University of Exeter
College of Engineering, Mathematics and Physical Sciences

Incorporating Domain Expertise into Evolutionary Algorithm Optimisation of Water Distribution Systems

Matthew Barrie Johns

June 2016

Supervised by Prof. Edward C. Keedwell and Prof. Dragan Savić

Submitted by Matthew Barrie Johns to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science, June 2016.

Abstract

Evolutionary Algorithms (EAs) have been widely used for the optimisation of both theoretical and real-world non-linear problems, although such optimisation methods have found reasonably limited utilisation in fields outside of the academic domain. While the causality of this limited uptake in non-academic fields falls outside the scope of this thesis, the core focus of this research remains strongly influenced by the notions of solution feasibility and making optimisation methods more accessible for engineers, both factors attributed to low EA adoption rates in the commercial space. This thesis focuses on the application of bespoke heuristic methods to the field of water distribution system optimisation. Water distribution systems are complex entities that are difficult to model and optimise as they consist of many interacting components each with a set of considerations to address, hence it is important for the engineer to understand and assess the behaviour of the system to enable its effective design and optimisation.

The primary goal of this research is to assess the impact that incorporating water systems knowledge into an evolution algorithm has on algorithm performance when applied to water distribution network optimisation problems. This thesis describes the development of two heuristics influenced by the practices of water systems engineers when designing water distribution networks with the view to increasing an algorithm's performance and resultant solution feasibility. By utilising heuristics based on engineering design principles and integrating them into existing EAs, it is found that both engineering feasibility and general algorithmic performance can be notably improved. Firstly the heuristics are applied to a standard single-objective EA and then to a multi-objective genetic algorithm. The algorithms are assessed on a number of water distribution network benchmarks from the literature including real-world based, large scale systems and compared to the standard variants of the algorithms. Following this, a set of extensive experiments are conducted to explore how the inclusion of water systems knowledge impacts the sensitivity of an evolutionary algorithm to parameter variance. It was found that the performance of both engineering inspired algorithms were less sensitive to parameter change than the standard genetic algorithm variant meaning that non-experts in the field of

meta-heuristics will potentially be able to get much better performance out of the engineering heuristic based algorithms without the need for specialist evolutionary algorithm knowledge.

In addition this research explores the notion that visualisation techniques can provide water system engineers with a greater insight into the operation and behaviour of an evolutionary algorithm. The final section of this thesis presents a novel three-dimensional representation of pipe based water systems and demonstrates a range of innovative methods to convey information to the user. The interactive visualisation system presented not only allows the engineer to visualise the various parameters of a network but also allows the user to observe the behaviour and progress of an iterative optimisation method. Examples of the combination of the interactive visualisation system and the EAs developed in this work are shown to enable the user to track and visualise the actions of the algorithm. The visualisation aggregates changes to the network over an EA run and grants significant insight into the operations of an EA as it is optimising a network.

The research presented in this thesis demonstrates the effectiveness of integrating water system engineering expertise into evolutionary based optimisation methods. Not only is solution quality improved over standard methods utilising these new heuristic techniques, but the potential for greater interaction between engineer, problem and optimiser has been established.

I would like to thank my supervisors Professor Edward Keedwell and Professor Dragan Savić for their continued guidance, encouragement and invaluable advice given to me throughout my time as their student. I am truly grateful for the academic and emotional support they have provided on this long and challenging journey and I can say wholeheartedly, that I could not have wished for better supervisors. Thank you also to Professor David Corne, Dr Prasad Tumula and Dr Alberto Moraglio for agreeing to be my examiners and for their insightful comments and considered corrections.

I must express my sincere gratitude to my family for their unwavering support and encouragement, without which I would have struggled to complete this thesis. I would also like to thank my friends and colleagues at the University of Exeter. Special thanks go to, Robert Allen, Stephen Mellor, Oliver Quinlan and Phillipa Wilson for being so supportive and for helping make the last few years a more enjoyable experience.

Contents

List of Tables

List of Figures

Publications

Some content within this thesis has previously been presented in the following publications:

- Johns M, Keedwell EC, Savic D. (2012) Adaptive Locally Constrained Genetic Algorithm for Least-Cost Water Distribution Network Design, *10th International Conference on Hydroinformatics*, Hamburg, Germany, 14th - 18th Jul 2012.

- Johns M, Keedwell EC, Savic D. (2013) Pipe Smoothing Genetic Algorithm for Least Cost Water Distribution Network Design, *Genetic and Evolutionary Computation Conference*, Amsterdam, 5th - 10th Jul 2013.

- Johns M, Keedwell EC, Savic D. (2014) Adaptive Locally Constrained Genetic Algorithm For Least-Cost Water Distribution Network Design, *Journal of Hydroinformatics*, volume 16, no. 2, pages 288-301, DOI:10.2166/hydro.2013.218.

- Johns M, Keedwell EC, Savic D. (2014) Interactive 3D Visualisation of Optimisation for Water Distribution Systems,*11th International Conference on Hydroinformatics*, New York, 17th - 21st Aug 2014.

- Johns M, Keedwell EC, Savic D. (2014) Multi-objective Pipe Smoothing Genetic Algorithm for Water Distribution Network Design, *11th International Conference on Hydroinformatics*, New York, 17th - 21st Aug 2014.

- Keedwell EC, Johns M, Savic D. (2015) Spatial and Temporal Visualisation of Evolutionary Algorithm Decisions in Water Distribution Network Optimisation, *VizGEC Workshop, Genetic and Evolutionary Computation Conference, 2015*, Madrid, Spain, 10th - 15th Jul 2015.

Chapter 1: Introduction

The human brain is the product of nearly seven million years of evolution and is extremely adept at problem solving; be it a seemingly simple problem such as packing a suitcase or a more complex problem such as traversing London using the public transport system. In the case of navigating London, a commuter generally has two objectives; reach their destination and minimize the travel time. The first objective is easily achievable; however choosing a route that will get you to your destination in the shortest possible time becomes increasingly difficult as the distance the commuter has to travel is increased. This is due to the fact that as the distance is increased, the number of available routes increases considerably. Although humans are relatively good at solving these types of problem, as the number of possible solutions grows it becomes increasingly hard for the human brain to formulate an optimal solution to the problem. However, with the rise of modern computing techniques it has now become increasingly easier to produce high quality solutions to complex problems due to the ability these systems have to evaluate millions of solutions in a relatively short amount of time, thus increasing the chance of finding a more efficient solution to the problem at hand. The field of optimization in computer science has utilized this computing power and produced techniques for solving complex problems using algorithms that thoroughly search the space in which the solutions to a problem reside.

These optimization algorithms utilize rules of thumb, or heuristics, to effectively search the problem solution space. Algorithms employing these relatively simple heuristics have proven to be very effective in finding solutions for problems such as route finding [1] and are actively employed in the route planning software in web mapping services and satellite navigation units. However, for problems of increased complexity, more general algorithms have been developed that combine multiple heuristics, this class of algorithm is commonly referred to as meta-heuristics. Evolutionary Algorithms (EAs) [2][3][4] are one such class of meta-heuristic algorithm, originally inspired by the process of natural selection, this type of meta-heuristic employs heuristics that mimic genetic processes to evolve a set of solutions to a problem.

Meta-heuristics, such as EAs have been applied to numerous problems ranging from tasks such as route finding and timetabling to more complex problems such as the design of radio antennas [5] and supersonic aircraft wings [6]. The inherent ability offered by meta-heuristics to provide a generalized approach to solving a wide array of problems with little to no adaptation has led to an explosion of research and utilization outside of computer science in fields such as biology, finance [7], art [8], music [9] and engineering.

Within the field of engineering for example, meta-heuristics have been applied to a wide range of design problems such as electronic circuit optimization [10], composite material design [11] and water systems optimization [12]. The majority of this research and development is being done within academic institutions and although EAs and similar techniques are now being utilized more in the commercial sector, they are mainly being used as design tools and are commonly perceived by the engineer as “black boxes” into which a problem is inserted and the solution extracted. It is usually the case where an engineer will have to manually tweak the solution produced by the algorithm to ensure engineering feasibility; this is due to it being difficult to fully define all of the objectives of the problem, as many may in fact be situationally sensitive and requires the knowledge of an experienced designer to effectively evaluate. This is a common problem when deploying an optimisation technique to solve real-world problems, as it is difficult to integrate the expert knowledge of an engineer or designer into the optimisation process. This thesis sets out to address this issue by developing a platform upon which problem specific heuristics commonly utilized by engineers can be integrated into an EA to increase the performance and engineering feasibility of the resultant solutions.

1.1 Research Objectives

It is evident from the literature that the use of evolutionary algorithms to solve real-world water systems engineering problems is a growing area of focus. However as a number of researchers have highlighted [13], as problem complexity grows so does the need for the inclusion of domain specific knowledge to ensure high quality practical solutions. Although some researchers in the field of hydroinformatics have started to explore the use of water systems knowledge to aid in the optimisation of water distribution network

design problems, the focus of knowledge application is mainly concentrated on the initial generation of solutions and not the genetic operators such as mutation or crossover. The use of knowledge guided mutation operators for example has proven to be effective in a number of other domains and therefore presents an interesting avenue of exploration in the field of water system optimisation. This leads to the first research question posed in this thesis:

To what extent does the incorporation of expert water systems knowledge into an evolutionary algorithm mutation operator impact performance of the optimisation of water distribution network problems?

An important consideration when using any evolutionary algorithm is the algorithm's robustness to parameter value change. EAs will have a set of parameters that result in optimal algorithm performance for a specific problem; however the optimal parameter settings will vary as the complexity and size of a problem is changed. This poses a problem when applying an EA to a real-world problem as it is often infeasible to manually tune the parameters of an algorithm due to computational complexity and resultant run time. Parameter tuning in EAs has long been studied in the wider field of computer science; however there has only been a very limited amount of research conducted into the selection of EA parameters for water distribution system optimisation. The less sensitive an algorithm is to parameter variation, the greater likelihood that a quality solution is found. This is an important feature of any algorithm especially in a practical setting, as an engineer would not necessarily have the time or resources to conduct extensive experimentation to identify the optimal configuration of parameter values. This leads to the second research question posed by this work:

How does the inclusion of water systems knowledge impact the sensitivity of an evolutionary algorithm to parameter variance?

As stated earlier, although practising engineers have started to utilise techniques such as EAs for the optimisation of real-world problems, their understanding of such methods is commonly quite limited, seeing an EA as a "black box" that generates solutions to aid in the design process. It is proposed that providing engineers with a greater insight into how an EA operates in the

design space will result in better understanding and pave the way for greater interaction between engineer and algorithm. Much research has been conducted into the visualisation of objective and decision spaces [14] however, less research has been conducted into the visualisation of the design space; this is particularly apparent in the field of hydroinformatics. This leads to the third research question posed in this thesis:

Can state of the art visualisation techniques provide engineers with a greater insight into the operation and behaviour of an EA when applied to the problem of water distribution network design?

1.2 Thesis structure and novelties

This thesis is divided into eight chapters, including this introductory chapter. The core component of this thesis describes the development of a number of heuristics influenced by the practices of water systems engineers when designing water distribution networks with the view to increasing an algorithm's performance and resultant solution feasibility. In addition, a novel three-dimensional visualisation technique is applied to pipe based water systems with the view to allowing an engineer to visualise the various parameters of a network and also allowing the user to observe the behaviour and progress of an iterative optimisation method.

Prior to this a comprehensive review of the relevant literature relating to the work in this thesis is presented in Chapter 2. This includes a background study into the fields of optimisation, heuristics, meta-heuristics, constraint handling, water distribution network design, knowledge guided search and interactive visualisation. The results of the review have been used to identify gaps in the knowledge base leading to the formulation of the research questions presented in the previous section.

Chapter 3 presents the Water Distribution Network (WDN) design problem in detail and explores the methodologies applied to the problem from the literature. This information is utilised to develop both a single-objective Steady State Genetic Algorithm (SSGA) and Multi-objective Non-dominated Sorting Genetic Algorithm (NSGA-II) for application to the WDN design problem. Both of

these algorithms are intended to be a platform to which the knowledge based mutation operators can be applied. Both SSGA and NSGA-II also provide a baseline for comparison for the new engineering inspired algorithms developed later in this work. Following the formulation of the two standard Genetic Algorithms, a set of least-cost WDN design benchmark problems from the literature are presented which will form the testbed on which all algorithms will be assessed. The problems were selected for their varying size and complexity to enable thorough analysis of the engineering inspired algorithms. The final section of this chapter documents the development of two novel engineering inspired heuristic methods which are applied to both SSGA and NSGA-II in later chapters. These engineering inspired heuristics are formulated using observations from water systems engineers and techniques from the water systems literature.

Chapter 4 firstly details the results of a set of extensive parameter tuning experiments designed to analyse the performance and sensitivity of the SSGA. These experiments pave the way to addressing the second research question by providing insight into the sensitivity a Genetic Algorithm (GA) has to parameter variance when applied to WDN optimisation problems. The experiment set contributes to the body of knowledge as a study of this type has not been presented in the hydroinformatics literature. The next section in the chapter presents the development of two novel engineering inspired GAs based on the water systems heuristics presented in chapter 3. The first of which is the Adaptive Locally Constrained Genetic Algorithm (ALCO-GA), an adaptive variant of the Locally Constrained GA (LCO-GA) which uses hydraulic pressure information to promote hydraulically feasible solutions. The second algorithm presented in chapter 4 is the Pipe Smoothing Genetic Algorithm (PSGA) which uses the idea that in gravity fed WDNs pipe diameters smoothly transition from the source to the extremities. Both algorithms are tested on a benchmark WDN problem from the literature to assess performance compared with the SSGA and tune the level of heuristic application. The two engineering knowledge based GAs presented in this fourth chapter suggests that the incorporation of water system knowledge has the potential to improve algorithm performance, both in terms of cost and engineering feasibility, paving the way to addressing both the first and second research questions posed in section 1.1.

Chapter 5 presents an extensive set of experiments designed to explore the effect the single-objective engineering inspired heuristic GAs have on a large range of problems from the literature. The first set of experiments focus on the comparison between the engineering knowledge based algorithms (ALCO-GA & PSGA) developed in chapter 4 and the SSGA. The experiment is designed to explore the performance and behaviour of both ALCO-GA and PSGA on a range of problems, varying greatly in size and complexity. This set of studies provides significant insight into the impact engineering knowledge has on the performance of an EA when incorporated through the mutation operator and represents a significant proportion of the novel contribution offered by this research. The second set of experiments presented in this fifth chapter is designed to investigate the sensitivity of the algorithms to variation in parameter values using exhaustive search and sensitivity analysis, with the view of addressing the second research question posed in section 1.1.

Chapter 6 investigates the performance of the engineering heuristic based approaches when applied to multi-objective water distribution network design problems. The chapter includes the formulation of both a dual-objective and tri-objective WDN design problem including a novel network smoothness objective function inspired by the pipe smoothing heuristic presented in chapter 3. Extensive experimentation on the new engineering inspired multi-objective algorithms (MOALCO-GA & MOPS-GA) is conducted and the results compared with NSGA-II. The chapter concludes with a summary of results and discussion about how algorithm performance is affected by the incorporation of water systems knowledge into multi-objective formulations of water distribution network design problems.

Chapter 7 presents the development of a novel interactive three dimensional tool for the visualisation of water distribution networks and evolutionary optimisation. The chapter aims to address the third research question stated in section 1.1 by presenting a user friendly tool (WDNet3D) capable of not only visualising water systems and their operation but also enabling the visualisation of evolutionary optimisation in the design space, providing the engineer with a greater insight into algorithm operation and thus potentially aiding the design process. The first section of the chapter presents the development of WDNet3D and explores the benefits of a 3D representation

over that of the more traditional 2D plan view approach employed by current WDN design software. The second part of the chapter introduces a novel approach to the visualisation of water distribution network optimisation, visualising the actions of an algorithm by aggregating changes made to the network over a course of a GA run. The chapter ends with a summary section and a discussion regarding possible application of the techniques presented to other problem domains.

The thesis summary is presented in chapter 8. The conclusions reached from the application of water systems knowledge to the mutation operator of both single-objective and multi-objective genetic algorithms are also presented. The chapter concludes with a summary of future research recommendations.

Chapter 2: Background

This chapter presents a review of the applicable literature relating to the field of water distribution network design and a number of optimization techniques ranging from more traditional heuristic methods to advanced constraint handling and knowledge guided search practices. In addition to these areas of study, literature from the field of interactive evolution is investigated. It is the aim of this review to identify potential gaps in the body of knowledge to enable the formulation of the research questions posed by this work. The following list presents the structure of this literature review:

- A synopsis of what would be considered traditional optimization methods, including heuristic optimization, especially meta-heuristics with an emphasis on genetic algorithms.
- A summary of multi-objective optimization, including multi-objective evolutionary algorithms.
- An exploration of the field of least cost water distribution network design, from origins to current practices.
- A review of optimization techniques that have been applied to the least cost water distribution network design problem.
- A description of various constraint handling techniques employed by evolutionary algorithms.
- An exploration of knowledge guided search techniques concentrating on current knowledge incorporation in the field of hydroinformatics.
- A review of interactive evolution methods being used in the field of optimization.

2.1 Optimization

The field of optimization covers a wide range of scientific research and encompasses a large number of approaches and methods, all of which have one goal; to find the best performing solution for any given problem. However in cases where the problem complexity is very high it is normally unlikely that an optimization method will locate the true optimum solution but will still aspire to find the best solution with available computational resource. In computer

science, the field of optimization often refers to a class of algorithms designed to generate solutions to the given problem. An optimization problem in this field is often defined as a mathematical function which represents the quantifiable components of the problem to enable the evaluation of a resultant solution. In the field of evolutionary computation, solutions are often referred to as individuals due to being derived from the biological mechanisms of evolution. In the literature these individuals are frequently defined by two vectors which correspond to the parameters of the solution and the overall solution quality, also known as the objective value. The parameter vector is an encoded representation of the solution which is then applied to the problem and evaluated. The basic form of which an optimization problem can be formulated is presented below:

$$given\ f: X \rightarrow \mathbb{R}$$

$$minimize\ f(\boldsymbol{x})\ where\ \boldsymbol{x} \in X \qquad (1)$$

In this representation, the result of $f(x)$ is the objective value which is to be minimized; where smaller values are considered better than larger values. The function $f(x)$ evaluates solution, x, based on the values contained within vector x. The arrangement of vector x and the function $f(x)$ will vary for each specific problem. The elements in x are commonly represented utilising the following structures: Boolean, integer, real-valued categorical and permutation encodings. The encoding selected for use is dependent on the specific optimisation problem as each encoding method has characteristics that lend themselves to specific problems. A problem may utilise multiple encoding methods, however it is more common to only utilise one.

