, Vol. 2014, Article ID 289239, 10 pages.

Mustafa, A. A., Singh, M., Sahoo, R. N., Ahmed, N., Khanna, M., Sarangi, A. and Mishra, A. K. (2011). "Land Suitability Analysis for Different Crops: A Multi Criteria Decision Making Approach using Remote Sensing and GIS," Researcher, Vol. 3(12).

Naso, D., Surico, M., Turchiano, B. and Kaymak, U. (2007). "Genetic Algorithms for Supply Chain Scheduling: A Case Study in the Distribution of Ready-mixed Concrete," European Journal of Operational Research, Vol. 177(3), pp. 2069-2099.

Nyonyi, Y. (2010). "Modeling of hostel space allocation," African Institute for Mathematical Sciences (AIMS).

Oelofse, S. and Strydom, W. (2010). "A CSIR perspective on water in South Africa – 2010," CSIR Natural Resources and the Environment.

Pant, M., Thangaraj, R., Rani, D., Abraham, A. and Srivastava, D. K. (2008). "Estimation using differential evolution for optimal crop plan in Hybrid Artificial Intelligence Systems," Lecture Notes in Computer Science, Vol. 5271, pp. 289–297.

Pant, M., Thangaraj, R., Rani, D., Abraham, A. and Srivastava, D. K. (2010). "Estimation of optimal crop plan using nature inspired metaheuristics," World J. of Modelling and Simulation, Vol. 6(2), pp. 97–109.

Plante, R. D., Lowe, T. J. and Chandrasekaran, R. (1987). "The Product Matrix Traveling Salesman Problem: An Application and Solution Heuristics," Operations Research, Vol. 35, pp. 772-783.

Raju, K. S. and Kumar, D. N. (2004). "Irrigation planning using genetic algorithms," Water Resources Management, Vol. 18(2), pp. 163–176.

Ratliff, H. D. and Rosenthal, A. S. (1983). "Order-Picking in a Rectangular Warehouse: A Solvable Case for the Travelling Salesman Problem," Operations Research, Vol. 31, pp. 507-521.

Reddy, M. J. and Kumar, D. N. (2007). "Optimal reservoir operation for irrigation of multiple crops using elitist-mutated particle swarm optimization," Hydrological Sciences J. Vol. 52(4), pp. 686–701.

Rocha, M. and Neves, J. (1999). "Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation," Multiple Approaches to Intelligent Systems, Lecture Notes in Computer Science, Vol. 1611, pp. 127-136.

Ronconi, D. P. and Kawamura, M. S. (2010), "The single machine earliness and tardiness scheduling problem: lower bounds and a branch-and-bound algorithm", Computationaland Applied Mathematics, Vol. 29(2), pp. 107-124.

Sarker, R. and Ray, T. (2009). "An improved evolutionary algorithm for solving multi-objective crop planning models," Computers and Electronics in Agriculture, Vol. 68(2), pp. 191–199.

Schmitz, G. H. Schütze, N. and Wöhling, T. (2007). "Irrigation control: towards a new solution of an old problem," Vol. 5 of IHP/HWRP-Berichte, International Hydrological Programme (IHP) of UNESCO and The Hydrology and Water Resources Programme (HWRP) of WMO, Koblenz, Germany.

Sidhoum, S. K., Solis, Y. R. and Sourd, F. (2004). "Lower Bounds for the Earliness Tardiness Scheduling Problem on Single and Parallel Machines," Laboratoire LIP

Silva, J. D. L. (2003). "Metaheuristic and multiobjective approaches for space allocation," Ph.D. Thesis. University of Nottingham.

Singh, S. and Garg, D. (2011). "JIT System: Concepts, Benefits and Motivation in Indian Industries," International Journal of Management & Business Studies (IJMBS), Vol. 1(1), pp. 26-30.

Snyman, J. (2005). "Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms (Applied Optimization)," Vol. 97, Springer Science and Buisness Media Inc., ISBN: 0-387 24348-8.

Sunantara, J. D., Ramirez, J. A. (1997). "Optimal stochastic multicrop seasonal and intraseasonal irrigation control," J. of Water Resources Planning and Management, Vol. 123(1), pp. 39-48.

Soubeiga, E. (2003). "Development and application to hyperheuristics to personal scheduling," Ph.D. Thesis, University of Nottingham.