2.2 Heuristic Optimization

Many methods for solving mathematical optimization problems have been presented over the years, one of the first such methods was Linear Programming (LP), introduced by Kantorovich in 1940 [15]. Linear Programming and similar methods have proven popular for solving numerous optimization problems due to their efficiency and guaranteed convergence. However these techniques are somewhat limited as they can only be used to solve convex, continuous, differentiable functions. When presented with more complex

problems of a multimodal nature, alternate methods must be considered. Problems of convex nature only have one optimal objective value which LP and similar methods are able to find in polynomial time [16]. However, in the case of a non-convex problem, methods such as LP get trapped in local minima, which are optimum from the perspective of the algorithm, although the achieved solution is actually worse than the true global optimum. In 1951 Robbins & Monro [17] addressed the issue of non-convex optimization problems by proposing one of the first heuristic algorithms. The proposed technique utilized stochastic sampling to explore the optimization search space which went some way to surmount the problems with LP methods on non-convex and/or non-differentiable functions and was capable of avoiding the trap of local minima. Additionally, stochastic heuristic techniques can in theory find all global optima in problems where multiple optimal solutions exist in the search space.

Stochastic Approximation [17] and later Random Search [18] broke ground in the field of meta-heuristic search techniques which started to gain traction in the latter half of the 20th century. Meta-heuristics are a type of heuristic method that uses an iterative method to generate solutions for a problem which are then evaluated and used to guide later sampling; achieved through an amalgamation of heuristic methods. By comparison, the term heuristic method can commonly refer to a class of algorithm that produces solutions to a problem. Burke, et al. [19] notes that heuristics can refer to algorithms which are limited to only producing a single solution and can be deterministic in operation, resulting in the heuristic always producing an identical solution to each problem. Meta-heuristics, by contrast are iterative techniques which keep a set of candidate solutions which are constantly updated and compared with newly created solutions until the stopping criteria of the algorithm is achieved. Unlike LP and similar methods, heuristic and hence meta-heuristic techniques are not certain to converge and therefore are not guaranteed to locate the true global optima. Instead, it is anticipated that these algorithms produce a set of highly optimal solutions within a feasible timeframe.

2.2.1 Meta-heuristics

Pearl [20] presents a simple model that can be utilized to describe any stochastic, iterative heuristic (meta-heuristic) by outlining three principal

features of a meta-heuristic: the encoding, the operators or rules of production and the control strategy. These elements are more commonly known in recent literature as solutions, heuristic operators and selection strategies. The formulation of most modern meta-heuristics tends to include these elements and generally conform to this model.

Introduced by Glover, the term meta-heuristic was used to describe higher-level heuristic techniques; however the paper did not present a definitive definition and the term meta-heuristic remained a fluid concept. In 1996 Osman & Laporte [21] described a meta-heuristic as an iterative generation process that guides subordinate heuristics through learning mechanisms and intelligent strategies. In Voß, et al [22] added to this definition through the inclusion of high level procedures within the subordinate heuristics. At the same time, Stützle [23] described a meta-heuristic as a biased random search, enabling the exploration of the search space through diverging moves. Following these, and other papers, Blum & Roli [24] produced a summary of the key features of a meta-heuristic: usually non-deterministic; avoid local minima; efficient exploration; domain specific awareness to form heuristics; not problem specific; and search guidance through the use of memetic features.

Further to the summary of meta-heuristic definitions presented by Blum & Roli, [24] their paper also proposes a classification of meta-heuristic techniques which is founded upon 5 orthogonal attributes which can be used independently or amalgamated to define a meta-heuristic. The first feature is determined between nature inspired and non-nature inspired techniques, however the authors suggest this distinction is often hard to qualify. The next feature is a decision between trajectory search approaches and population based techniques. The third distinction is made between static and dynamic based objective function implementations. In the case of a dynamic objective function, the algorithm alters the objective function as the search progresses. The fourth classification is based on whether the algorithm uses a single representation of the search space or multiple representations. The final feature is whether the technique utilises memory or not. For example, an algorithm that stores information past the previous iteration such as past populations can be said to use memory.

Although there is a somewhat large variance in terms of the different classifications presented in the literature to what defines a meta-heuristic, the majority of definitions agree on the heuristic foundation from which meta-heuristic algorithms are derived. The literature has presented a large number of meta-heuristic approaches over the past 4 decades, most of which offer a great deal of diversity. Some of the most notable algorithms are: Genetic Algorithms [2], Simulated Annealing [25], Tabu Search [26][27] and Ant Colony Optimisation[28]; all are capable of tackling combinatorial problems. Meta-heuristics have also been developed to solve continuous problems, these include: Evolutionary Strategies [3][29], Particle Swarm Optimisation [30] and Differential Evolution [4].

2.2.2 Genetic Algorithms

In the 1950s and 1960s scientists had postulated that fundamental evolutionary processes could be applied as a tool for the optimisation of engineering problems. This research grew independently into three main streams of research: Evolutionary programming (EP) [31]; evolution strategies (ESs) [32][33][34][35]; and genetic algorithms (GAs) [2].

Genetic algorithms were invented and developed by John Holland, his colleagues and his students at the University of Michigan in the 1960s and 1970s [36][37]. In his book entitled *Adaptation in Natural and Artificial Systems*, Holland [2] reasons that the ability of organisms to adapt to highly uncertain environments stems from evolution by natural selection, he concluded that biology could be used as a metaphor for artificial systems. Using the field of genetics as inspiration Holland began to translate the fundamental genetic structures and operators into code; he then applied these principals to the development of software.

Genetic algorithms are a form of search algorithm which follows a set of reproductive plans and consists of a number of generalised genetic operators. To implement a genetic algorithm, it is first necessary to code the decision variables describing a trial solution as a string of bits or “chromosome” consisting of “genes.” A collection of chromosomes are generated, resulting in a “population” of individual solutions which the genetic algorithm assesses;

assigning an assessment value or “fitness” to each solution present in the population. In a basic genetic algorithm, a pair of individuals from the population are selected to be parents according to their fitness. Two genetic operators are then applied to the selected parents: crossover and mutation. Crossover is the partial exchange of corresponding segments of bits between the two parent chromosomes that results in the production of two offspring chromosomes. Mutation is an operator that randomly flips some of the bits within a chromosome to ensure that potential solutions are not lost.

Although the term “genetic algorithm” has been used by the evolutionary-computation community for over 30 years, the exact definition of the genetic algorithm fails to be agreed [37]. However, there are some common elements present in the majority of methods which call themselves genetic algorithms: population of chromosomes, selection according to fitness, crossover to create offspring, and mutation of offspring. Variations of the classical genetic algorithm set out by Holland have been used in a wide variety of scientific and engineering problems. A large number of these fall into the area of optimisation, including numerical optimisation and combinatorial optimisation problems. These can range from complex engineering design problems such as aero-dynamical optimisation of transonic wing design [38] to relatively simple scheduling problems such as high school timetabling [39]. In the mid 1990s genetic algorithms were starting to be applied to the design of water distribution networks, a problem which had previously only been approached with classical optimisation techniques.

2.2.2.1 Selection Methods

Evolutionary algorithm operators reside within two broad classes; selection schemes and variational schemes [40]. The combination and implementation of these operators is an important consideration when applying an evolutionary algorithm to a problem. The most commonly known genetic algorithm selection operators in current use are tournament selection, truncation selection, roulette wheel selection, and Boltzmann selection. The majority of modern evolutionary algorithm codes employ forms of tournament and/or truncation selection. These selection schemes are scaling invariant and when used in combination they have proven to be implicitly elitist ensuring the fittest population members

survive into the next generation. Reed et al. [41] summarises the theoretical relationships developed for population sizing and timescale analysis which model the effects of the primary operators of a simple genetic algorithm. This research and others [42][43] shows that elitism and scaling invariance are important properties that can enable convergence to near-optimal or optimal solutions. The roulette wheel selection method and other stochastic proportionate schemes have been applied in many early water resources applications; however research has since illuminated significant limitations with these methods [40]. Unlike tournament selection, roulette wheel selection can be severely affected by fitness scaling. Applications employing constraint violation penalties often result in the occurrence of a single super-solution which dominates selection probability, frequently resulting in premature convergence [44]. Also solutions with similar fitness values produce selection probabilities nearly identical to one another resulting in search drift or stall [40]. For these reasons roulette wheel selection is avoided in more modern applications of evolutionary algorithm. Boltzmann selection has been used in a wide range of water resources applications [45][46][47] with its basic sampling scheme originating from simulated annealing [25].

2.2.2.2 Evolutionary Operators: Mating and Mutation

In modern evolutionary algorithms, mating and mutation approaches are heavily influenced by how decision variables are encoded: Binary vectors, integer vectors, real-coded vectors or mixed integer/real vectors. Early work in the field of genetic algorithms focused largely on a binary framework which influenced the type of mating and mutation operators that were employed. Theoretical work [48][49] in binary genetic algorithm applications has shown that uniform crossover is often preferred as it adds more search pressure to explore new regions of the application's decision space. Uniform crossover combines the strings of two binary parent chromosomes by randomly choosing a locus and exchanging the sub sequences before and after the locus resulting in two offspring [37]. With respect to real-coded representations, there are two common types of mating operator in use: (1) Crossover and (2) intermediate recombination. Intermediate recombination originated from evolutionary strategy [50] literature and as a result is more commonly employed for real or mixed integer formulations. Through the use of statistical averaging and decision

variable perturbation schemes, intermediate recombination combines multiple real-valued parent vectors to produce a new set of individuals [40]. The current state-of-the-art in real-valued intermediate recombination strategies include; simulated-binary crossover [51], parentcentric crossover [52] and self-adaptive weighted recombination [53]. The use of these operators in the field of water resources has shown they can potentially do better than binary-coded crossover and can increase search efficiency for binary decisions [42][43][54]. Mutation is an operation that is performed on a singular individual solution to produce a new candidate solution. Binary genetic algorithm applications primarily utilise jump mutation, where the operator randomly flips some of the bits in the chromosome. In the case of real-value coded evolution strategy algorithms, the Gaussian mutation method is the most commonly employed [55], in which it generates new candidate solutions by adding normally distributed perturbations to the decision variables [40].

2.2.2.3 Parameters and Solution Evaluation

An important challenge that the user faces, is specifying the parameters that control the search of an evolutionary algorithm. These parameters include population size, run length and the probability of mating and mutation. Aly and Peralta [56][57] show that given the constraints of a computationally intensive application, selecting sturdy search operator parameters is a challenging and critical task. Reed et al. [41] proposed a methodology comprised of three steps for parameterising binary-coded genetic algorithms, as long as tournament selection, uniform crossover and jump-mutation were utilised. The work highlighted some important considerations when parameterising evolutionary algorithms: Random seed variability should be minimised to ensure similar results are achieved despite the initial randomly generated population; increasing the size of the population frequently can progress the reliability of a single random seed search; the duration of a run should be influenced by the amount of decision variables present in the model; and the computational time taken to process design evaluations should steer run duration and population size. Some useful modifications to the Reed et al. [41] parameter methodology has been shown in more recent research [58][43][54][59]. A number of these studies have shown that preconditioning the search by injecting good solutions into the population can produce reliable and efficient search results.

2.3 Multi-objective Optimization

Single-objective meta-heuristic techniques have been used to solve a wide range of real-world problems most notably in the fields of operations research and engineering. Many of these problems utilised a single scalar objective value to produce a measure of quality for a given solution. However, most real-world problems have multiple criteria that need to be assessed and therefore require multi-criteria decision analysis methods such as goal programming [60] to combine quality measures into a single objective formulation. Whilst methods facilitating the combination of objective values or the enforcement of constraints allow the location of quality solutions, techniques such as these prevent an algorithm identifying solutions with differing characteristics: When the performance of a solution is represented by a single objective it is impossible for the optimisation algorithm to differentiate between solutions with the same objective value. Take for example, the design of an aircraft wing with two criteria, maximise lift whilst minimising drag, when combined into a single objective formulation it is possible for two very different solutions to have the same objective value; high lift with high drag and low lift with low drag. In cases such as this, there is normally a trade-off between criteria and this has led researchers to develop multi-objective optimisation methods.

Multi-objective problems consist of multiple objective functions which are assessed in parallel, where the measure of a solution's quality is represented by a vector of objective values unlike the scalar value representation of single-objective problems. Utilising a vector of objective values enables a problem to assess multiple objectives separately, enabling the exploration of trade-offs between objectives. A multi-objective optimisation problem can be generally formulated as follows:

$$
\begin{aligned}
&given\ [\ f_1 : X \to \mathbb{R}, f_2 : X \to \mathbb{R}, f_3 : X \to \mathbb{R}, \dots, f_n : X \to \mathbb{R}\ ] \\
&minimize\ f_1(\boldsymbol{x}) where\ \boldsymbol{x} \in X \\
&minimize\ f_2(\boldsymbol{x}) where\ \boldsymbol{x} \in X \qquad (2) \\
&\vdots \\
&minimize\ f_n(\boldsymbol{x}) where\ \boldsymbol{x} \in X
\end{aligned}
$$

This formulation enables an optimisation method to explore the trade-offs between the different objectives ultimately producing a set of solutions with similar performance but differing objective values: A single solution can then be chosen from the set of solutions based on the user's discretion.

2.3.1 Multi-objective Evolutionary Algorithms

In 1989 Goldberg [36] suggested a non-dominating sorting method for Pareto optimal solutions. This technique first identifies all of the Pareto optimal solutions in a population and assigning them with a rank of one and then removing them. The remainder of the population are assessed, identifying the non-dominated solutions, assigning them a rank of two and removing them from consideration. This process is repeated until all solutions in the population have been assigned a rank. A variant on this ranking technique, multi-objective ranking, was developed by Fonseca & Fleming [61] and implemented in their Multi-Objective Genetic Algorithm (MOGA). This technique assigned a rank to each individual according to the number of individuals that dominate it. In this approach an individual's rank is simply the number of individuals that dominate it plus 1, for example, if an individual is dominated by 4 solutions then its rank will be 5. This approach generates a larger range of ranks than Goldberg's original method and also discourages clusters of high density solutions. These non-dominated sorting methods had proven to be quite effective and soon a number of more advanced EAs were developed offering improvements to the technique. One of the most notable of these algorithms was the Non-dominated Sorting Genetic Algorithm (NSGA) [62] followed by the improved and optimised Fast Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) [63]. In addition to these GAs a number of Evolutionary Strategies (ES) employed non-dominated selection techniques; some of the most notable examples include the Strength Pareto Evolutionary Algorithm (SPEA) [64] and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [65], Pareto Archived Evolutionary Strategy (PAES) [66] and the Pareto Envelope Selection Algorithm (PESA) [67]. Both type of EA (GA and ES) have been applied in a wide range as cases and have proven to be valuable when solving multi-objective optimisation problems [68].

2.4 The Design and Optimization of Water Distribution Networks

Water is mainly transported through pipelines and distributed through complex underground pipe networks. Water distribution networks of this kind are a relatively recent addition to modern civilisation and have only become common in the last couple of centuries [69]. Because of the integral part water distribution networks play in modem civilisation, a significant amount of research has gone into the economical design of new water distribution networks, and the strengthening and expansion of existing ones.

There are a great number of optimisation techniques at the disposal of engineers when designing and optimising water distribution networks. Eusuff and Lansey [70] state that 'virtually every optimization method has been applied to the water distribution network optimization problem' although industry still relies heavily on the experience of engineers and their traditional trial and error approach.

Although to date there has been a substantial amount of research into the use of genetic algorithms for the optimisation of water distribution networks, the genetic algorithm has for the most part failed to be adopted by practising water system engineers. To understand why this is, it is first necessary to understand the traditional approach still utilised by the water system engineers of today. The traditional design approach consists of two stages: (1) Define the configuration of all elements present in the network; and (2) analyse the hydraulic competence of the solution, making modifications where necessary. This trial and error approach through repetitive analysis usually results in an adequate solution, however this method can be incredibly time consuming and becomes highly dependent on the ability and experience of the design engineer. Also this methodical design approach does not address the economic aspects of the network in the core design process.

Goulter [71] discuses some possible reasons for the lack of optimisation approaches in use by practitioners, he states “although the subject has been studied by researchers, there is little or no acceptance of optimization into

engineering practice". In his paper, Goulter suggests four key reasons for the lack of practical implementation of optimisation techniques.

Optimisation models do not derive solutions to practical problems. This statement holds true to an extent as the majority of works have only assessed the performance of optimisation techniques using small or medium sized benchmark networks and very few have tested these methods on large-scale real networks [72]. However more recently there have been a number of studies undertaken that have tackled large-scale real water distribution networks. Reca et al. [72] applies several meta-heuristic techniques to a large irrigation water distribution network which was proposed in a previous work [73] with the view to assessing their performance in a practical application. The study assesses several different search methods including, genetic algorithms, simulated annealing and tabu search. The research concludes that these techniques, although performing well for medium sized networks, require a high amount of solution evaluations and subsequently greater computation time to accurately achieve a viable solution. Also, it has yet to be seen whether a single optimisation model would provide an optimal solution for all types of design problems stated by Walski [74].

Solutions obtained by optimisation techniques offer no improvements over established methods and professional judgement. This statement does not hold true. Although an experienced design engineer may obtain by traditional methods, a design comparable in cost to that obtained through optimisation techniques [69], it has been observed in practice [75], having met the network's desired level of performance, design through optimisation methods can result in considerable network cost savings, sometimes as high as 20 to 30%.

Optimisation techniques are too difficult to use in practice. This is, for the most part true. Optimisation approaches for the design of water distribution networks mainly reside in the academic domain, confined to the research paper [69]. The scope of the algorithms that are developed only stretches as far as the immediate goals of the research. Also computer programs applicable to water distribution network design which employ optimisation techniques are not readily available for general use within the commercial sector.

Practitioners are not comfortable with the overall optimisation approach. This remains mainly true. Many of the established engineers in the field of water distribution network design have not had the opportunity to study formal optimisation techniques [71]. Also, unless systems analysis and operations research become a greater part of undergraduate and postgraduate studies, the graduate engineer will feel reluctant to utilise optimisation approaches when designing water distribution networks.

The last two points highlight a substantial gap between the academic and commercial sectors which will need to be addressed before the benefits of optimisation techniques, including genetic algorithms can be obtained in the commercial sector. It has been observed that the effort required to use current optimisation models is a large factor that can prevent the practitioner from using the models on a frequent basis thus failing to learn the limitations of the approach. Goulter [71] states that “Improvements in the user interfaces will accomplish a great deal toward overcoming this problem.”

Having said this, some organisations have started developing applications to solve complex optimisation problems with more emphasis being placed on the user interface. An example of one of these packages is GANetXL developed at the University of Exeter which uses multi-objective evolutionary algorithms to solve complex optimisation problems. The program contains a user friendly wizard which guides the user through the entire configuration process.

2.4.1 Application of Genetic Algorithms to Water Distribution Networks

It is only in the last two decades that research into the application of genetic algorithms to the design of water distribution networks has been undertaken. Some of the first work in this field was done by A. R. Simpson, G. C. Dandy and L. J. Murphy. In their paper [76], entitled Genetic Algorithms Compared to Other Techniques for Pipe Optimization, a relatively basic genetic algorithm is applied to a simple pipe network previously used by Gessler [77] to assess methods of enumeration when optimising pipe networks. The aim of the paper [76] was to compare a genetic algorithm approach with complete enumeration and non-linear programming techniques. The Gessler [77] pipe network was chosen as it displayed some “interesting features”; these included

the option to replace specific pipes with new pipes from a range of discrete diameters. Also, specific restrictions applied to certain pipes that could only be cleaned, duplicated or left alone. In addition to these restrictions, the viability of the network would be assessed using three different demand patterns.

In the Gessler [77] network problem, eight decision variables are present in the form of either new pipes or existing pipes that can be duplicated, cleaned of left in their initial state. This information was coded by Simpson's [76] genetic algorithm in a 24-bit binary string (chromosome) where each decision variable corresponded to a 3-bit binary sub-string representing eight possible alternatives as either new pipes, duplicated pipes or the cleaning of existing pipes. The initial population of solutions was generated using a commonly used pseudorandom number generator algorithm called a linear congruential generator.

The fitness function for a solution was defined as the inverse of the total network cost, which consisted of the network cost and any penalty cost incurred. The network cost was calculated by decoding each substring into the corresponding pipe size therefore enabling the computation of material and construction costs for each decision variable. A penalty cost is incurred if a network solution does not meet the minimum pressure constraints; in the event of this the maximum pressure deficit in the network is multiplied by a penalty factor which in this case is a predefined figure of $70,000.

Due to the relatively small amount of decision variables and decision variable choices present in the Gessler problem, it was feasible to use complete enumeration to evaluate every possible set of pipe configurations for the network problem. Complete enumeration of this problem took a total of 81 central processing unit (CPU) hours on a SUN 4/280 (16.67 MHz CPU). However, as a problem's number of decision variables is increased the number of solutions present in that problem increases exponentially. Therefore, problems consisting of slightly more decision variables than the Gessler problem become impractical to solve using complete enumeration. For example, Savic & Walters [12] consider a pipe network consisting of 20 pipes and a set of 10 possible discrete pipe diameters. The search space for this problem is therefore equal to 10^{20} different design combinations. Even if

1,000,000 designs could be evaluated by a central processing unit (CPU) every second, it would take over 3,000,000 years to evaluate every possible solution.

Simpson's genetic algorithm performed well, identifying one of the two global optimum solutions on a number of runs, which was verified by the complete enumeration of the solution space. Also the genetic algorithm obtained multiple solutions which lay close to the optimal solution. The non-linear programming technique also identified the global optima. However, this technique produces a single continuous solution which has to be rounding up or down to the nearest discrete pipe size. This leads to the need for additional computer runs to ensure the rounded solution satisfies the pressure constraints. For small optimisation problems such as this, the rounding process can be completed manually however, for larger systems the rounding process soon becomes a secondary optimisation problem [76].

Because the genetic algorithm approach produces a whole class of solutions close to the optimum, it may be preferable that one of these alternate solutions be selected instead of the optimal due to unquantifiable factors. This is a key advantage genetic algorithms have over the non-linear programming technique which only produces a single solution.

Although only a basic network model was used, this paper illustrated that genetic algorithms had the potential to become a powerful asset when designing and modifying water distribution networks.

2.4.1.1 GANET

In 1997 D. A. Savic and G. A. Walters presented the development of their genetic algorithm program called GANET for the least-cost design of water distribution networks. In their paper [12], Genetic Algorithms for Least-Cost Design of Water Distribution Networks**,** a comparative study is performed using GANET to provide insight into the expected performance of solutions identified in previous studies. They make the observation that water distribution networks are commonly viewed as a least-cost optimisation problem with pipe diameters constituting the only decision variables. Network layout, connectivity and node head constraints are fixed and considered known, hence simplifying the problem. Other objectives such as reliability, redundancy and water quality are

excluded from optimisation design models as they are difficult to quantify, thus resulting in the formulation of single, least-cost objective problems. However, it has been observed that networks designed on a purely cost effective basis and for a single loading condition often result in a branched network [78]. In most cases, it is required that loops be preserved due to redundancy considerations. This is commonly achieved by the introduction of a minimum pipe diameter constraint which ensures that any loops present in a network are retained in the final solution [12].

Similar to previous genetic algorithms developed for the optimisation of water distribution networks, GANET coded its solutions using binary strings consisting of decision variables represented by substrings. This method is simple and effective however; an issue arises when the number of possible substring combinations is greater than the number of available decision variable values. If this occurs it results in redundant substring values which would void the solution. Savic & Walters [12] solve this problem using fixed remapping; a procedure that maps a particular redundant substring value to a specific valid pipe diameter. This method handles the problem; however it could introduce bias into the system, as this approach would increase the probability that specific mapped values are selected.

Instead of using a simple binary numeral system for the decoding of the bit strings, Savic & Walters [12] opted for a reflected binary code, or Gray code interpretation. The defining characteristic of Gray code is that adjacent or successive integers differ by only a single bit [36]. It is assumed that good solutions tend to lie close together in the search space [12], therefore the “adjacency property” [36] of Gray code allows neighbouring solutions to be acquired more easily through mutation than they would with binary-coded strings.

The solution fitness was calculated using an evaluation function which determined the solution cost by summing the cost of all the pipes in the network. To evaluate the feasibility of each solution, the network flows and pressure heads were simulated using the solver EPANET, an open-source tool used for modelling dynamic water distribution. Similar to Simpson’s work [76], GANET did not dismiss infeasible solutions but rather allowed them to join the

population and help guide the search. However, the point was made that in allowing an infeasible solution into the population, that solution could achieve a fitness greater than that of a feasible solution. This was prevented by adding a graded penalty function when evaluating the fitness of a solution. Also a penalty multiplier was incorporated into the evaluation function which allowed a gradual increase in the penalty term. This was achieved by setting the penalty multiplier to a value which normalised the nominal penalty values to the same scale as the basic cost of the network.

Rank selection was used for the selection of parents for recombination as to avoid problems with scaling and premature convergence which can arise with traditional proportional fitness schemes such as roulette wheel selection. After experimentation, the uniform crossover [79] operator was selected which produces a single offspring from two parents. Random jump mutation was employed with a relatively low mutation rate which on average only affected 1 in the 32 genes of the chromosome.

GANET performed well and produced good designs without unnecessary restrictions that are required in some other optimisation techniques and highlighted that genetic algorithms are particularly suited to the problem of least-cost design of water distribution networks.

2.4.2 Evolutionary Algorithms and Water Resources

As stated previously, Genetic algorithms and Evolutionary Strategies were developed as individual algorithm classes. However, in 1991 a joint conference between the two communities sought to collaborate under the term “Genetic and Evolutionary Computation.” Early work in the fields of Genetic Algorithms and Evolution Strategies were fashioned by their corresponding representation; genetic algorithm solutions were coded as binary strings where real-valued continuous variables were employed for the representation of Evolutionary Strategy solutions [40]. Early work in the field of water system optimisation [76][80][12] focused mainly on the use of the binary framework laid-out by Goldberg [36]. However, later works started to code variable decisions as strings of integers corresponding to design decisions. Halhal et al. [81] employed this integer coding approach as it suited their problem formulation

and proved to be more flexible than standard binary coding. Another example of real-valued gene coding is seen in the research conducted by Yoon & Shoemaker [42] in ground-water bioremediation. The research found that the real-integer genetic algorithm with directive recombination and screened replacement operators was computationally superior to binary coded genetic algorithms; finding much better solutions to groundwater bioremediation problems in significantly less computation time.

2.4.2.1 Multi-Objective Evolutionary Algorithms and Water Resources

Most real-life problems can be classed as being multi objective, where a solution is always a trade-off between the applications conflicting objectives. Xu et al. [82] states that “In practise, the optimal design of a water distribution network is a complex multiple objective process involving trade-offs between the cost of the network and its reliability”. Having completed the multi objective optimisation process, the resulting set of solutions that are superior to all other solutions in at least one objective are deemed to be Pareto-optimal solutions [83]. When plotted according to their objective values, these solutions make up what is known as the Pareto-optimal front from which the user is able to select a suitable solution. Some of the first work in multi objective evolutionary algorithms was performed in the field of water resources [84]. Subsequent research [81][85][86] in this area has proven to return excellent results. More recently, a larger number of researchers have focused their attention on multi objective evolutionary algorithms in many objective applications where three or more objectives are simultaneously optimised [87][85][88]. One of the core strengths of multi objective evolutionary algorithms is their capacity to quickly approximate the genuine Pareto surface even if it is not exactly quantified, which can prove to be adequate in the presence of computational constraints [40]. It is worth noting that multi objective evolutionary algorithms have been found to display quadratic computational complexity when applied to water resources problems [88]. This implies that any increase in the number of decision variables present in an application will result in a drastic increase in the function evaluations required to solve the problem. This problem was addressed by Kollat & Reed [88] by archiving solutions to decrease the computational complexities of the multi objective evolutionary algorithm to achieve a near linear state.

2.5 Constraint Handling in Evolutionary Algorithms

In their basic form, EAs are unconstrained optimisation procedures. However, many problems have constraints imposed upon them, especially in real-world optimisation problems. A common approach to dealing with constrained optimisation problems is to incorporate the constraints into the fitness function of the EA by adding a penalty function to the fitness function, where the value obtained from the penalty function represents the solution's distance from feasibility.

2.5.1 Penalty Functions

A frequently used approach is the static penalty [89], where the penalty factors remain constant throughout the evolutionary process. Another approach is the use of a dynamic penalty where the penalty function is varied over time, commonly tightening the constraints as the EA's population develops. The notion of allowing an EA to explore the search space unimpeded before increasing the focus of the search, and therefore potentially improving the scope of the search, has led some researchers to argue that dynamic penalties perform better than a static penalty approach. However, it has been found that deriving an effective dynamic penalty function is as difficult to achieve as producing good penalty factors for static functions [90]. Penalty annealing [91] is an approach inspired by the notion of simulated annealing [25] where the penalty for violating constrains is gradually increased over time such that by the end of the optimisation infeasible solutions are penalised greatly. Although efficient, this process is very sensitive to its parameter values and also requires the user to provide a feasible starting point. Other notable approaches for handling problem constraints include the death penalty [92], segregated GA [93] and co-evolutionary penalties [94].

2.5.2 Special Representations

A number of researchers have developed methods to approach particularly complex problems using special representation instead of more traditional representation schemes. These representations are generally used to simplify the search space and promote solution feasibility through specially modified operators [95]. Examples of the use of special representation schemes

and operators include: GENOCOP [96], random keys [97][98] and constraint consistency [99]. A further method is to use an indirect representation where the genes do not code for variables in the problem directly, but via a heuristic that determines the phenotype given the genotype developed by the algorithm. These approaches have been shown to work well in timetabling problems [100] but the relationship between the genotype and phenotype is more complex leading to a more multimodal fitness landscape.

2.5.3 Repair Algorithms

Another approach when handling the constraints of a problem is to employ a repair algorithm. The repair algorithm has proven a popular choice for many combinatorial optimisation problems [101][102] as it is often relatively easy to "repair" an infeasible solution through the iterative modification of individual decision variables. When a solution can be transformed from infeasible to feasible at a low computational cost, repair algorithms have proven to be effective. However, it is not always possible to repair an infeasible solution at an acceptable computational cost and in some cases the algorithm can harm the evolutionary process by introducing a strong bias in the search [103].

2.6 Knowledge Guided Search

There have been a number of approaches in the literature which use knowledge of the problem or search space to aid the search of an EA. One such method is Guided Local Search (GLS) [104], a metaheuristic technique bearing similarities to tabu search and simulated annealing. GLS has displayed good performance on a number of combinatorial optimization problems [105][106][107][108] as it helps prevent the search from becoming stuck in a local minima. GLS functions by penalising certain solution features that it deems would not occur in near optimal solutions through the use of weighted penalties. Another approach is Guided Mutation (GM) [109], an offspring generating operator for EAs which is considered a combination between an EA's standard mutation operator and the offspring generating method of an Estimation of Distribution Algorithm [110]. GM works on the notion that good solutions have a similar structure and new offspring should be generated close to the good solutions already found during the search.

The use of Evolutionary Algorithms (EAs) by researchers in the field of hydroinformatics for the design and optimisation of water systems has grown over the past two decades and with the emergent maturity of the field has come an increased focus on real-world application. These real-world water distribution problems present a much greater challenge due to their drastically increased size and complexity. It has been shown that discovery of the globally optimal Pareto fronts for large multi-objective water distribution network problems is particularly challenging [13].

In the case of the Battle of the Water Networks II [13] a number of participant researchers utilised domain knowledge and heuristic information to either decrease the size of the search space or locate favourable areas of the solution space to initialise the search [13]. These knowledge guided techniques are generally aimed at achieving near-optimal solutions with the use of limited computational resources, rather than attempting to find the globally optimal Pareto front of a complex problem [111][59][112][113][114]. An important consideration when applying EAs to real-world problems is the large computational overhead incurred when solving complex hydraulic models [115]. It becomes apparent that there is a need for approaches that are capable of finding near-optimal solutions within the constraints of available computational resources and in doing so will aid in the effective application of EAs in the practical domain [115]. Tolson et al. [116] have shown that with limited computational resources high quality solutions can be achieved if a significant amount of engineering judgement is used. Marchi et al. [13] suggests there is always going to be a trade-off between the engineering experience and computational resources needed to solve complex water distribution network problems and that engineering judgement can never be completely avoided. This notion expands beyond hydroinformatics to a wider set of problem domains where domain knowledge has been shown to be an important factor when tackling real-world problems. Some examples of this can be found in the wider field of engineering, including aeronautical [117] and mechanical [118] design.

As previously stated, there is a growing interest in the use of domain specific knowledge in the design of water distribution networks. Keedwell and Khu [119] developed a hybrid cellular automaton and genetic approach which included a hydraulically based heuristic used in the formulation of initial EA

populations, a method which was found to be highly effective when tested on a set of large scale networks. The hydraulically based heuristic was based on the premise that the diameter of a pipe connected to a demand node in pressure deficit can be expanded to increase pressure and the diameter of a pipe connected to a node in pressure excess can be decreased to improve network cost. Zheng et al. [120] used knowledge of pipe network topology and a nonlinear programming technique to identify promising areas of the solution space, subsequently seeding the initial population of a differential evolution (DE) algorithm. Another initialisation method was proposed by Kang and Lansey [121] which used pipe flow velocity thresholds to form a set of initial solutions, Bi et al. [122] then adapted this idea and added a heuristic based on the notion that pipe diameters generally reduce with the distance from the source [123].

2.7 Interactive Evolution

Interactive Evolution (IE) aims to incorporate human knowledge through interaction with an EA which requires the input from a user during an algorithm's search. User interaction is commonly used to assess a solution's fitness; however the user can also be involved during the variation and selection phases of the algorithm. A common issue when applying an EA to a problem, especially in a real-world setting, is there are often non-explicit conditions that are very difficult to define. Various design approaches require the engineer to make subjective decisions based on human intuition, such as the ability to judge a design's aesthetic qualities. The fitness criterion in cases such as these cannot be explicitly formulated and often require case-by-case comparison to effectively assess a solution. The interaction of a human user can also be employed to more effectively guide an algorithm's search of the solution space with the view to speed up convergence and prevent local optima trappings.

2.7.1 Water resources design and management

Water resources design and management problems are complex to solve; not only from a mathematical perspective, but also from political, sociological, and other subjective viewpoints. The majority of research in the field of water resources concentrates on the improvement of simulation models and their

incorporation with optimisation techniques such as evolutionary algorithms. The problem lies in real life cases where the optimisation technique employed returns a mathematically optimal solution, however the solution may become infeasible when considering subjective preferences [40]. Recently, researchers in the field of water resources have developed methods for the calibration of models through the use of interactive evolution which enables the incorporation of unmodeled objectives in the search procedure [124]. The field of interactive evolution is a rapidly growing area of research; with the aim to utilise the subjective responses from human users to guide the search of evolutionary algorithms [125]. Singh et al. [124] used an elitist non-dominated sorting genetic algorithm (NSGA-II) [63] and human responses to find optimal solutions for groundwater problems which were both mathematically optimal and feasible. This was achieved through the consideration of human responses as one of the multiple criteria for the computation of the solution fitness. Although the interaction element of the process was simple (solution ranking) the results of the study were successful in generating superior solutions than non-interactive optimisation runs of NSGA-II.

2.8 Conclusion

This chapter introduced the concepts at the core of mathematical optimisation to provide a basis for further exploration of heuristic techniques for the solving of optimisation problems. The development of heuristic based optimisation methods and the move in the literature to more complex meta-heuristic techniques such as Evolutionary Algorithms (EAs) was presented. Unlike the heuristic methods offered in the literature, meta-heuristics have the ability to be applied to wide range of problems with little to no modification; offering a truly global search process. One prominent EA discussed in this chapter was the Genetic Algorithm (GA), a technique inspired by natural selection and genetics. The GA has been applied to numerous optimisation problems from a wide range of fields with good effect. Although the basic structure of a GA is generally agreed upon, the various components of the process have been altered throughout the literature and provide a researcher with a wide range of choices when formulating a GA. The chapter provides an in-depth look at the various components of a GA, including the selection methods, evolutionary operators, parameters and solution evaluation.

A large proportion of real-world problems have multiple criteria, often requiring a solution to compromise between different objectives. With problems such as these, combining all objectives into a single function can result in a situation where two solutions with the same fitness value are drastically different from one another in terms of their criteria. This resulted in the development of multi-objective optimisation techniques where two or more objectives are optimised simultaneously which enabled the greater exploration of the search space and trade-offs between solutions. This chapter presented the traditional formulation of the multi-objective optimisation problem and provided a summary of seminal works in the field of multi-objective EAs.

Evolutionary Algorithms have been applied to a large and varied set of optimisation problems spanning a substantial number of academic and commercial fields. One such field is water distribution network design. In this chapter the design and optimisation of water distribution networks was discussed along with the potential reasons for low uptake of optimisation approaches in the commercial sector and suggestions on how to make EAs more accessible to practising engineers. Following this a detailed review of the application of single and multi-objective EAs to the water distribution network design problem was conducted.

The majority of real-world optimisation problems not only consist of primary objectives but also constraints that ensure a solution is feasible. In this chapter various methods of constraint handling within an EA are detailed including the commonly utilised penalty function, special representations, repair algorithms and guided search. Finally a review of interactive evolution was presented, focusing on the application of IE in the field of water systems design and management.