Syam, P, W and Al-Harkan, M. I. (2010). "Comparison of Three Meta Heurisics to Optimize Hybrid Flow Shop Scheduling Problem with Parallel Machines," World Academy of Science, Engineering and Technology, Vol. 62, pp. 825-832.

Tan, C. M. (2008). "Simulated Annealing", In-Tech Publisher, ISBN-13:978-953-7619-07 7.

Tasgetiren, F. M., Suganthan, P. N. and Pan, Q. K. (2007). "A Discrete Particle Swarm Optimization Algorithm for the Generalized Traveling Salesman Problem," Genetic and Evolutionary Computation Conference (GECCO) 2007, pp. 158-167.

Trevisan, L. (2011). "Combinatorial Optimization: Exact and Approximate Algorithms," Stanford University.

Tsai, C. Y. and Wu, M. C. (2010). "Applying a Two-Stage Simulated Annealing Algorithm

for Shelf Space Allocation Problems," Proceedings of the World Congress on Engineering 2010, London, U.K., Vol. 3.

Tsai, H. K., Yang, J. M., Tsai, Y. F. and Kao, C. Y. (2004). "An Evolutionary Algorithm for Large Traveling Salesman Problems," IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 34(4), pp. 1718-1729.

Tsubakitani. S. and Evans, J. R. (1998). "Optimizing Tabu List Size for the Travelling Salesman Problem," Computers Operational Research, Vol. 25, pp. 91-97.

Wardlaw, R., Bhaktikul, K. (2004). "Application of genetic algorithms for irrigation water scheduling," Irrigation and Drainage, Vol. 53(4), pp. 397–414.

Whelan, J. and Msefer, K. (1996). "Economic Supply & Demand," MIT System Dynamics in Education Project.

Yagiura, M. and Ibaraki, T. (2001) "On Metaheuristic Algorithms for Combinatorial Optimization Problems," Systems and Computers in Japan, Vol. 32(3), pp 33–55.

Yan, X., Zhang, C., Luo, W., Li, W., Chen, W. and Liu, H. (2012). "Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm," IJCSI International Journal of Computer Science Issues, Vol. 9(6), pp. 264-271.

Yang, X. S. (2010). "Nature-Inspired metaheuristic algorithms," 2nd Edition, Luniver Press, United Kingdom.

Yao, X. (1992). "Dynamic Neighbourhood Size in Simulated Annealing," International Joint Conference on Neural Networks (IJCNN '92), Vol. 1, pp. 411-416.

APPENDIX A

To explain the strategic search techniques employed by the eBPA, a hypothetical example will be used in performing the illustration. This example will broadly represents an optimization problem. Assuming that the optimization function $f(x)$ is a maximization problem, the objective will be to determine the optimal solution vector $x^* \in X$; X represents the solution space of feasible solutions. This solution space is constrained by linear and non-linear equations $g(x)\ \{\leq, =, \geq\}\ 0$. The intent of this illustration is to discuss the possible steps taken by the eBPA in locating the global optimum point.

Figure A.1 graphically illustrates the problem. The search space is seen to have three local optimum points; these are located at points $f(x_c)$, $f(x_g)$ and $f(x_j)$ respectively. The solution vectors used to determine these points are x_c, x_g and x_j respectively. The global optimum point is situated at point $f(x_j)$. The neighborhood regions underlining these local optimum points are N_1, N_2 and N_3 respectively. Falling within these neighborhood regions, are solution points which will be to explain the trajectory of the search. Amongst these is $f(x_a)$. This point will be the point of departure.

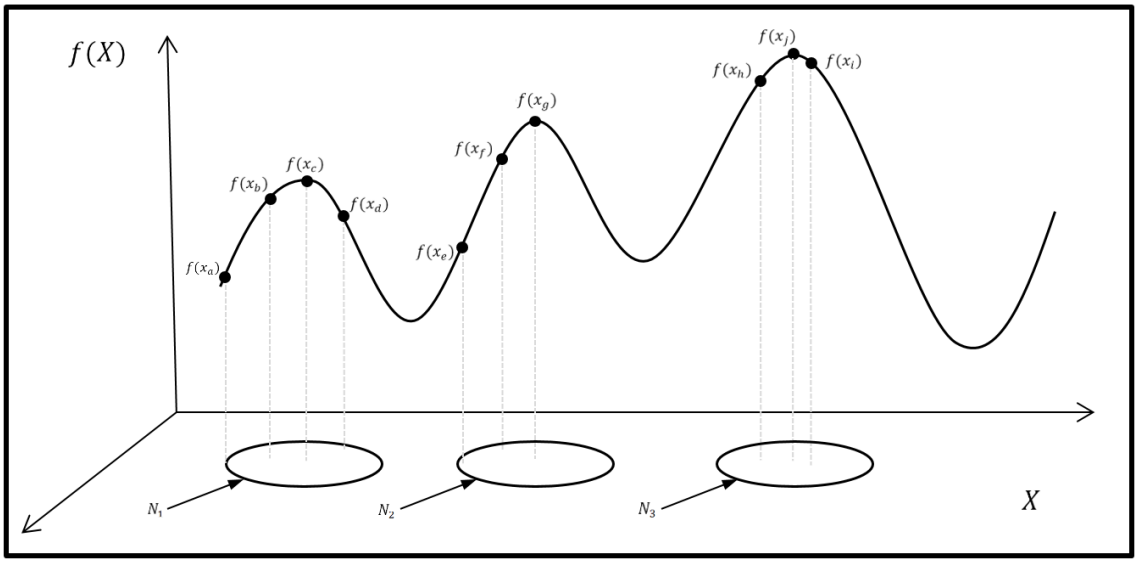


Figure A.1: Illustration of a hypothetical optimization problem having three local optimum points

The parameter settings of the eBPA will be as follows: the Performance List size (PL_size) will be 3, the probability factor (p_a) will be set at 0.05, and the $noOfIterations$ for which to execute will be set at 30.

The PL size will be strategically reduced by 1 every $reducePL$ number of iterations; let's assume that $reducePL = noOfIterations/PL\_size$. To trace through the steps of the algorithm, we maintain a table consisting of the decision variables. The table updates will monitor the variable state changes, in tracing through the search from its initial point of departure at point $f(x_a)$ to its optimum point $f(x_j)$. For consistency, the first iteration will be indexed at 0.

Table A.1 shows the initial values of the decision variables, at iteration 0. PL_0 is initialized to x_a, and $PL\_Fitness_0$ is initialized to $f(x_a)$. We assume solution vector x_a had been randomly generated. $working$ is set to x_a, all indices point to index 0, and $toggle$ is initialized to be true.

Table A.1: Variables state changes at iteration 0

		Iteration
		0
Performance List size	PL_size	3
Working Solutions	$working$	x_a
	$working^*$	-
The PL and $PL\_Fitness$ lists	0	$x_a \mid f(x_a)$
	1	-
	2	-
Solution Indices	$bestIndex$	0
	$workingIndex$	0
	$worstIndex$	0
Toggle Variable	$toggle$	true

Figure A.2 shows the sequences of events for iterations 1 and 2. The decision variable updates are seen in Table A.2. Arc 1, in Figure A.2 (which relates to iteration 1), shows a transition from point $f(x_a)$ to point $f(x_b)$. This transition had been determined by implementing a move on solution vector x_a; this determined solution vector x_b. Solution vector x_b, and its fitness value $f(x_b)$, got inserted

into the Performance List's at index 1. As the Performance List's are not fully populated as yet, the admittance criterion of the worst solution does not come into play. $workingIndex$ immediately points to the newly inserted solution, and now has the value of 1. Since point $f(x_b)$ has improved upon point $f(x_a)$, $bestIndex$ has now been set to 1. The $worstIndex$ remains at 0.

At this point, a random number in the range of [0,1] is generated and compared against p_a. Assuming that the probability condition did not get met, $toggle$ remains true. However, in these particular cases, even if $toggle$ were to be set to false, it would make no difference. Reason being, the updated working solution $working^*$ is the same solution which just got inserted into the memory structure. Therefore, this solution would be the next $working$ solution.

Arc 2, in Figure A.2 shows a transition from point $f(x_b)$ to point $f(x_d)$; this occurred as a move was implemented on solution vector x_b. This move transition relates to iteration 2. With index 2 in the Performance List's being unpopulated, the solution vector x_d, and its corresponding fitness value $f(x_d)$, got inserted at this index. As can be seen, solution vector x_d is a dis-improved move; it falls below the point of $f(x_b)$. Being a dis-improved move, the $bestIndex$ remains unchanged. The $workingIndex$ has now been updated to be 2. The $worstIndex$ remains unchanged, as $f(x_d)$ is still an improved point over $f(x_a)$.