The growing body of research in hydroinformatics which focuses on the use of specific domain knowledge and heuristic information to boost EA performance has produced many promising results, often outperforming standard methods on a range of problems. Unlike other domains however, the majority of techniques presented in the hydroinformatics literature tend to focus on the use of specific domain knowledge for the initialisation of starting populations and not the operators such as crossover and mutation. Therefore it

is interesting to explore the impact that integrating engineering knowledge into the operators of an evolutionary algorithm would have on performance and therefore filling this gap in the body of research. Another observation is that the majority of hydroinformatic knowledge based EAs discussed in this chapter have only been applied to single-objective water distribution network problems with the exception of Keedwell and Khu [119] and Bi et al. [126]. Therefore exploring the impact knowledge based operators have on a multi-objective evolutionary algorithm adds to the body of knowledge. Also, due to the relative infancy of this area of research within hydroinformatics there is also a distinct lack of extensive studies which explore the impact that incorporating engineering knowledge into an EA has on the parameter robustness of an evolutionary algorithm, something that has been studied at great length in the wider field of computer science. Doing so provides a greater insight into the effect knowledge guided search methods have on an algorithm's sensitivity to parameter value change.

Chapter 3: Genetic Algorithm for Water Distribution Network Design and Heuristic Development

The focus of the first section of this chapter is the formulation of single and multi-objective genetic algorithms for the effective optimization of water distribution networks. This section provides a detailed background of the formulation of the algorithms that will later provide a vehicle for the novel component of this work; the application of engineering inspired heuristics to the search procedures.

Section 3.4 of this chapter describes the development of two heuristics that are based on the practices of engineers when designing real-world water distribution systems. At the heart of this section lies the identification of key principles that drive the traditional design process employed by water systems engineers. These principles are then broken down into heuristic methods that can be applied to the optimization procedures with the view of improving search performance and improving engineering optimality.

3.1 Single Objective Genetic Algorithm for Least Cost Water Distribution Network Design

An approach to the optimal design of water distribution networks [127] is presented in this section with a focus on the single-objective least cost design of water distribution networks through the use of a Steady State Genetic Algorithm (SSGA). Firstly the formulation of the single-objective water distribution network design problem is presented followed by an explanation of the steady-state genetic algorithm and its application to the problem.

3.1.1 Water Distribution Network Design Problem

Water distribution network (WDN) design is a complex non-linear optimisation problem, commonly involving a large number of different network components and hydraulic constraints. Due to the inherent complexity of WDN design, a simplified formulation of the problem is commonly employed when applied to optimisation techniques. This method is commonly comprised of the allocation of a diameter to each pipe in a given network layout, with the

objective of minimising cost whilst satisfying pressure constraints at the nodes and in some cases velocity constrains in the pipes [127]. In this simplified version, design considerations such as water quality and network reliability are not included in the formulation of the problem. This method provides the designer with a base from which to solve the overall problem and allows the comparison of new optimisation techniques with the large amount of literature that employs this technique of problem formulation.

The optimal design of a water distribution network is presented here using the following mathematical statement. The objective function is defined as the total cost of the network with regard to pipe length and diameter:

$$f(D_1, \dots, D_n) = \sum_{i=1}^{N} c(D_i, L_i) \qquad (3)$$

where $c(D_i, L_i)$ = cost of pipe i with diameter D_i and length L_i with N = number of pipes in the network. This function is to be minimised whilst satisfying the following constraints. For each junction (excluding the source) the following continuity constraint has to be satisfied:

$$\sum Q_{in} - \sum Q_{out} = Q_e \qquad (4)$$

where Q_{in} = inflow to the junction, Q_{out} = outflow from the junction and Q_e = external flow or junction demand which in this case is always positive. Head loss h_f for a specific pipe i is calculated using the following equation:

$$h_f = \omega \frac{L_i}{C_i^a D_i^b} Q_i^a \qquad (5)$$

where C_i = Hazen–Williams roughness coefficient, Q_i = flow and a, b, and ω are parameters of the equations.

The minimum head constraint for each junction in the network is as follows:

$$H_j \geq H_j^{min}; j = 1, \dots, M \qquad (6)$$

where H_j = head at junction j, H_j^{min} = minimum head requirement at junction j and M = total number of junctions present in the network.

The maximum flow velocity for each pipe in the network is as follows:

$$V_i \leq V_i^{max}; i = 1, \ldots, N \qquad (7)$$

where V_i = flow velocity in pipe i, V_i^{max} = maximum allowable flow velocity in pipe i and N = total number of pipes present in the network.

In the case of this single-objective formulation of the WDN design problem the optimisation is exclusively concerned with the selection of pipe diameters. Water supply system pipes are generally manufactured in a set of discrete sized diameters, making the problem relatively easy to encode for an algorithm such as a GA. Each individual problem has a set of available pipe diameters which can be selected for each decision pipe in the network. These decisions are encoded as a binary bit sub-string with a length dictated by the number of pipe sizes available, the substrings are then concatenated to form the chromosome to represent the entire solution. Integer encoding can also be utilised, where a string of integers represent the solution. In this case each string position represents a decision pipe and each available pipe diameter is paired with a unique integer value.

When evaluating the hydraulic performance of a network it is common practice to utilise a hydraulic solver which models the performance of the water distribution system. Such solvers ensure that the majority of the hydraulic constrains of a network are met by calculating junction inflow and outflow (2) and headloss (3) along with other hydraulic considerations. Although the majority of hydraulic constrains are satisfied by the hydraulic solver, the optimisation method still needs to deal with problem constraints such as junction head and pipe velocity constrains. In the case of Evolutionary Algorithms (EAs), a common approach to dealing with constrained optimisation problems is to incorporate the constraints into the fitness function by adding a penalty function to the fitness function, where the value obtained from the penalty function represents the solution's distance from feasibility. A frequently used approach is the static penalty [89], where the penalty factors remain constant throughout the evolutionary process.

3.1.2 Steady-state Genetic Algorithm

One of the first examples of a Steady-state Genetic Algorithm (SSGA) was Genitor [128] [129], although the term Steady-state Genetic Algorithm was introduced by Syswerda [79]. Steady-state Genetic Algorithms tend to exhibit increased variance compared with standard generational GAs with regard to hyperplane sampling behavior [130], hence are more susceptible to genetic drift and sampling error. The benefit that SSGAs have over generational GAs is that it naturally enforces elitism [131] which ensures the best performing solutions are retained in the population, which many researchers have found results in a more aggressive search of the solution space often increasing performance.

The core differences between a standard SSGA and a standard generational GA lie within the reproduction, replacement and fitness allocation procedures. Firstly, in the standard formulation of a SSGA two parents are selected for reproduction and produce one or two offspring which are placed immediately back into the population. Instead of directly replacing its parents, the offspring replace the worst performing individual in the current population according to the active replacement criteria, which can be based on solution fitness, age etc. In a study of selection schemes by Goldberg and Deb [132] it is shown that replacing the worst individual of the population increases the selection pressure more than that of purely random replacement. Selection pressure In some SSGAs fitness is assigned according to rank instead of assigning a proportionate fitness value. Ranking solutions in the population has shown in some cases to sustain a more constant selection pressure during the algorithm's search.

3.1.3 Formulation of a Steady-state Genetic Algorithm for the Least-cost WDN Design Problem

This section provides a detailed description of a steady-state genetic algorithm (SSGA) and its application to the least-cost water distribution network design problem. This formulation of a SSGA will be used as a platform on which the heuristics (discussed later in this chapter) will be applied. The SSGA will also be used to provide a benchmark when assessing the performance of the heuristic modified algorithms.

Figure 3-1 shows a flow chart of the steady-state genetic algorithm. At initialization *N* individual solutions are randomly created to form the initial population. Each individual in the starting population is assigned a fitness value by evaluating the individual's solution using the problem's objective function. In this formulation of the SSGA a tournament selection procedure is employed to choose which individuals will be selected to reproduce. A tournament size of *tN* individuals are chosen at random from the population for each of the two resultant parents, where *t* dictates the number of individuals selected with a value normally between 0.02 & 0.1 resulting in 2% - 10% of the population being selected for each tournament. As with most selection procedures, the fitter the individual, the more likely it is to be selected for reproduction due to the t individuals undergoing a tournament with the best individual being selected as parent solutions for the next iteration.

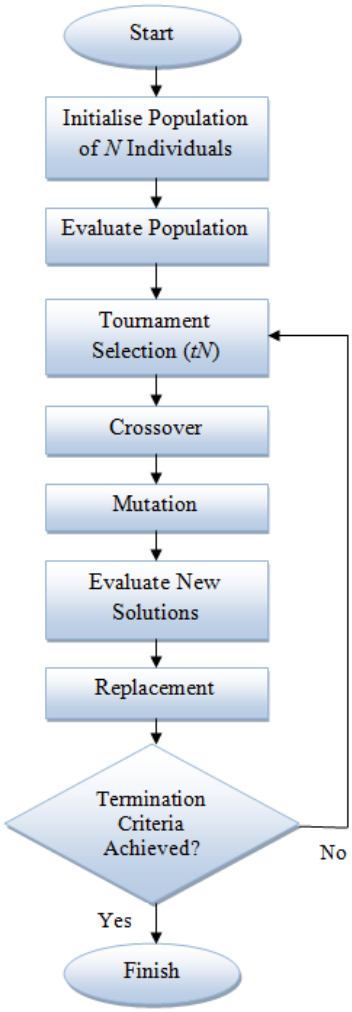


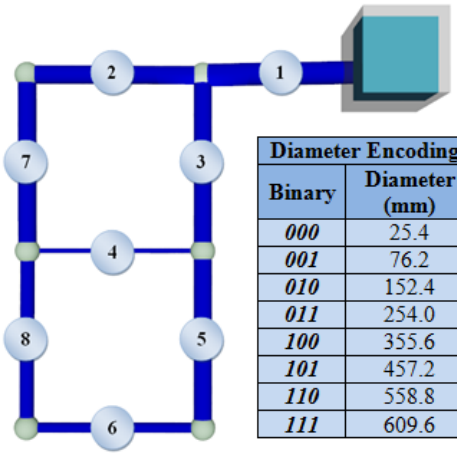
Figure 3-1. Flow chart representation for the steady-state genetic algorithm

Following the tournament selection process, the two individuals chosen for reproduction are passed to the crossover operator which combines the parent solutions to form, in this case, two child solutions. This is achieved using a simple single point procedure where the operator randomly chooses a locus (position on the chromosome) and exchanges the sub-sequences before and after the locus between the two chromosomes resulting in the formation of two offspring. The newly formed child solutions are then passed to the mutation operator. A common approach to the mutation of chromosomes in binary encoded genetic algorithms is to use a simple bitwise process where the operator randomly flips a number of bits in the chromosome. However, in this formulation of the SSGA, due to the implementation of the heuristic based mutation operators described later in this work, it was necessary, for direct comparison purposes for the operator to mutate at the decision variable (pipe) level. For example, if a pipe is selected for mutation, any number of the bits in the relevant substring can be altered. Following mutation, the newly formed individuals are evaluated using the objective function described in section 3.1.1 and assigned a fitness value. The individuals are then subjected to the replacement operator which utilises a conditional worst solution strategy where each candidate solution is compared with the current worst solution in the current population. If the candidate solution has a better fitness value than the worst solution in the population then that individual is replaced with the candidate, otherwise the new solution is discarded. At this point if the algorithm termination criterion has not been met the method returns to the selection phase of the process.

3.1.3.1 Solution Coding

As stated previously, pipes for water distribution systems are generally produced in discreet sized diameters thus making it a relatively trivial task to encode for a GA. Although methods exist which treat pipes as continuous variables, this approach has a limiting factor on the size of problem which can be attempted [133] [134]. The use of genetic algorithms for the optimization of these types of problems has proven quite popular, partly due to the way genetic algorithms encode discrete variables, as commercially available pipes are only produced in discrete sizes. No manufacturer is going to specially produce pipes

with oddly specific diameters to facilitate an optimal solution produced by an algorithm.



Diameter Encoding	
Binary	Diameter (mm)
000	25.4
001	76.2
010	152.4
011	254.0
100	355.6
101	457.2
110	558.8
111	609.6

Pipe	1	2	3	4	5	6	7	8
Chromosome	*1 0 1*	*1 0 0*	*1 0 0*	*0 0 0*	*1 0 0*	*0 1 0*	*1 0 0*	*0 1 1*
Diameter	457.2	355.6	355.6	25.4	355.6	152.4	355.6	254.0

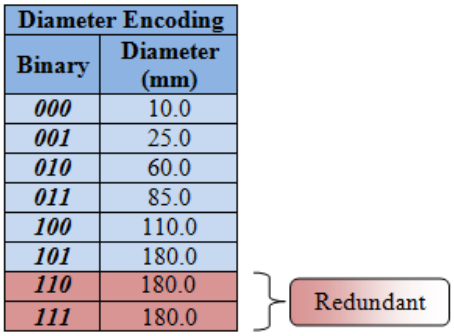
Figure 3-2. Water network chromosome decoding example

This formulation of the steady-state genetic algorithm employs a standard binary encoding method where each pipe is represented by a binary substring which translates to a specific diameter. Figure 3-2 shows an example solution for the Two Loop [127] problem which has been slightly modified for the purpose of this example by reducing the number of available pipe diameters from 14 to 8. In this example there are 8 available discrete diameters to choose from and 8 decision variables (pipes); therefore a 3-bit substring encoding has been utilized ($2^3 = 8$) resulting in a 24-bit string representation of the chromosome. One drawback of binary encoding is there are often instances where the number of decisions available does not exactly equal the capacity of the *n*-bit substring representation. Therefore in the event of equation $log_2 x = \mathbb{N}_1$ (8) not being satisfied, the number of discrete pipe diameters (x) needs to be increased until the expression is satisfied, but by doing so introduces redundant decisions into the encoding.

$$log_2 x = \mathbb{N}_1 \qquad (8)$$

In this composition of the SSGA the redundant binary encodings are allocated the highest available pipe diameter, promoting hydraulic feasibility.

Figure 3-3 shows an encoding for 6 pipe diameters; however the necessary 3-bit binary representation leaves 2 redundant allocations. In this case the largest pipe diameter (180mm) is duplicated at ***110*** and ***111***.



Diameter Encoding	
Binary	**Diameter (mm)**
000	10.0
001	25.0
010	60.0
011	85.0
100	110.0
101	180.0
110	180.0
111	180.0

Figure 3-3. Diameter encoding redundancy example

Although popular in the water systems optimisation literature [12], it was decided that the use of Gray coding would be redundant due to the way the mutation operator (see section 3.1.3.4) handles chromosome data elements as complete entities, thus a standard binary implementation was used. Even though the use of such a mutation operator greatly negates the use of a binary encoding implementation, it was decided that a real-valued implementation would undermine the effect the crossover operator‘s ability to insert additional variance into the search.

3.1.3.2 Solution Evaluation - Fitness

Utilizing the mathematical formulation of the water distribution network design problem presented in section 3.1.1 the solutions produced by the SSGA are determined. To evaluate the hydraulic behavior of the solution network it is necessary to employ a water distribution pipe system modeler, in this case, EPANET 2[1]. The EPANET 2 open-source API is a computational engine developed originally by the United States Environmental Protection Agency (EPA) which enables the modeling of water distribution piping systems. The EPANET API allows for the extended period simulation of hydraulic and quality behavior within pressurized pipe networks.

[1] United States Environmental Protection Agency: *www.epa.gov/water-research/epanet*

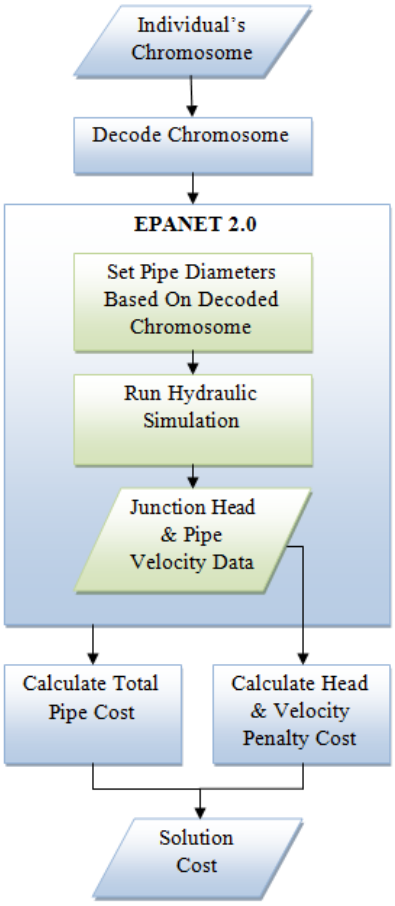


Figure 3-4. Flow chart representation of the water system cost calculation

Figure 3-4 shows a representation of the process to obtain the total cost of a solution which includes both infrastructure and hydraulic performance considerations. Firstly the individual's chromosome is decoded to form a solution to the problem, as described in section 3.1.3.1 and the pipe diameter information is passed to EPANET 2. The pipe diameters are then set within the EPANET network model and the hydraulic simulation of the system is run. In this implementation of the SSGA only a single period simulation is required for each solution as the problems tackled by this work (see section 3.3) do not necessitate extended period simulation which is more usually used for water quality and operational optimisation. Following the simulation of the solution network, the relevant junction head and pipe velocity values are extracted from the model and passed to the penalty function which calculates the cost of any junction head or pipe velocity constraint violations. In this implementation of the SSGA a simple static penalty function is employed to assign a cost to any

hydraulic constraint violations. Equation (9) shows the formula for calculating the hydraulic head deficit penalty cost (C_{ph}):

$$C_{ph} = \sum_{i=1}^{N^j}(P_h H_i^d) \qquad (9)$$

Where N^j is the number of junctions present in the network, P_h is the static penalty cost per unit of junction head deficit and H_i^d is the head deficit at junction *i*. A junction is said to be in deficit when the head at a junction (H_i) is less than the minimum head requirement at that junction (H_i^{min}).

$$(H_i < H_i^{min} \Rightarrow H_i^d = H_i^{min} - H_i) \qquad (10)$$

When the junction head (H_i) is greater than H_i^{min} the junction is said to be in excess. In this formulation of the static penalty function a junction in excess is given a value of zero head deficit.

$$(H_i \geq H_i^{min} \Rightarrow H_i^d = 0) \qquad (11)$$

The formula for calculating the excess velocity penalty cost ($\mathrm{C_{pv}}$) is shown in equation (12), employing a similar approach to that of the head deficit penalty calculation. The method is applied to each of the N^p pipes in the network

$$C_{pv} = \sum_{i=1}^{N^p}(P_v V_i^x) \qquad (12)$$

where $\mathrm{V_i^x}$ is the excess velocity in pipe *i* and P_v is the penalty cost for each unit of excess velocity present in a pipe. When the velocity in a pipe ($\mathrm{V_i}$) exceeds the maximum allowable velocity for that pipe ($\mathrm{V_i^{max}}$), the excess velocity is calculated as such:

$$(V_i > V_i^{max} \Rightarrow V_i^x = V_i - V_i^{max}) \qquad (13)$$

In the case of when the velocity constraints are satisfied in a pipe the following expression is used:

$$(V_i \leq V_i^{max} \Rightarrow V_i^x = 0) \qquad (14)$$

The presence of penalty factors within the algorithm's objective function adds an increased challenge when tuning the parameters of the algorithm to

achieve near optimal performance. The issue of algorithm parameter tuning is covered in more detail accompanied by extensive experimentation in section 0.

$$C_t = C_n + C_{ph} + C_{pv} \qquad (15)$$

Following the calculation of head deficit and excess velocity penalty costs, the network infrastructure cost is calculated in accordance with equation (3). All network and hydraulic costs are then summed to give the total network cost (C_t).

$$F = \frac{1}{C_t} \qquad (16)$$

In this instance of the SSGA the reciprocal of the total network cost is taken for the fitness value (*F*) of the candidate's chromosome, thus resulting in the algorithm treating the task as a maximisation problem.