At this point, with the Performance List's being fully populated, the admittance criterion of the worst solution will come into play. This is seen as a horizontal line across point $f(x_a)$. The level set by this horizontal line is the minimum requirement of acceptance across the entire search space X (i.e. any solution determined below this level will immediately be rejected). Assuming the probability condition remained unsatisfied, $toggle$ remains true.

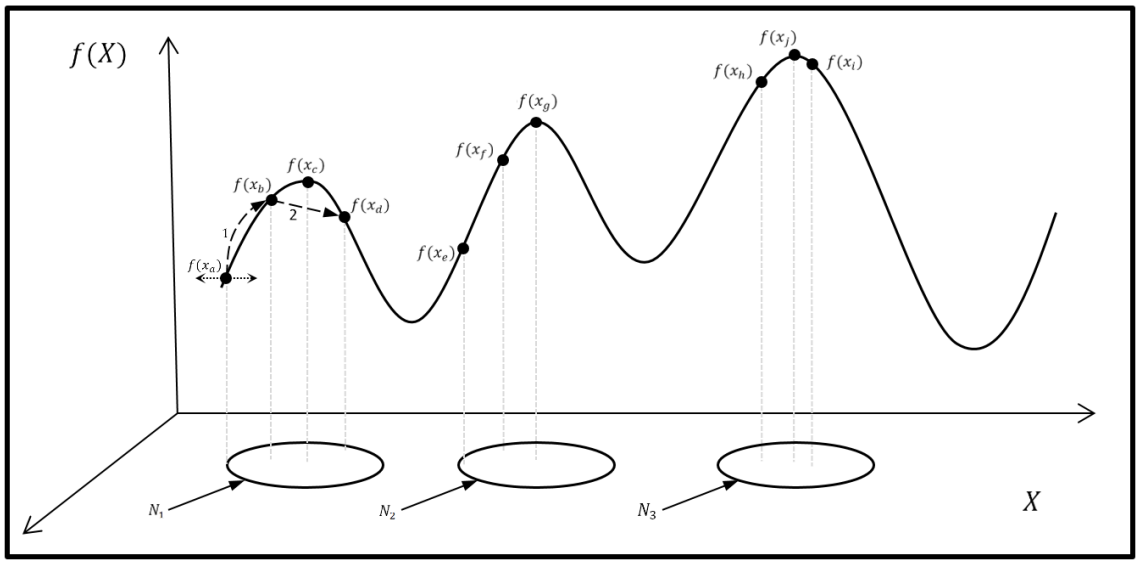


Figure A.2: Sequences of moves for iterations 1 and 2

Table A.2: Variables state changes for iterations 0 to 2

		Iterations		
		0	1	2
Performance List size	PL_size	3	3	3
Working Solutions	$working$	x_a	x_a	x_b
	$working^*$	-	x_b	x_d
The PL and $PL\_Fitness$ lists	0	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$
	1	-	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$
	2	-	-	$x_d \mid f(x_d)$
Solution Indices	$bestIndex$	0	1	1
	$workingIndex$	0	1	2
	$worstIndex$	0	0	0
Toggle Variable	$toggle$	true	true	true

Figure A.3 shows a move transition from point $f(x_d)$ to point $f(x_c)$. This is indicated by arc 3. $f(x_c)$ is the local optimum point of neighborhood structure N_1. Point $f(x_c)$ improves upon the fitness as indicated by the $worstIndex$ (i.e. $PL\_Fitness_{worstIndex} = f(x_a)$). Therefore, solution vector x_c, and its fitness value $f(x_c)$, has now been inserted at index 0 into the Performance List's. As an update

of the memory structure has just been realized, the $workingIndex$ immediately points to index 0. Since point $f(x_c)$ has improved upon point $f(x_b)$, the $bestIndex$ has also been assigned to index 0. At this step, $worstIndex$ needs to be re-determined; it resultantly had been assigned to index 2. Hence, the horizontal lower-bound of admittance just got elevated to point $f(x_d)$. Assuming the probability condition remained unsatisfied, $toggle$ remains true. The variable updates are seen in Table A.3, under iteration 3.