3.1.3.3 Tournament Selection

There are a number of different selection methods available when formulating a GA and it is important to consider a number of different factors. The primary function of a selection operator is to select individuals in the population for reproduction, creating offspring to further the algorithm's search. This is often achieved by promoting the fitter individuals in the population with the expectation that their offspring will possess an even higher fitness, however it is important to consider how the selection method will affect the performance and behavior of the algorithm in combination with the other operators to promote fitness whilst maintaining diversity in the population. A common selection approach is to utilize a fitness proportionate method such as the roulette wheel, where the probability of selecting an individual is proportionate to the fitness of that individual. However this method can exploit the high fitness individuals in early populations resulting in a less diverse population which ultimately causes premature convergence. Such problems can be addressed through the use of scaling methods such as sigma scaling [135] to reduce the likelihood of premature convergence. Another approach is rank selection [136] where individuals are assigned a rank according to fitness, and the probability of the selection of an individual is dependent on its rank. The core issue with ranking methods is that it is difficult to tune the selection pressure to achieve

adequate performance. An additional consideration with the above methods is computational efficiency; the fitness proportionate methods require multiple passes of the population to calculate the selection probability values for the individuals and the rank selection method requires the population to be sorted at each selection stage, both of which are potentially computationally expensive. However, in the case of water distribution network optimization, where the objective function calculations are relatively lengthy, it can be suggested that computation efficiency of the selection operators has a lesser impact on runtime, although it cannot be discounted entirely.

The tournament selection method was chosen for this implementation of the SSGA, amongst other rationale, as it is more computationally efficient compared to that of the other methods mentioned above. This was an important consideration when developing the SSGA as the algorithm would be later used to perform an extensive number of runs (Section 0). Tournament selection also allows the selection pressure to remain relatively constant through the search and is maintained when the variance in fitness of the population is low.

The tournament selection operator first forms a tournament of size tN, where N is the population size and t is a user assigned factor, normally with a value between 0.02 and 0.1 (2% - 10%). Following the formation of a tournament, the operator compares the individual's fitness values and selects the candidate with the highest fitness. The 'winning' candidate is then chosen for reproduction and the rest of the individuals are returned to the population. In the event that a group of individual shares the same highest fitness value, the operator selects an individual at random from that group for reproduction. The tournament selection process is then repeated for the selection of the second parent individual. The operator is implemented in such a way that the individual selected from the first tournament cannot be included in the second tournament.

3.1.3.4 Crossover & Mutation

The crossover procedure is a coarse representation of genetic recombination between two haploid (single-chromosome) organisms in nature. The SSGA employs a single-point crossover operator in which a single crossover position is selected at the gene level at random and the strings

following the crossover point are swapped to form two offspring. This process can result in the effective mutation of a decision variable in the event that the crossover point falls within the divisions of a chromosome sub-string. Other methods such as multi-point and uniform crossover were considered and although effective in some cases greatly increase the variance of the search by increasing the probability of sub-string (pipe) mutation within the chromosome. Although the added variance produced by uniform crossover encourages a more exploratory behavior, which in some cases can be beneficial to the algorithm's search, this method would also be greatly destructive when used in conjunction with the modified heuristic mutation operators (see Chapter 4: for implementation details).

The mutation operator in this formulation of the SSGA mutates the decision variable (pipe) as a discrete entity instead of acting at the single gene level. This 'Pipewise' approach was necessary as the heuristic mutation operators detailed later in this work act at the pipe level and therefore the SSGA's mutation operator needed to operate in the same operational plane to facilitate the unbiased comparison between the different approaches. The SSGA Pipewise operator selects a random substring (decision variable) from the chromosome and decodes it to find the diameter of that pipe. The operator then randomly selects another diameter from the available diameters whilst excluding the current diameter of the pipe which is being acted upon. The new substring is encoded based on the selected diameter and inserted into the chromosome, overwriting the original substring. Unlike bitwise mutation, which acts on each allele in turn and inverts the bit dependant on the mutation probability, the Pipewise operator acts upon each sub-string and has the potential to alter up to and including the number of alleles in the sub-string, thus making the direct comparison between the two approaches unfeasible. The probability of a pipe being mutated is a user defined parameter commonly set between the range of 0.001 & 0.15.

3.1.3.5 Replacement Strategy

Unlike a generational GA where new offspring are placed into a new population separate from their parents, the SSGA inserts offspring back into the active population. To maintain a population of constant size it is necessary to

implement a strategy for the replacement of individuals in the current population with the new offspring. The first step is to select a member of the current population to be replaced; this is normally based on the fitness value of the individuals in the population where the worst individual is replaced. Other approaches include the replacement of the oldest individual and replacement of a random individual within the population. Following the selection of the individual being replaced, the next step is to implement a replacement condition; normally either conditional or unconditional. In the case of conditional replacement, the selected individual is replaced only if the offspring has a better fitness. Unconditional replacement requires no such comparison and the selected individual is replaced by the new offspring. A commonly utilized approach is conditional worst replacement where the worst individual in the population is replaced if the offspring has a better fitness value [137]. It has been suggested that the replacement of the worst solutions in a population will result in a high selection pressure [129].

A number of replacement methods were implemented, including combinations of conditional & unconditional, worst, oldest & random strategies. From preliminary experiments run during the development of the SSGA it was observed that the conditional worst replacement strategy seemed to perform best for this implementation of the water distribution network design problem. Further more extensive experimentation later reaffirmed this (see section 4.1.1.1.2).

3.2 Multi-Objective Genetic Algorithm for Water Distribution Network Design

The formulation of a multi-objective genetic algorithm for the optimal design of water distribution networks is described in this section. As with the single-objective SSGA described previously in this chapter, the multi-objective GA presented here will form a platform for the application of the heuristic based operators described later in this work. It was decided to utilize an adapted version of the existing and well known NSGA-II [63] multi-object algorithm as it has shown to perform well in the field of water system optimization [138].

3.2.1 Multi-Objective Water Distribution Network Design Problem

The multi-objective formulation of the water distribution network design problem is similar to that of the single-objective approach detailed in section 3.1.1. As with the single-objective formulation, the multi-objective approach is primarily concerned with the allocation of diameters to the pipes in the network, with the primary objective to minimize network infrastructure cost. Unlike the single-objective version of the problem however, this approach adapts the junction head deficit and pipe excess velocity constraints into separate objectives. This enables the designer to observe the tradeoff between the hydraulic performance of the network and the infrastructure cost with the view to making better design decisions. The primary objective is the total network cost or infrastructure cost which is given by:

$$f(D_1, \dots, D_n) = \sum_{i=1}^{N} c(D_i, L_i) \qquad (17)$$

Where $c(D_{i,}L_i) =$ cost of pipe i with diameter D_i and length L_i with $N =$ number of pipes in the network. This function is to be minimized during the optimization process. The second objective is to minimize the total head deficit within the network and is given by the following expression:

$$f(H_i, \dots, H_j) = \sum_{i=1}^{J} (H_i) \qquad (18)$$

Where the head deficit in junction *i* is H_i with *J* = the number of junctions present in the network. The third objective is to minimize the total pipe velocity excess in the network and is given by the following:

$$f(V_1, \dots, V_n) = \sum_{i=1}^{N} (V_i) \qquad (19)$$

Where the excess velocity in pipe *i* is V_i with *N* = the number of pipes in the network. The final objective used in this formulation of NSGA-II for the optimization of least cost water distribution networks is a measure of network smoothness. A smooth network is achieved when pipes can be seen to 'smoothly' transition from large to small diameters from source to the extremities of the network. The subject of network smoothness is covered in more detail in section 3.4.2 of this chapter. In this case the objective is to minimize the number

of pipe smoothing violations in a candidate network and is given by the following expression:

$$f(S_1, \ldots, S_n) = \sum_{i=1}^{N}(S_i) \qquad (20)$$

Where the smoothing violations of pipe *i* is S_i with *N* = the number of pipes in the network. For example, in the case where a pipe *i* violates the smoothing rule S_i = 1otherwise if the rule is satisfied S_i = 0.

As with the single-objective formulation, the multi-objective version of this problem also relies on a hydraulic solver to ensure the majority of hydraulic constraints are met by calculating junction inflow and outflow (4) and headloss (5) along with other hydraulic considerations.

3.2.2 Multi-objective Genetic Algorithm for the Design of Water Distribution Networks

This section describes the application of NSGA-II to the multi-objective water distribution network design problem described in section 3.2.1. As with the SSGA described earlier in this chapter, NSGA-II will form a base on which the heuristic operators described later in this work will be applied. The notion being that integrating the heuristic based operators into an existing algorithm will provide a good standard for the assessment of the operators when comparing the base algorithm to the algorithm with modified operators.

Figure 3-5 shows an overview of the NSGA-II implementation used in this work. The initial population P_0 is initialized by creating *N* randomly generated individuals. This implementation of NSGA-II employs an encoding scheme identical to that of the SSGA described earlier in this chapter, utilizing a binary pipe diameter representation and largest pipe redundancy. Each of the starting individuals is evaluated to obtain values for each objective, again solutions are decoded and evaluated in much the same way as in the SSGA; utilizing EPANET 2 to run the hydraulic simulations of the system to obtain the information required to assess the hydraulic performance of the network. The only difference between this implementation and that of the SSGA is that the hydraulic data obtained is used separately as objective values rather than being combined in one fitness function.

3.2.2.1 NSGA-II Execution

Given a set of objectives an individual is said to be Pareto dominate if the objective values are not inferior to those of the other individuals, and additionally there is at least one objective where the individual performs better. A solution is said to be non-dominated when no other solution present in the population provides any improved objective values without degrading other objective values. The non-dominated solutions in the population go to form the Pereto front. Utilizing the objective values obtained from evaluation, the individuals are sorted into fronts $F_1,\ldots,F_i$ where i is the number of resultant fronts) using a fast non-dominated sorting algorithm [63].

Following the non-dominated sort, the crowding distance is assigned to all individuals in the population. The crowding distance is an estimation of the density of solutions surrounding a particular point in the population. This is achieved by taking the average distance of the two points on either side of this point along each of the objectives.

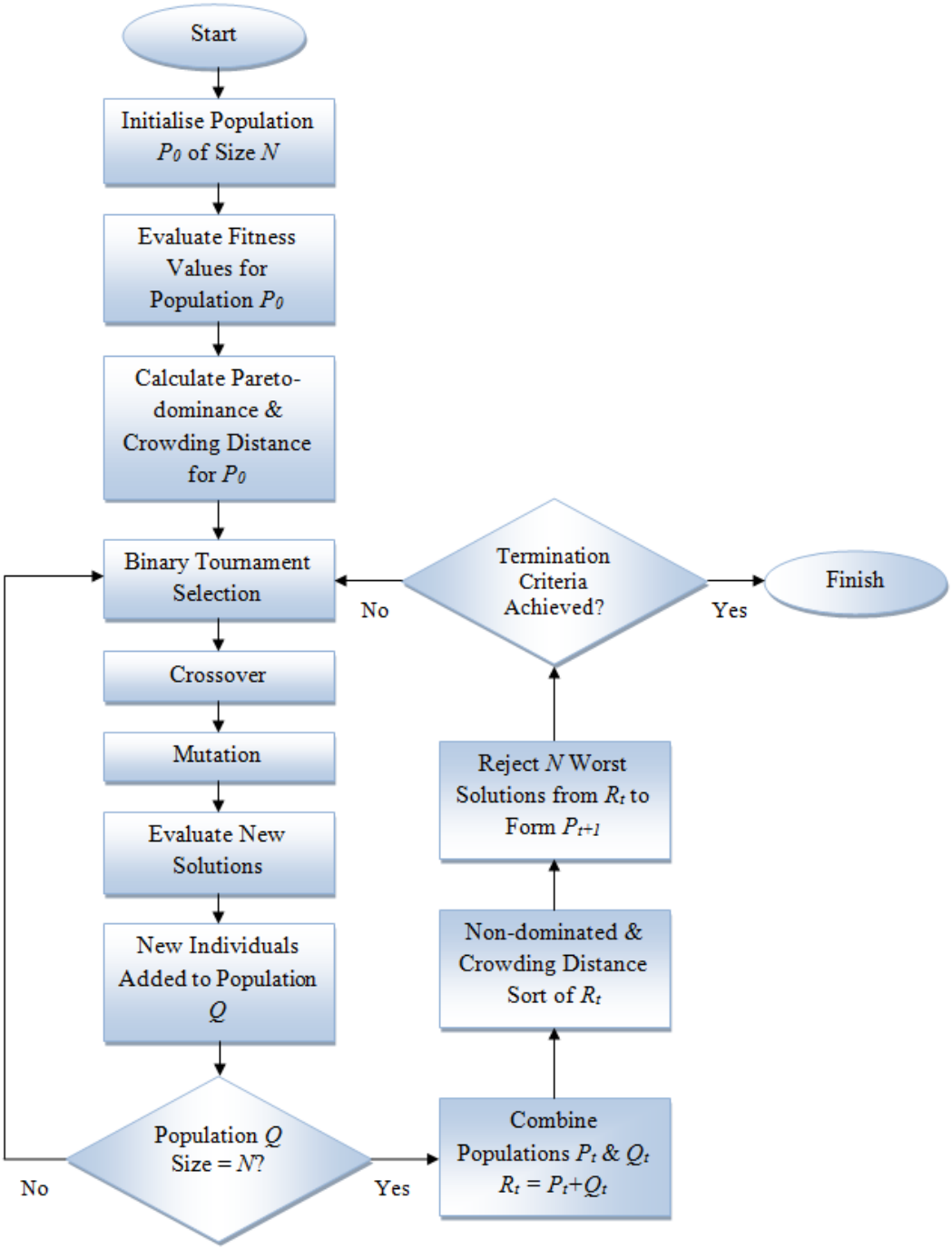


Figure 3-5. Flow chart representation for NSGA-II

Following the non-dominated sort and crowding distance operations, tournament selection is employed to select the individuals for the reproduction operators. A tournament size of *tN* individuals are chosen at random from the population for each of the two resultant parents, where *t* dictates the number of individuals selected with a value normally between 0.02 & 0.1 resulting in 2% - 10% of the population being selected for each tournament. The selected

individuals in the tournament are then subjected to the crowded comparison operator ($\geq_n$). Every individual *i* in the population has a non-dominated rank (i_{rank}) and a local crowding distance ($i_{distance}$), using these attributes individuals can be compared to one another. An individual is said to be better than another if it has a better rank or has the same rank and greater local crowding distance:

$$i \geq_n j \Leftrightarrow (i_{rank} < j_{rank}) \mid \left((i_{rank} = j_{rank}) \,\&\, (i_{distance} > j_{distance})\right) \qquad (21)$$

After the individuals in a tournament are subjected to the crowded comparison operator, the best performing solution is selected for reproduction. A second tournament is performed and the two selected parents are then passed to the crossover operator which in this implementation is a single-point procedure in which a single crossover position is selected at the gene level at random and the strings following the crossover point are swapped to form two offspring. The two offspring of this procedure are then mutated using the Pipewise mutation operator (see section 3.1.3.4).

Following mutation the new individuals are added to population Q_t. If population Q_t has not reached a size equal to *N* the algorithm returns to the tournament selection operator and selects another two individuals from the current population P_t. Upon population Q_t reaching a size of *N,* populations P_t *and* Q_t are combined to form R_t. Unlike the SSGA, NSGA-II is a generational genetic algorithm although it employs an elitist strategy that combines the best parents with the best offspring. Figure 3-6 shows the NSGA-II procedure for producing the new parent population P_{t+1}.

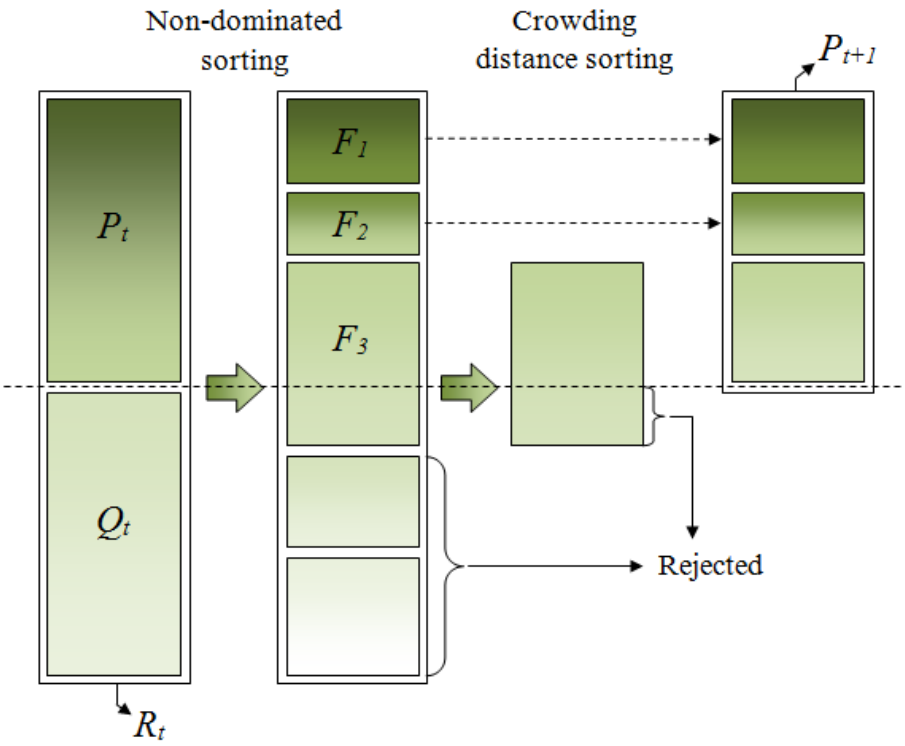


Figure 3-6. NSGA-II procedure

Following the formation of population R_t of size $2N$ the individuals are sorted according to non-domination. The new parent population P_{t+1} is created by adding individuals from the first front until the population size exceeds N. Subsequently, the individuals from the last accepted front are sorted according to the crowded comparison operator and the first N solutions are selected. In the example shown in Figure 3-6 the first and second fronts are added to the new population, however, in this case front three straddles the population size cut off and is sorted according to local crowding distance. The solutions from the sorted third front are added to the new population in order until P_{t+1} reaches size N. The remaining individuals from front three are rejected along with those from the remaining fronts.

After the formation of the new parent population the algorithm checks whether the termination criteria has been achieved. In this implementation of NSGA-II the algorithm will terminate after a given number of generations. If the termination criterion has not been met, the algorithm continues from the tournament selection stage of the process.

3.3 Benchmark WDN Design Problems

The following WDN design problems were selected from the literature to assess the performance of the algorithms presented in this work. The problems range in size and complexity from a single source network with 8 decision variables to a multi-reservoir, quad source network with 317 decision variables. Also included in this set of benchmark problems are two large real-world networks, Network A and Network B with 632 and 1277 decision variables respectively. All of the following benchmark networks are least-cost water distribution network design problems where the goal is reduce network cost through the selection of pipe diameters whilst satisfying the hydraulic constraints set by the problem. All of the problems presented here have hydraulic junction head constraints although only a selection of the networks have pipe flow velocity constraints.

3.3.1 Two Loop

The Two Loop [127] problem consists of a single reservoir and 8 decision pipes arranged in to twin loop. There are 14 available pipe diameters, resulting in an effective search space size of 1.48×10^{9}. This problem includes junction hydraulic head constraints but does not include flow velocity constraints.

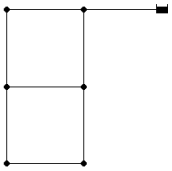


Figure 3-7. Layout diagram of the Two Loop network

3.3.2 Foss Poly 1

Foss Poly 1 is an instance of the real-world Fossolo network [139], which consists of a single reservoir and 58 decision pipes. Foss Poly 1 uses polyethylene as a pipe material and has a large selection of pipe diameters with 22 available, with a resultant solution space of 7.25×10^{77}.

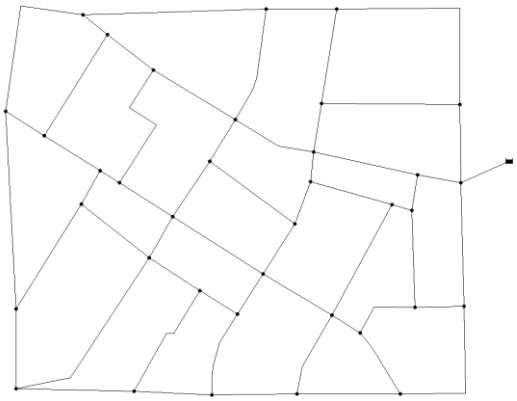


Figure 3-8. Layout diagram of the Foss network

3.3.3 Hanoi

The Hanoi problem [140] is a representation of single source network consisting of three loops, 34 decision pipes and 6 available pipe diameters with a resultant search space of 2.86×10^{26}. Based upon the trunk main layout for the city of Hanoi, Vietnam, the problem requires that a minimum fixed head of 30m is reached at all nodes in the network. In this implementation of the problem, there are no pipe flow velocity constraints imposed.

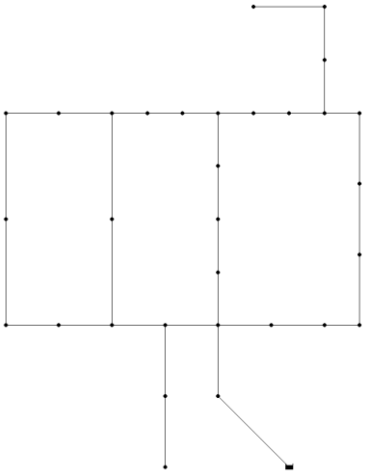


Figure 3-9. Layout diagram of the Hanoi network

3.3.4 New York Tunnels

The New York Tunnels network is a parallel expansion problem where 21 pipes are considered for duplication with the objective to determine the most economically effective design for adding to the existing network of pipes that make up the primary water distribution system for the city of New York, USA.

There are 15 available pipe diameters available as well as the option to ‘do nothing’ which results in no duplicate pipe being added. The problem has fixed minimum head requirements at all junctions in the network, but does not have any restrictions on the velocity of pipe flow.

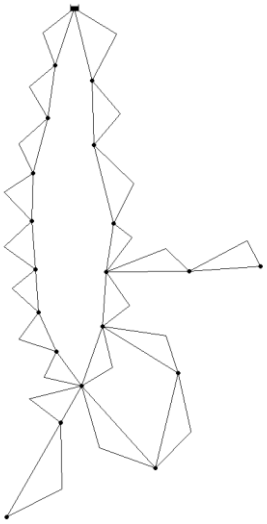


Figure 3-10. Layout diagram of the New York Tunnels network

3.3.5 Modena

The Modena water distribution network [139] is a representation of the water supply system of the city of Modena, Italy. The network consists of 4 sources and 317 decision pipes with 13 available pipe diameters to choose from. The formulation of this problem includes both junction minimum head requirements and pipe flow velocity constraints.

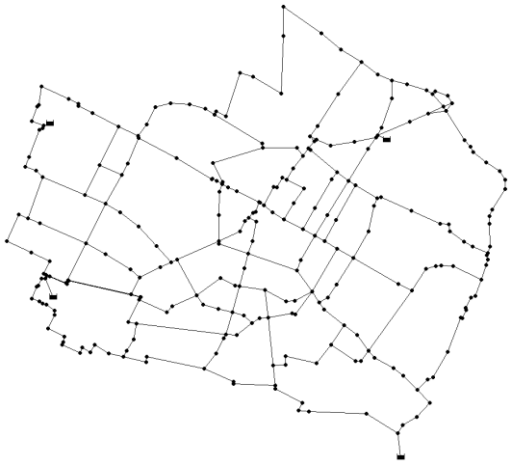


Figure 3-11. Layout diagram of the Modena network

3.3.6 Network A

Network A [141] is a real-life industrial water distribution network consisting of a single source reservoir with 632 decision pipes and 20 available pipe diameters. The problem has fixed minimum head requirements at all junctions in the network, but does not have any restrictions on the velocity of pipe flow.

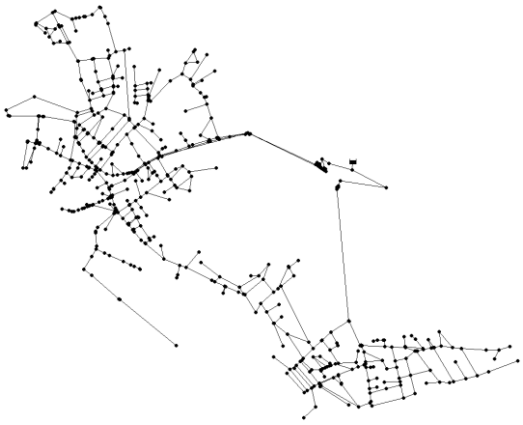


Figure 3-12. Layout diagram of the Industrial Network A network

3.3.7 Network B

Network B [141] is a real-life industrial water distribution network consisting of a single source reservoir with 1277 decision pipes and 26 available pipe diameters. The problem has fixed minimum head requirements at all junctions in the network, but does not have any restrictions on the velocity of pipe flow.

Figure 3-13. Layout diagram of the Industrial Network B network

3.4 Water System Heuristic Development

The Genetic Algorithm has proven to be a versatile process for solving a large variety of optimization problems spanning many fields and disciplines. The strength of the approach comes from the ability the GA has to traverse large search spaces, avoiding local optima and therefore can be viewed as a truly global search technique. The performance and versatility of the GA can be attributed partly to the independence it has over the problem being undertaken. Although seen as an asset, this problem independence can have a detrimental effect on performance in the case where the algorithm has not been tuned to a great enough extent to solve the problem at hand.

For the problem of water distribution network design the GA relies on genetic operators such as crossover and mutation to alter the configuration of the network. These operators however are blind to the direct effect any changes made to elements of the network have on the overall performance of the resultant solution. For example from the perspective of the GA, a change in the diameter of a pipe has no bearing on the hydraulic behaviour of connected elements until the resultant design is evaluated, although an engineer would know that the head at adjacent junctions would be affected. The performance of a newly created network is only known following solution decoding and hydraulic simulation and although this abstraction is partly why GAs can be applied to a large number of different water system design problems, there is definite scope for the integration of problem specific knowledge into the approach. An important consideration when integrating problem-specific knowledge into a genetic algorithm is computational efficiency. The most computationally demanding operations are solution evaluations and in the case of water distribution design problems this comes in the form of the hydraulic simulations. Therefore it is important not to incur any additional objective function evaluations where possible.

Another consideration is the apparent lack of uptake and utilisation of techniques such as EAs by engineers in the field of water distribution network design. One likely reason for this is the solutions produced by such methods are usually only ‘mathematically feasible’ and not ‘engineering feasible’ which

results in the engineer having to manually correct features of a solution network to better suit real world application and deployment.

In this section two separate water system heuristic methods are described both of which draw upon expert engineering knowledge and techniques with a view of integration into a genetic algorithm to improve search performance and solution feasibility. These heuristic approaches will form the basis of the novel contribution of this work.

3.4.1 Hydraulic Deficit Approach

One of the primary constraints of the least-cost water distribution network design problem is ensuring junction head requirements are met throughout the network. This can be a complex task as this constraint is in direct conflict with the primary objective of minimizing cost through the reduction of pipe diameters. The key issue here is headloss; as a fluid flows through a pipe water pressure is lost due to friction along the inner surfaces of the pipe. It can be seen from the Hazen–Williams equation (see section 3.1.1) that headloss is directly influenced by the length, diameter, roughness and flow rate of a pipe. When solving the least-cost water distribution network design problem, a GA only has direct influence over the diameters of pipes in the network as the length and roughness of the pipes are normally fixed parameters of the problem. Therefore to reduce headloss, the diameter of a pipe has to be increased; however as stated previously this increases the cost of the pipe and directly conflicts with the objective function which is trying to minimize infrastructure cost.

One of the key characteristics of a water distribution network is that the diameters of pipes close to the source have a greater hydraulic influence over the whole system. For example if a pipe close to the source has a small diameter, large amounts of headloss can be introduced and subsequently the downstream junctions will not receive the required hydraulic head, this can be referred to as a ‘bottleneck’. Figure 3-14 shows two versions of a simple water distribution network. The first contains a pipe (2^{nd} from source) which is introducing a large amount of headloss due to its small diameter thus resulting in the downstream junctions not receiving enough pressure and therefore reporting a head deficit. The bottleneck is eliminated by increasing the diameter

of the offending pipe, hence reducing headloss and increasing the subsequent pressure in the downstream junctions. This approach is often applied by water systems engineers when designing distribution networks to eliminate hydraulic bottlenecks, unlike a standard genetic algorithm which cannot implement this simple process as the operators do not have awareness of the hydraulic behavior of the individual parts of the system during the crossover and mutation stages.

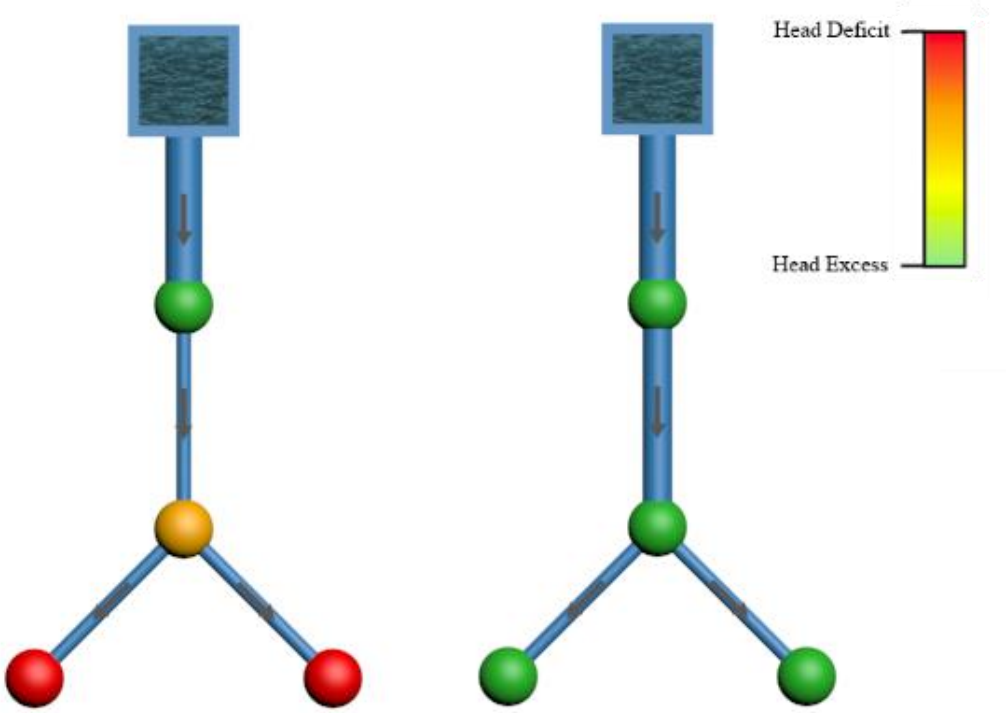


Figure 3-14. Bottleneck identification and elimination example

It is proposed that this method of bottleneck identification and elimination can be integrated into a GA by applying the heuristic directly to a modified version of the mutation operator. The aim of this operator is to guide the algorithm's search to the feasible solution space in a fast and efficient manner utilising constraint violation data from previous fitness evaluations. As stated previously, computational efficiency is an important consideration when applying any rule based operator into a standard algorithm and unlike some other constraint handling techniques such as repair algorithms, the proposed mutation operator will not perform any additional partial or full fitness evaluations. This is achieved by applying this constraint based rule directly to the genotype without evaluating the effect this process has on the phenotype.

During the evaluation of the solutions in the initial population the program logs the directional flow information of each pipe. Using this data, each junction and its immediate upstream pipe and junction are logged making it possible to identify pipes that are limiting junction head down-stream.

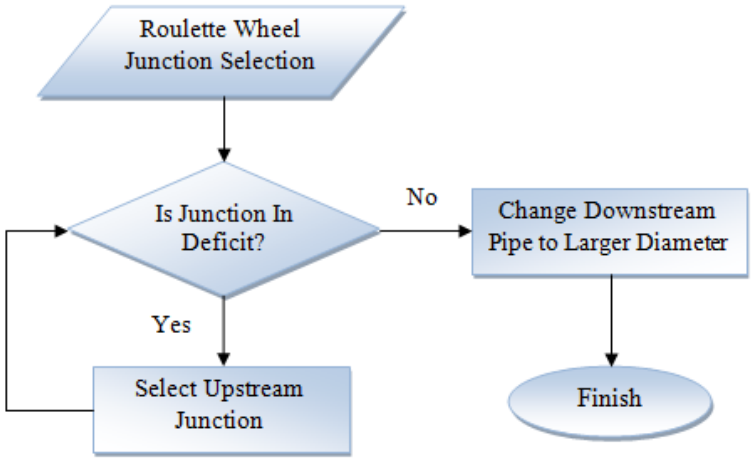


Figure 3-15. Flow chart representation of the hydraulic bottleneck elimination algorithm

Figure 3-15 shows a flow-chart representation of the hydraulic bottleneck elimination mutation operator process. The modified mutation operator first selects a junction through the use of a roulette wheel method which assigns wheel segment sizes using head deficit information obtained during the fitness evaluation of the solution, resulting in junctions with a high pressure head deficit having a greater probability of being selected. Once a junction is selected the heuristic searches upstream of the selected junction until a junction is found which is in pressure head excess. The pipe immediately downstream of the discovered junction is then mutated to a greater available diameter. It has been shown that incremental pipe diameter changes during mutation are normally beneficial to the search of a GA as large changes to network elements can have a drastic effect on the overall solution quality, sometimes for the worst. However it was decided that only allowing the operator to make single diameter increments would potentially slow the rate of search of the algorithm and therefore a weighted roulette wheel approach is used to select the new diameter. This is achieved by firstly populating a list of all available pipe

diameters greater than the diameter of the selected pipe and placing them in ascending order. Each diameter is then assigned a probability of selection ($\mathbb{P}(I)$) using the following expression:

$$\mathbb{P}(I) = \begin{cases} \frac{1}{2^i} & \Rightarrow \quad (i < N) \\ \frac{1}{2^{i-1}} & \Rightarrow \quad (i = N) \end{cases} \qquad (22)$$

Where *i* is the list position of the diameter and *N* is the total number of available pipe diameters present in the list. This results in the smaller diameters in the list having the greater probability of selection and the largest diameters having a smaller selection probability.

In the event where a network contains no junctions in deficit, the modified mutation operator concentrates on reducing network cost by targeting oversized pipes. Firstly a junction is selected through the use of a roulette wheel where the segment size is directly proportional to the amount of head excess at each junction, resulting in junctions with higher head excess having a greater probability of being selected. The pipe directly upstream of the selected junction is then mutated to a smaller diameter using a similar weighted roulette wheel approach to that of the diameter increasing method described above. In this case the available pipe diameters smaller than the diameter of the selected pipe are placed in a list in descending order. The probability of a diameter being selected from the list is dictated by equation (22) where the smaller the diameter the greater the probability of selection.

Due to the dependency the modified mutation operator has on a solution's pressure head information, mutation cannot be applied post crossover without the need to re-evaluate the hydraulic network of resultant solutions. Therefore the mutation operator needs to precede the crossover operator in order to preserve the hydraulic information gained from the hydraulic simulation of the original solution.

3.4.2 Pipe Smoothing Approach

The pipe smoothing approach described in this section directly targets elements of a network with the aim to increase network smoothness (in terms of

progression from one diameter to the next) using network element awareness and an elementary heuristic. This is based around the principle that in gravity fed WDNs the diameter of any pipe is never greater than the sum of the diameter(s) of the directly upstream pipes. Networks that adhere to this rule can be seen to ‘smoothly’ transition from large to small diameters from source to the extremities of the network. Figure 3-16 shows an example of a ‘smooth’ solution for the Hanoi problem where the arrows indicate flow direction.