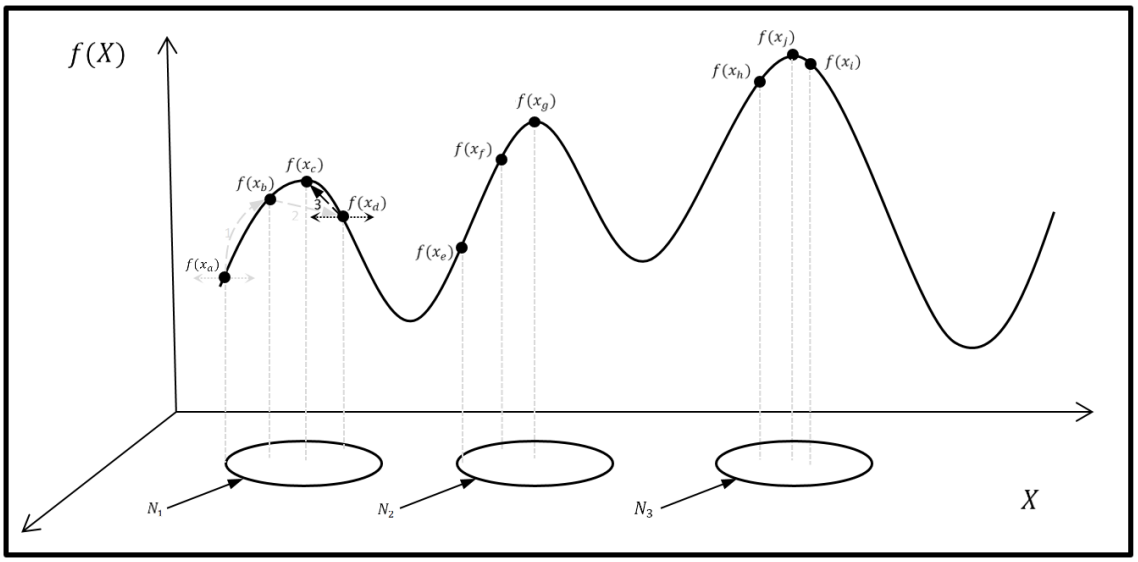


Figure A.3: Sequences of moves for iteration 3

Table A.3: Variables state changes for iterations 0 to 3

		Iterations			
		0	1	2	3
Performance List size	PL_size	3	3	3	3
Working Solutions	$working$	x_a	x_a	x_b	x_d
	$working^*$	-	x_b	x_d	x_c
The PL and $PL\_Fitness$ lists	0	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$	$x_c \mid f(x_c)$
	1	-	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$
	2	-	-	$x_d \mid f(x_d)$	$x_d \mid f(x_d)$
Solution Indices	$bestIndex$	0	1	1	0
	$workingIndex$	0	1	2	0
	$worstIndex$	0	0	0	2
Toggle Variable	$toggle$	true	true	true	true

Assuming no improved moves were determined at iterations 4, 5 and 6, Figure A.4 shows move transitions in breaking out of a possible cycle. This is indicated by arcs 4 and 5, and relates to iterations 7 and 8 respectively. Iterations 7 and 8 are documented in Table A.4. A cycle could occur if solution x_c continuously determined a dis-improved move to solution x_e, which then returned back to x_c. The move from point $f(x_c)$ to point $f(x_e)$ is indicated by arc 4. This move has occurred at iteration 7. At this iteration, we see that the probability condition $Random[0,1] < p_a$ has now been satisfied. Immediately, $toggle$ is set to false. This means that $working^*$ will become the next $working$ solution, at iteration 8. The acceptance of this move shows a transition from neighborhood region N_1 to neighborhood region N_2.

Arc 5, at iteration 8, shows an improved move made from point $f(x_e)$ to point $f(x_f)$. $f(x_f)$ improves upon the worst point $f(x_d)$, therefore an update of the Performance List's are required. The insert is performed at index 2. The $workingIndex$ and $bestIndex$ (as $f(x_f)$ is the best point found so far) get re-assigned to point to index 2. The $worstIndex$ gets re-assigned to point to index 1. With the probability condition being unsatisfied, $toggle$ remains true; $toggle$ had been reset to true at the point of having determined x_f.

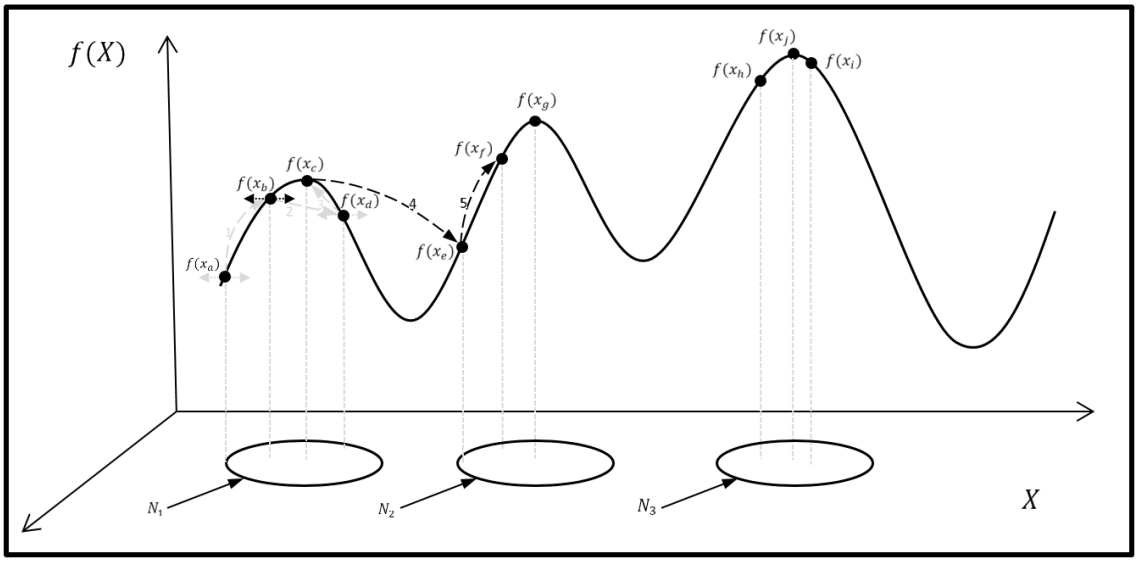


Figure A.4: Illustrating state transitions in breaking out of a possible cycle

Table A.4: Variables state changes for iterations 0 to 8

		Iterations						
		0	1	2	3	…	7	8
PL size	Performance List size	3	3	3	3	…	3	3
Working Solutions	$working$	x_a	x_a	x_b	x_d	…	x_c	x_e
	$working^*$	-	x_b	x_d	x_c	…	x_e	x_f
The PL and $PL\_Fitness$ lists	0	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$	$x_c \mid f(x_c)$	…	$x_c \mid f(x_c)$	$x_c \mid f(x_c)$
	1	-	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$	…	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$
	2	-	-	$x_d \mid f(x_d)$	$x_d \mid f(x_d)$	…	$x_d \mid f(x_d)$	$x_f \mid f(x_f)$
Solution Indices	$bestIndex$	0	1	1	0	…	0	2
	$workingIndex$	0	1	2	0	…	0	2
	$worstIndex$	0	0	0	2	…	2	1
Toggle Variable	$toggle$	true	true	true	true	…	false	true

Iterations 9 and 10, as seen in Table A.5 show no improvement. However, at iteration 10, the resize() condition gets satisfied. To perform the resize, solution swaps are required. Here, the solutions referred to by the $worstIndex$ (i.e. index 1) need to be swapped with the solutions referred to by the last index (i.e. index 2), in the Performance List's. As the solutions referred to by index 2 is the $bestIndex$ and $workingIndex$, these indices get re-assigned to point to index 1. The PL_size then gets reduced by 1; it now has the size of 2. At this point the $worstIndex$ gets re-determined, and points to index 0 which refers to point $f(x_c)$. The horizontal line correspondingly elevates to the level at point $f(x_c)$. This is seen in Figure A.5. This strategy further restricts the admittance criterion, making it more difficult for a memory structure to be updated.