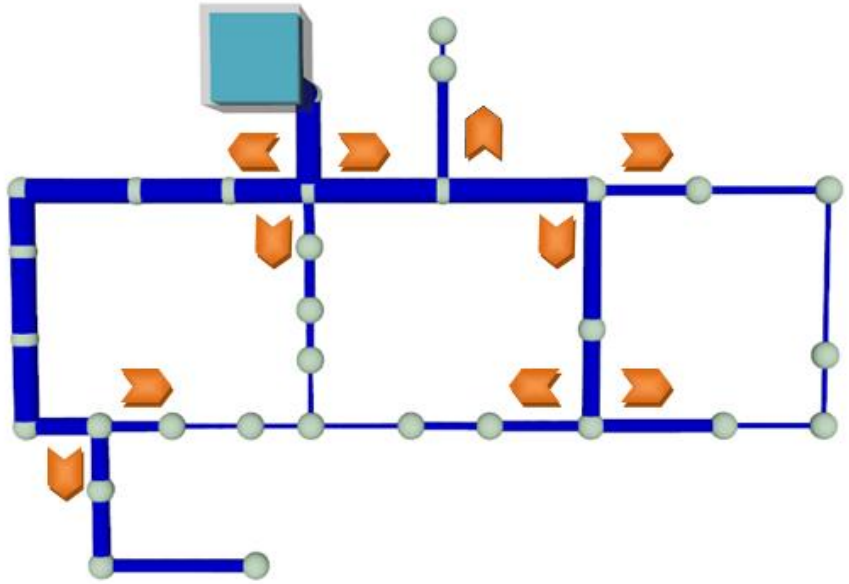


Figure 3-16. Smooth pipe diameter transistions example on the Hanoi network

This rule is routinely and implicitly applied by engineers when designing such networks as it makes little sense to follow a smaller diameter pipe with a larger one in the majority of circumstances. A larger pipe downstream will cost more to install and will not add to the hydraulic capability of the system as it will be constrained by the smaller diameter pipe upstream. One further negative aspect of this arrangement is that velocities will be lower in the larger pipe and high water age can become an issue leading to poor water quality. A standard GA of course will mutate some of these inconsistent pipe selections from the final solution as they have a corresponding improvement in the cost function and no hydraulic penalty. However extensive experimentation has shown that even well-optimised solutions after hundreds of thousands of generations of a standard EA still contain significant numbers of incorrectly sized pipes in larger networks.

The pipe smoothing mutation operator applies the rule described above directly to the genotype without evaluating the effect this process has on the phenotype. The heuristic employed by the pipe smoothing mutation operator is developed from the network topology of a specific problem and remains constant throughout the evolutionary process. It is the aim of the heuristic to guide the algorithm's search to the engineering feasible solution space to locate smoother WDN designs whilst maintaining the performance of a standard genetic algorithm. The pipe smoothing mutation operator does not perform any additional partial or full fitness evaluations, except a single hydraulic simulation at initialisation to determine flow directions. This was an important consideration when developing the Pipe Smoothing Genetic Algorithm (PSGA) as additional fitness evaluations would require further hydraulic evaluations, increasing algorithm run time.

Figure 3-17 shows two configurations of parallel pipes entering and exiting a junction, the first of which (left) violates the pipe smoothing rule as the sum of the downstream pipe diameters is greater than the sum of the diameters of the upstream pipes. It is the goal of the pipe smoothing heuristic to modify the diameters of the downstream pipes so that the sum of the diameters is equal or less than the sum of the diameters of the upstream pipes, resultant in a configuration which satisfies the pipe smoothing heuristic (right).

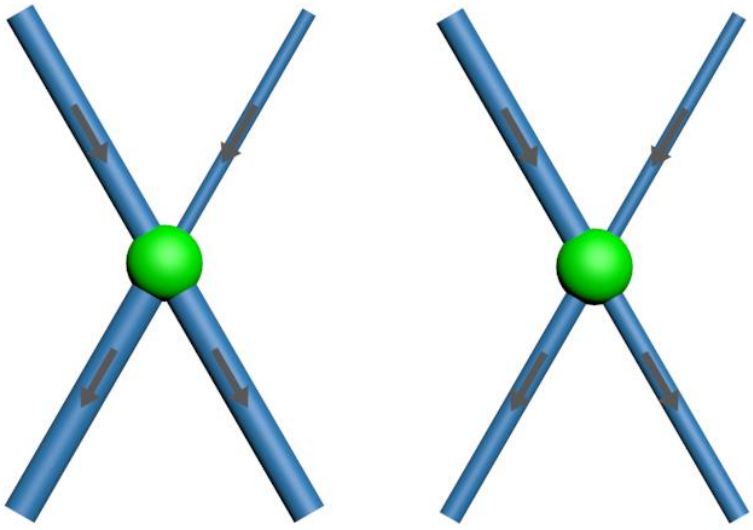


Figure 3-17. Downstream pipe smoothing rule violation (left) & corrected downstream diameters that satisfy the smoothing constraint (right)

The pipe smoothing mutation operator randomly selects a pipe to be mutated. The maximum allowable diameter of the current pipe is calculated by taking the sum of the diameters of the immediately upstream pipes and subtracting the sum of the diameters of any pipes parallel to the selected pipe. This is described by the following expression:

$$D_s^{max} = \left(\sum_{i=1}^{U} D_i\right) - \left(\sum_{j=1}^{P} D_j\right) \qquad (23)$$

Where D_s^{max} is the maximum allowable diameter of selected pipe *s*, D_i is the diameter of upstream pipe *i* with *U* being the total number of directly upstream pipes and D_j is the diameter of parallel pipe *j* with *P* being the total number of pipes parallel to the selected pipe.

Much the same as the hydraulic deficit approach, this operator employs a skewed roulette wheel approach to the random selection of the pipe diameter. This is achieved by weighting the larger pipe diameters that fall within the maximum allowable size so that the larger the diameter, the higher the probability there is of selection. A list is first populated of all available pipe diameters equal to and less than the maximum allowable diameter of the selected pipe. The list is sorted into descending order by diameter and each diameter is then assigned a probability of selection ($\mathbb{P}(I)$) using the expression detailed in the previous section (equation (22)). This process prevents the heuristic from selecting small diameters on every application. With an upper-bound on possible diameters, the repeated application of a uniform probability of selection would result in an undersized, hydraulically infeasible network. Upon a diameter being selected the pipe being mutated is changed to the selected diameter.

To function correctly the mutation operator requires each pipe in the network to be ‘aware’ of the pipes directly up and down stream of their location. When changes are made to a WDN there is a possibility that flow direction could change in some pipes hence swapping up & down stream pipes relative to the pipe in question. The flow direction is logged at each hydraulic evaluation of the network; therefore to preserve this hydraulic data the pipe smoothing mutation operator precedes the crossover operator.

3.5 Conclusion

An approach to the optimal design of water distribution networks has been described in this chapter along with the application of the problem to both single and multi-objective genetic algorithms. The chapter provides a detailed description of the formulation of a single-objective Steady-state Genetic Algorithm (SSGA) and the multi-objective NSGA-II for the design of water distribution networks. The reasoning behind the selection of certain operators was discussed as well as their configuration. The SSGA was formulated to provide a platform on which the engineering inspired heuristics can be applied; providing a direct method for measuring the effect each heuristic method has on the performance of the algorithm. A variety of benchmark water distribution system problems have been selected from the literature and presented in this chapter. These problems which range in complexity were selected to assess the performance of the algorithms presented in this work (Chapters 4 & 5).

The genetic algorithm has shown to be a versatile process for solving a large variety of optimization problems due partly to the independence it has over the problem being solved. Although a seen as a strong asset, this problem independence can sometime have a detrimental effect on performance if the algorithm has not been tuned correctly to the problem being solved. In the case of water system design problems it was stated that the operators of the genetic algorithm are ‘blind’ to the direct effect that changes made to a solution have on the hydraulic performance of the resultant network. It was therefore proposed that a standard genetic algorithm could potentially benefit from the integration of engineering inspired heuristics. Two such heuristics were proposed in this chapter, both of which are inspired by engineering principles employed by water system engineers. The first heuristic presented in this chapter is based on the process of hydraulic bottleneck identification and elimination to promote solution feasibility in terms of hydraulic junction head. The second heuristic presented in this chapter is built upon the idea of network smoothing where the diameter of any pipe is never greater than the sum of the diameter(s) of the directly upstream pipes. This rule is routinely and implicitly applied by engineers when designing water distribution networks as it makes little sense to follow a smaller diameter pipe with a larger one in the majority of circumstances. As with the

previously mentioned heuristic, the pipe smoothing rule is applied through the mutation operator of the genetic algorithm. In both of these implementations, computational efficiency was an important consideration when incorporating these heuristics into an algorithm therefore a method for the integration of these heuristics into the mutation operator has been developed which does not require any additional partial or full solution evaluations. It is shown in the following chapters that the integration of these heuristics will not only improve the general performance of a standard genetic algorithm in terms of the objective function(s) but also promote hydraulic and engineering feasibility and potentially provide a mechanism to improve their uptake in the industry.

Chapter 4: Heuristic Based Genetic Algorithm Development

An important factor to consider when implementing a genetic algorithm is deciding which values to use for the various parameters, such as population size, tournament size, crossover rate and mutation rate. Typically speaking, these parameters have a nonlinear relationship with one another, making the simultaneous optimization of algorithm parameters infeasible. The initial focus of this first section is to explore the effect that varying parameter values has on the performance of a single-objective Steady State Genetic Algorithm (SSGA) when it is applied to a range of Water Distribution Network (WDN) design problems. The parameters of the GA are modified systematically for a number of different water distribution network problems. There are two primary aims for this section of work: to characterise the near-optimal parameter settings for GAs on different problem types and to assess the differences between problem instances in the WDN optimisation domain; and to develop a highly tuned formulation for the GA to take forward to the next experimentation section. It is believed that a systematic and robust study such as this has not been conducted in this field and therefore represents new knowledge.

Following the parameter tuning section the development and testing of an Adaptive Locally Constrained Genetic Algorithm (ALCO-GA) is described. The algorithm is based on a standard Steady State Genetic Algorithm (SSGA) with the addition of an adaptive heuristic based mutation operator which utilises hydraulic head information to guide the search of the algorithm into the feasible solution space.

The final section of this chapter introduces the Pipe Smoothing Genetic Algorithm (PSGA) which is based around the principle that in gravity fed water distribution networks the diameter of any pipe is never greater than the sum of the diameter(s) of the directly upstream pipes. This heuristic is integrated into the mutation operator of the SSGA to form the new PSGA. Single Objective

4.1 Genetic Algorithm Parameter Tuning

The purpose of this section is to explore the effect of varying the different parameters of a single-objective steady state genetic algorithm on a number of small to medium size WDN design problems from the literature. The results of the following experiments will not only give an insight in to the effect that different parameters have on the performance of the algorithm but also ensure the algorithm is performing at near peak performance to provide a fair comparison to the heuristic based algorithms presented later in this thesis.

4.1.1 Experimental Setup

In this section an experiment is described which is designed to identify the optimal parameter settings for the Steady State Genetic Algorithm (SSGA) for each of the small to medium sized water distribution network problems detailed earlier in this work. It is also the aim of this experiment to provide an opportunity to draw conclusions into how the parameter values effect algorithm performance in a number of differing problems. For all of the following experiments presented in this section the population size of the SSGA was set to 100 and single-point crossover was employed, the various parameters for each experiment are described in their respective section. The SSGA was implemented in C++ with the EPANET 2.0 toolkit employed to perform hydraulic simulations. All experiments presented in this section were run on an Intel Core i7-4770K 3.5GHz PC.

4.1.1.1 Parameter Tuning

The SSGA has a number of parameters which need to be assigned by the user. Arriving at appropriate parameter values for a Genetic Algorithm (GA) is still considered a great challenge in the field of evolutionary computation. For the most part researchers in this field acknowledge the importance of selecting good parameter values is the key to ensuring good GA performance. However, only limited research has been conducted into this area with the majority of practitioners still relying on decades old conventions, impromptu operator choices and very limited scale experimentation. The problem of setting the parameters of a GA commonly falls into two areas, parameter tuning and parameter control[142], also known as parameter adaptation. In the case of

parameter control the parameter values are changed during the course of the GA run dictated by a control strategy. However, methods of this type still require initial parameter values and often the control strategy itself relies upon some form of tuning to achieve optimal effectiveness. Parameter tuning on the other hand is simpler in the sense that only one value for each parameter is required to be provided. Nonetheless, the problem of tuning a GA for a specific application can still be considered difficult due to the large number of options available to the user and the limited information regarding the effect GA parameters have on GA performance. Although there are a number of algorithms available for the tuning of GA parameters, employing such a method would somewhat prevent the observation of the effects differing parameter values have on the GA performance. Therefore it was decided that a constrained exhaustive search approach would be employed to tune the SSGA to each network problem.

4.1.1.1.1 Penalty, Tournament Size & Mutation Rate

The parameters selected for the initial experiment were penalty factor, tournament size, and mutation rate. From initial experimentation these parameters had the greatest effect on algorithm performance compared with the other available parameters such as population size and crossover rate. It was decided that the inclusion of these additional parameters into the experiment would be infeasible to implement due to computational run time constraints.

For each individual network problem eight evenly distributed parameter values were selected for each of the test parameters. These values were acquired through preliminary small scale experimentation designed to highlight the parameter range in which the optimal parameter value lies. Using an exhaustive approach, each of the possible parameter value combinations were tested on the selected network problem. For each parameter value combination SSGA was run 50 times, each for 200,000 objective function evaluations utilizing a different randomly seeded starting population for each run. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size was fixed at 100 individuals, single-point crossover was utilized and a conditional worst individual replacement strategy was implemented.

To evaluate the performance of the algorithm for a given parameter value, the mean results from all runs pertaining to that value where calculated and placed in a table. Due to the large number of runs (512 parameter combinations and 25,600 individual runs) and resultant data for each network problem it was decided that it was also necessary to use a standard box plot (1st and 3rd quartiles with min/max whiskers) to visualize the data for each parameter value.

4.1.1.1.2 Replacement Strategy

Selecting the correct replacement strategy for a SSGA is an important consideration with regard to the performance of an algorithm. Thus following the tuning of the core parameters detail above a second experiment was conducted to gauge the best strategy for the replacement of individuals within the active population at the end of each algorithm iteration. As detailed in section 3.1.3.5 the first step in the replacement process is the selection of the individual to be replaced. The selection of this individual can be based on a number of criteria: The worst fitness value in the population, the oldest individual (measured in algorithm iterations) and random selection. Following the selection of the individual to be replaced there are normally two options, conditional and unconditional replacement. Conditional replacement requires the newly created offspring to be fitter than the individual selected to be replaced whereas unconditional replacement replaces the selected individual to be replaced with the new offspring regardless of fitness value.

The replacement strategy experiment presented later in this section tests all combinations of the afore mentioned approaches: Worst, oldest and random replacement individual for both unconditional and conditional strategies. For each network problem, SSGA utilises the optimal parameter values obtained from the previous parameter tuning experiments. As with the parameter tuning experiments, for each replacement strategy configuration SSGA was run 50 times, each for 200,000 fitness evaluations again using a different random seed for each individual run. The mean fitness and feasible network cost was then calculated from resultant data and presented in a table. The data was also compiled into a standard box plot to enable the visualization of the data for each replacement strategy.

4.1.2 Results

This section presents the results and discussion for both the parameter tuning and subsequent replacement strategy experiments detailed above. Due to the large number of runs and subsequent amount of data, the majority of processed results are not presented in this section, however they are all available in Appendix i.

4.1.2.1 Penalty, Tournament Size & Mutation Rate

The first part of this section presents the parameter tuning results for the Hanoi network, with each algorithm parameter results separated into table and box plot format. The first parameter results presented are for the penalty factor. In this experiment the penalty variable ranged between $500,000 and $4,000,000 in $500,000 increments. For ease of interpretation the values of the table (and subsequent tables) have been highlighted utilizing a conditional grey scale where the tone of the highlight relates directly to table value compared with the other values in that column. In addition, the best result in each table column is displayed in bold for additional clarity.

The first two result columns in the table show the best feasible fitness and best feasible cost values for each penalty factor. A feasible result is classed as a solution which satisfies all the hydraulic constraints of the problem, in this case solutions with no hydraulic head deficit. These values are obtained by taking the best feasible result from each set of 50 runs with that penalty cost and calculating the mean value. The mean best feasible fitness, mean best feasible cost and mean iterations to feasible values are found by taking the best results from all the runs for that parameter value and then calculating the mean. The best know solution values are the average percentage of best solutions which are equal to the best known solution for that problem, in the case of the Hanoi problem this is $6,081,220.

Table 4-1. Mean penalty cost results for the Hanoi problem – SSGA parameter tuning

Mean Penalty Cost Results							
Penalty	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
$ 500,000.00	1.63578E-07	$ 6,113,631.88	**1.58877E-07**	**$ 6,298,390.16**	$ 117,962.98	**249.61**	2%
$ 1,000,000.00	1.636E-07	$ 6,112,699.69	1.58739E-07	$ 6,303,661.41	**$ 114,756.34**	251.17	2%
$ 1,500,000.00	1.63601E-07	$ 6,112,725.78	1.58812E-07	$ 6,300,633.75	$ 114,906.78	250.69	2%
$ 2,000,000.00	1.63593E-07	$ 6,113,027.03	1.58722E-07	$ 6,304,321.72	$ 118,038.95	251.24	2%
$ 2,500,000.00	**1.63651E-07**	**$ 6,110,783.91**	1.58654E-07	$ 6,307,037.66	$ 118,964.97	251.25	2%
$ 3,000,000.00	1.63467E-07	$ 6,117,838.59	1.58666E-07	$ 6,306,793.75	$ 116,336.45	251.29	2%
$ 3,500,000.00	1.63569E-07	$ 6,113,989.38	1.58612E-07	$ 6,309,069.38	$ 117,772.46	250.49	2%
$ 4,000,000.00	1.63567E-07	$ 6,114,033.59	1.58628E-07	$ 6,308,491.72	$ 118,967.78	250.83	**2%**

From the table it can be seen that a penalty value of $500,000 provides the best results in terms of the mean best feasible fitness and network cost, with the average performance of the algorithm diminishing as the penalty factor is increased. However on closer inspection, the difference between the best and worst mean best feasible cost values is only approximately $10,679, less than a tenth of the best found standard deviation.

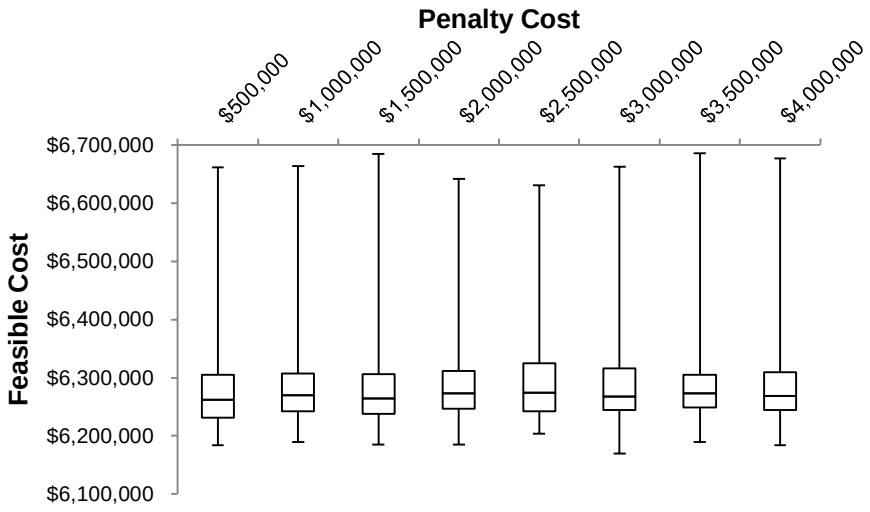


Figure 4-1. Box plot showing mean penalty cost results for the Hanoi benchmark problem

The box plot shows in more detail the spread of these results and suggests there is very little significant difference between the different penalty

factor values. Further statistical testing utilizing the Mann-Whitney U test on these results shows that there is no statistical significance ($p > 0.05$) between the best (penalty = \$500,000) and the worst (penalty = \$3,500,000) set of results. The penalty factor also does not seem to have a much of an effect on the evaluations it takes to reach a feasible solution.