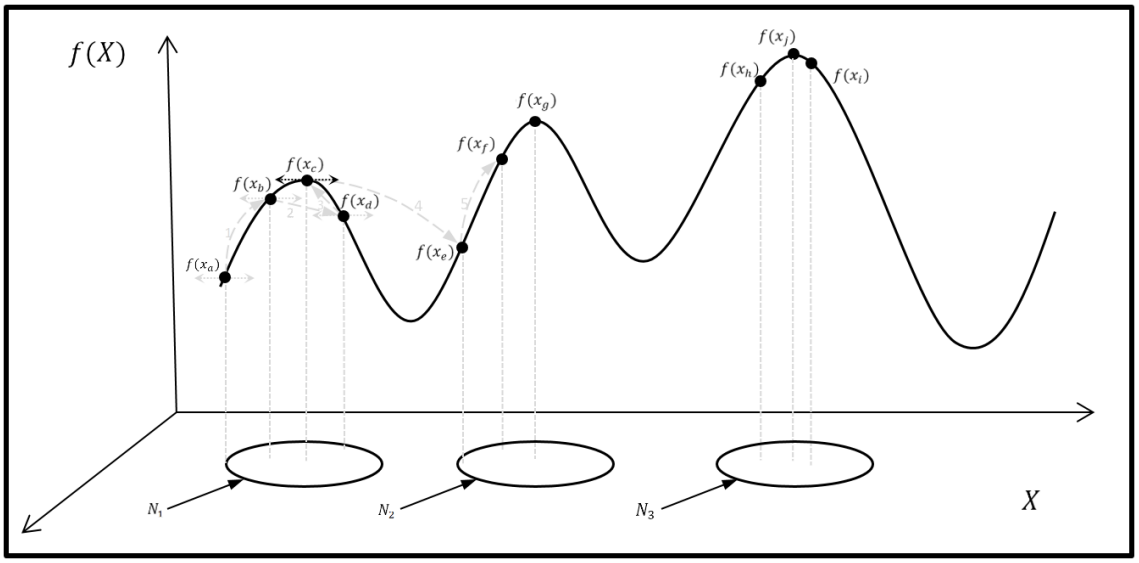


Figure A.5: Illustration of how the admittance criterion further restricts when the *PL* reduces in size

Table A.5: Variables state changes for iterations 0 to 10

		Iterations								
		0	1	2	3	…	7	8	9	10
PL size	Performance List size	3	3	3	3	…	3	3	3	2
Working Solutions	$working$	x_a	x_a	x_b	x_d	…	x_c	x_e	x_f	x_f
	$working^*$	-	x_b	x_d	x_c	…	x_e	x_f	$x_{f'}$	$x_{f'}$
The *PL* and *PL_Fitness* lists	0	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$	$x_a \mid f(x_a)$	$x_c \mid f(x_c)$	…	$x_c \mid f(x_c)$	$x_c \mid f(x_c)$	$x_c \mid f(x_c)$	$x_c \mid f(x_c)$
	1	-	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$	…	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$	$x_b \mid f(x_b)$	$x_f \mid f(x_f)$
	2	-	-	$x_d \mid f(x_d)$	$x_d \mid f(x_d)$	…	$x_d \mid f(x_d)$	$x_f \mid f(x_f)$	$x_f \mid f(x_f)$	~~$x_b \mid f(x_b)$~~
Solution Indices	$bestIndex$	0	1	1	0	…	0	2	2	1
	$workingIndex$	0	1	2	0	…	0	2	2	1
	$worstIndex$	0	0	0	2	…	2	1	1	0
Toggle Variable	$toggle$	true	true	true	true	…	false	true	true	true

In Figure A.6, arc 6 shows an improved move transition from point $f(x_f)$ to point $f(x_g)$; this occurs at iteration 16. The variable state changes are seen in Table A.6. Point $f(x_g)$ improves upon point $f(x_c)$, therefore the Performance List's get updated at index 0. The $workingIndex$ and $bestIndex$ (as point $f(x_g)$ is now the best solution determined so far) are assigned to point to index 0. The

$worstIndex$ is re-determined and now points to index 1. Point $f(x_g)$ is the local optimum point of the neighborhood region N_2. With the horizontal line moving up to point $f(x_f)$, the admittance criterion constrains even further. Assuming that the probability condition did not get satisfied, $toggle$ remains unchanged. A point of interest is that no solution from the neighborhood region N_1 will be accepted, as the point $f(x_f)$ supersedes the local optimum point $f(x_c)$.

At iteration 20, no improved solution gets registered. However, the resize() condition has now been satisfied. At this step, the solutions pointed to be the $worstIndex$ (i.e. index 1) need to be swapped with the solutions at the last index (i.e. index 1). Since both indices are the same, no swap is required. Therefore, the $bestIndex$ and the $workingIndex$ remain the same. The PL_size is then reduced by 1; it now has the size of 1. The $worstIndex$ gets re-determined, and now points to index 0. With the quality of the worst solution having been increased, the admittance criterion restricts yet further. This is indicated by the horizontal line being elevated to point $f(x_g)$. We assume $toggle$ remains unchanged.

The restriction of the admittance criterion forces the search to break beyond the local optimum point $f(x_g)$, of neighborhood region N_2, to point $f(x_h)$, of neighborhood region N_3. The transition is seen by arc 7 at iteration 23. Solution x_h, and its fitness value $f(x_h)$, get inserted into the Performance List's at index 0. The $workingIndex$, $bestIndex$ and $worstIndex$ remains at 0. $toggle$ remains true.

At iteration 24, a move is applied to solution vector x_h; this determined an improved solution x_i. This is seen by arc 8, which points to location $f(x_i)$. The Performance List's get appropriately updated. The $workingIndex$, $bestIndex$, $worstIndex$ and $toggle$ remain unchanged.

At iteration 29, greater levels of exploitation are experienced. This pushes the trajectory of the search to global optimum point $f(x_j)$. x_j and $f(x_j)$ get inserted into the Performance List's. At the end of this iteration, the termination criterion is satisfied. At this point, solution vector x_j is returned as the final solution.

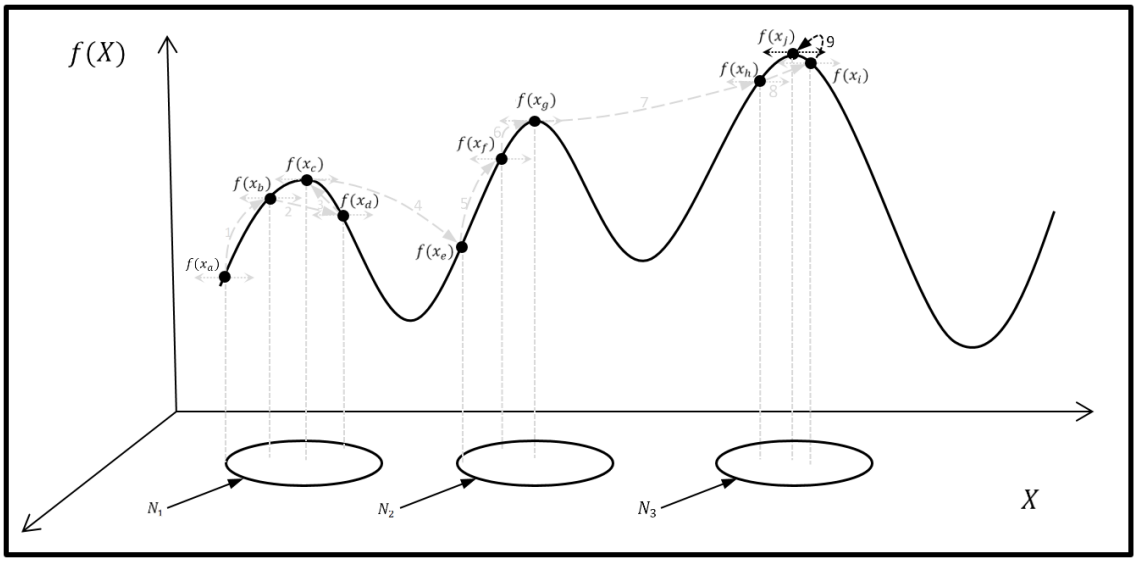


Figure A.6: Illustration of steps leading to the global optimum point

Table A.6: Variables state changes for iterations 0 to 29

		Iterations								
		…	16	…	20	…	23	24	…	29
PL size	Performance List size	…	2	…	1	…	1	1	…	1
Working Solutions	$working$	…	x_f	…	x_g	…	x_g	x_h	…	x_i
	$working^*$	…	x_g	…	$x_{g'}$	…	x_h	x_i	…	x_j
The PL and PL_Fitnes lists	0	…	$x_g \mid f(x_g)$	…	$x_g \mid f(x_g)$	…	$x_h \mid f(x_h)$	$x_i \mid f(x_i)$	…	$x_j \mid f(x_j)$
	1	…	$x_f \mid f(x_f)$	…	~~$x_f \mid f(x_f)$~~	…	~~$x_f \mid f(x_f)$~~	~~$x_f \mid f(x_f)$~~	…	~~$x_f \mid f(x_f)$~~
	2	…	~~$x_b \mid f(x_b)$~~	…	~~$x_b \mid f(x_b)$~~	…	~~$x_b \mid f(x_b)$~~	~~$x_b \mid f(x_b)$~~	…	~~$x_b \mid f(x_b)$~~
Solution Indices	$bestIndex$	…	0	…	0	…	0	0	…	0
	$workingIndex$	…	0	…	0	…	0	0	…	0
	$worstIndex$	…	1	…	0	…	0	0	…	0
Toggle Variable	$toggle$	…	true	…	true	…	true	true	…	true