The following table shows the mean results for the tournament size experiments. For this experiment the tournament size ranged between 2% and 9% in 1% intervals. The tournament size is the percentage of the total population size, which in this case is 100. Therefore, a tournament with size 5% will contain 5 randomly selected individuals. From this set of results it is clear that tournament size has a stronger influence over the performance of the algorithm than the penalty factor. In this case the SSGA performs better with a lower tournament size, obtaining the best results for mean best feasible fitness and network cost at a tournament size of 2% of the population. The number of best known solutions found was also highest at this tournament size.

Table 4-2. Mean tournament size results for the Hanoi problem – SSGA parameter tuning

Mean Tournament Size							
Tournament Size	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
2%	1.63604E-07	\$ 6,112,581.72	**1.59595E-07**	**\$ 6,268,503.13**	**\$ 103,194.13**	431.31	**3%**
3%	1.63652E-07	\$ 6,110,768.59	1.594E-07	\$ 6,276,852.97	\$ 108,471.44	330.72	2%
4%	**1.63671E-07**	**\$ 6,110,058.59**	1.59094E-07	\$ 6,289,173.59	\$ 114,794.58	265.88	2%
5%	1.63572E-07	\$ 6,113,766.72	1.58742E-07	\$ 6,303,606.88	\$ 117,801.97	232.77	2%
6%	1.63611E-07	\$ 6,112,403.44	1.58448E-07	\$ 6,315,366.56	\$ 120,282.00	214.21	2%
7%	1.63524E-07	\$ 6,115,659.84	1.58248E-07	\$ 6,324,097.81	\$ 124,176.58	178.82	1%
8%	1.63524E-07	\$ 6,115,659.84	1.58248E-07	\$ 6,324,097.81	\$ 124,176.58	178.82	1%
9%	1.63466E-07	\$ 6,117,831.09	1.57935E-07	\$ 6,336,700.78	\$ 124,809.43	**174.04**	1%

However it is apparent that the tournament size has the inverse effect on the mean iterations to feasible as the larger the tournament size, the less iterations it takes for the algorithm to find a solution which satisfies the constraints of the problem. This behaviour is expected as when the tournament size is larger the weaker individuals in the population have a lesser chance of

being selected. This has a greater effect in the early stages of the search due to the hydraulic penalty having a heavy influence on the fitness value of hydraulically infeasible solutions.

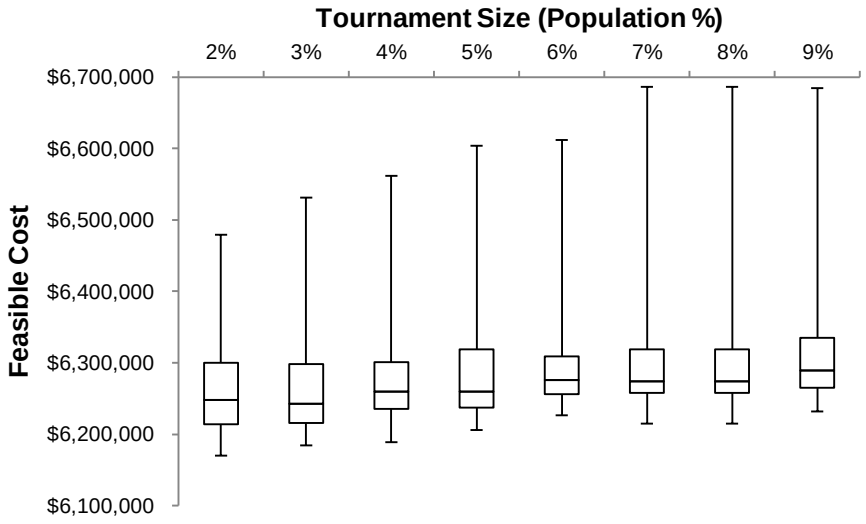


Figure 4-2. Box plot showing mean tournament size results for the Hanoi benchmark problem

The box plot above shows the feasible network cost for tournament size results in more detail and it is clear that there is more significance between the best performing tournament size (2%) and the worst (9%). Therefore it is proposed that tournament size does indeed significantly affect the performance of the SSGA for the Hanoi problem.

The following table presents the results for the mean mutation rate experiments for the Hanoi problem. This experiment tested different pipe mutation rates ranging from 0.029 to 0.235 in 0.029 intervals. These pipe mutation probabilities effectively equate to 1 to 8 pipe mutations for each execution of the mutation operator. The table shows that the SSGA performance is highest at a pipe mutation probability of 0.147 (~5 mutations per operator execution). This is the case for not only the mean best feasible fitness and cost but also the best feasible fitness and cost. The slightly lower mutation probability of 0.118 achieves the highest number of best known solution values closely followed by the mutation rate of 0.147.

Table 4-3. Mean mutation probability results for the Hanoi problem – SSGA parameter tuning

Mean Mutation Rate							
Mutation Rate	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
0.029	1.61476E-07	$ 6,193,254.22	1.52234E-07	$ 6,575,597.19	$ 199,225.19	228.41	0%
0.059	1.64046E-07	$ 6,095,890.31	1.58142E-07	$ 6,325,686.88	$ 117,623.28	**216.32**	1%
0.088	1.64344E-07	$ 6,084,804.84	1.59538E-07	$ 6,269,939.84	$ 105,510.89	223.41	3%
0.118	1.64388E-07	$ 6,083,188.75	1.60149E-07	$ 6,246,024.84	$ 104,582.60	230.84	**6%**
0.147	**1.64393E-07**	**$ 6,082,995.31**	**1.60684E-07**	**$ 6,225,125.63**	$ 102,081.24	245.52	4%
0.176	1.64059E-07	$ 6,095,393.28	1.60355E-07	$ 6,237,960.31	$ 103,957.04	259.53	0%
0.206	1.63415E-07	$ 6,119,417.97	1.59812E-07	$ 6,259,094.22	$ 104,110.79	298.07	0%
0.235	1.62503E-07	$ 6,153,785.16	1.58798E-07	$ 6,298,970.63	$ **100,615.70**	304.45	0%

The mean iterations taken by the algorithm to find a feasible solution appears to be affected by the mutation rate; generally increasing with greater mutation rates. This behaviour is somewhat expected: The higher the mutation rate, the greater the probability of the diameter of more pipes in the system being changed. This becomes a problem in the case of a water distribution system due to the hydraulic dependence junctions have on the flow of upstream pipes. If for example a pipe positioned reasonably close to a source is mutated to a much smaller diameter then the downstream junctions can potentially experience a significant hydraulic head drop and thus increasing the penalty cost of the resultant solution. Therefore the higher the mutation rate, the higher the probability that a solution will be made less hydraulically feasible. Also it is important to point out that although the mutation rate is shown to affect the iterations taken to find a feasible solution, it seems to have a lesser effect on the search of the algorithm to the feasible solution space compared to the tournament size.

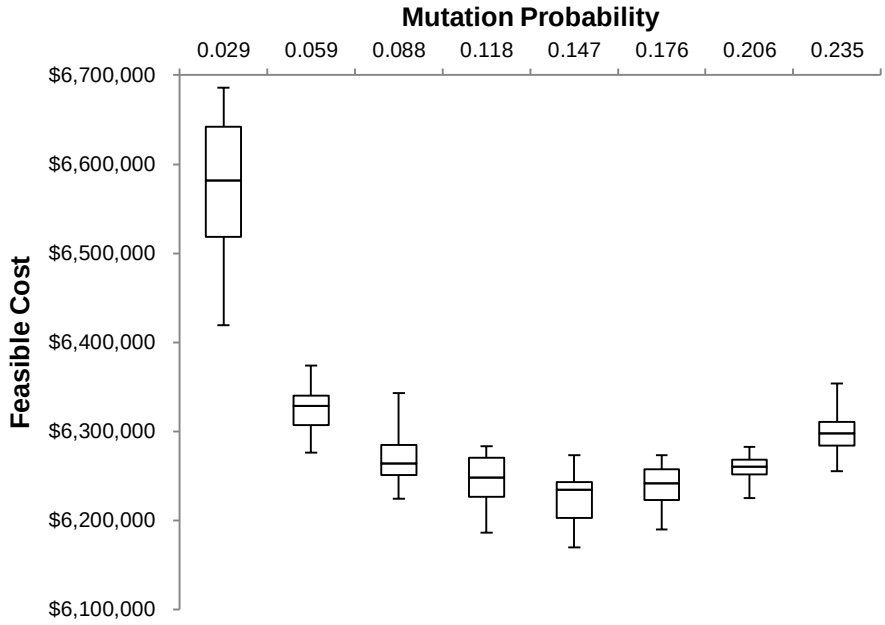


Figure 4-3. Box plot showing mean pipe mutation probability results for the Hanoi benchmark problem

The box plot from the mutation rate results is presented above. It is first apparent just how poor the performance of the algorithm is at the lowest pipe mutation probability of 0.029. When a mutation rate is too low there is a greater chance of premature convergence, where the algorithm search can get stuck in local optima. On the other hand if the mutation rate is too high the probability of searching a larger is area of the search space is increased, however, this can often prevent the algorithm converging on an optimal solution. The remainder of the mutation probabilities performed allot better than the first, displaying a relatively smooth curve highlighting the best performing pipe mutation probability at 0.147.

The following table presents the results for all of the 5 test problems, showing the best penalty factor cost, tournament size and pipe mutation probability for each. The best parameter values were chosen based on the mean best feasible cost results for each problem network. As stated previously, the detailed results for each problem can be seen in Appendix i.

Table 4-4. Best penalty cost, tournament size & mutation rate parameter values for each benchmark problem

Problem	Penalty Cost ($)	Tournament Size (% Population)	Mutation Rate (Pipe Mutation Probability)
Two Loop	10,000	2%	0.5
New York Tunnels	4,000,000	7%	0.143
Hanoi	500,000	2%	0.147
Foss Poly 1	16,000	2%	0.069
Modena	20,000	4%	0.132

With regard to the penalty factor, there is no clear correlation between the best penalty cost and the best known solution and overall complexity of the problem. Further experimentation and theorizing would be required to arrive at a viable method for predicting the optimal penalty factor for a SSGA, unfortunately this falls outside the scope of this work as the primary objective and focus of this section is to tune the SSGA as to provide a high quality performance benchmark for comparison purposes. Interestingly the results from some of the benchmark problems penalty cost experiments show that at a low penalty cost the performance of the algorithm is drastically reduced. This is the case for the New York Tunnels penalty cost experiment set (Figure 4-4). In a case where the penalty cost is too low, infeasible solutions are allowed to remain in the population resulting in a situation where feasible solutions have a lower fitness value to that of infeasible individuals. The result of this kind of behaviour can be observed in the figure below highlighted by the significant performance drop at a penalty cost of $1m.

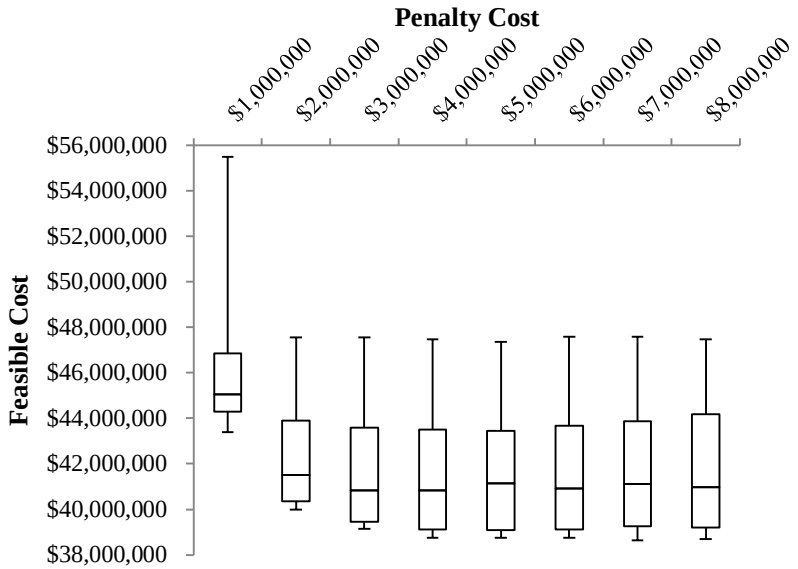


Figure 4-4. Box plot showing mean penalty cost results for the New York Tunnels benchmark problem

The optimal tournament size results range from 2% to 7% with 3 of the 5 network problems settling on 2%. The largest of the networks, the Modena problem has an optimal tournament size of 4% and the New York Tunnels network, a parallel expansion problem has the highest optimal tournament size of 7%. As discussed previously, the lower the tournament size the higher the probability that a more varied selection of solutions in the population are chosen for reproduction. With decreased selection pressure the algorithm has the opportunity to explore a larger proportion of the search space which in the case of the relatively small problems of Two Loop, Hanoi and Foss Poly 1 each with 8, 34 and 58 decision variables respectively improves the effectiveness of the overall search of the algorithm with the allotted number of objective function evaluations (200,000). These networks are also quite similar; each of which consist of multiple loops and a single water source. Comparing this to the larger problem of the Modena network which has a larger search space consisting of 317 decision variables and 4 separate water sources it is clear that a problem of this size requires a slightly higher selection pressure with a tournament size of 4%. In the case of the New York Tunnels problem the algorithm seems to require a relatively high selection pressure to achieve peak performance. This is thought to be partly due to the parallel expansion nature of the problem and the fact that hydraulically feasible solutions can be found in the initial population of every run. With no initial search for the feasible domain of the search space the

algorithm performs favourably with an increased selection pressure resulting in a more focused and iterative search. From these results it can be concluded that for smaller looped networks with hydraulically infeasible initial populations the algorithm benefits from a smaller tournament size. However as the size of the network increases so must the tournament size. From these results there is a case for varying the selection pressure as the algorithm progresses utilizing adaptive selection methods such as [143], although this falls outside the scope of this work.

The optimal pipe mutation probabilities range from 0.069 to 0.5 with the smallest test problem requiring the highest mutation rate for optimal performance. This is most likely due to the size of the problem; in the case of a relatively small search space the algorithm can afford to employ an aggressive search utilizing a higher mutation rate without risking non-convergence. The results suggest that as the problem complexity grows the optimal mutation rate falls. In the case of the larger problems such as Foss Poly 1 and Modena a slightly lower mutation rate is required to that of the smaller networks of New York Tunnels and Hanoi. This is due to the optimal balance between exploration and exploitation for each problem; searching the solution space whilst ensuring convergence. The higher the mutation rate the more the algorithm explores the search space; however the resultant population diversity can prevent the algorithm converging on an optimal solution.

4.1.2.1.1 Analysis of Variance and Interaction

An important factor to consider when assessing the sensitivity of an algorithm is whether there is an interaction between different parameters. Knowledge of how the different GA parameters interact and the resultant performance of the algorithm can aid in choosing appropriate parameter values when approaching new problems in the domain. Another important factor to consider is whether a change to a single parameter value has a meaningful impact on the performance of the algorithm.

This section aims to assess the level of variance between different parameter values and explore the interaction between parameters and the performance of the Single-objective Steady State Genetic Algorithm (SSGA) on

the least-cost water distribution network design problems being investigated. The variance and interaction between parameter values are investigated using the ANalysis Of VAriance (ANOVA) test, a widely used statistical technique primarily developed by R. A. Fisher [144] in the mid-1920s. The ANOVA test analyses the variation between the means of groups that have been split between independent variables (factors). In a case where there is more than one independent variable the ANOVA test can also be used to determine if there is an interaction between the independent variables and the dependent variable, in this case the parameters and solution quality respectively. The following set of results use the same data used to calculate the mean values reported earlier in this section.

4.1.2.1.1.1 Hanoi

In this section the variance and interaction between the three parameters (penalty cost, tournament size and pipe mutation rate) is tested on the Hanoi problem extensive run results where each parameter pair combination is tested with the view to understanding the impact each parameter has on its counterparts. Figure 4-5 shows the average best feasible costs obtained by the SSGA at varying tournament sizes and mutation rates from the best performing penalty cost ($500,000) results set for the Hanoi problem. It can be seen that as the tournament size is increased the effective mutation rate appears to narrow and increase slightly.

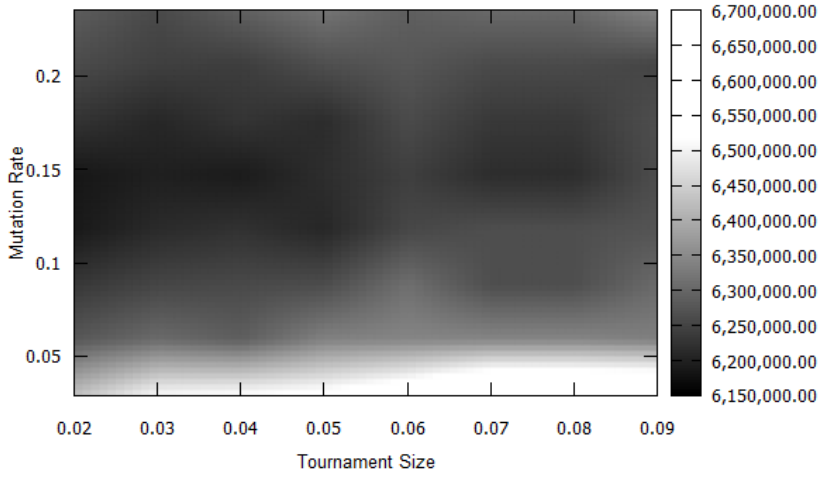


Figure 4-5 Heat map showing average best feasible cost between tournament size and mutation rate for the Hanoi problem

The table below shows the results from the two-way ANOVA test between tournament size and pipe mutation rate for all SSGA results from experiments using the best performing penalty cost of $500,000. Where *SS* is the sum of squares which is a measure of variation between factors, df is the degrees of freedom, MS is the mean squares which is SS / df, F is the F-ratio, P-value is the probability of significance and F_{crit} is the critical value based on the critical significance level, in this case α=0.05. It can be observed that both the tournament size and mutation rate have much higher F-values than their respective F critical values resulting in the rejection of the null hypothesis that all group means are equal. Therefore we can say for both tournament size and mutation rate that at least one group mean is different from another. Also from this test we can see that the interaction between the variables is statistically significant (P-value < 0.05) and therefore it can be stated that the relationship between the mutation rate and algorithm performance is dependent on the tournament size.

Table 4-5 Two-way analysis of variance (ANOVA) between tournament size & mutation rate for the Hanoi problem with $500k penalty cost

ANOVA - Hanoi (500k Penalty Cost)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Tournament Size	2.02E+12	7	2.88E+11	19.012	0.000	2.012
Mutation Rate	3.66E+13	7	5.22E+12	344.608	0.000	2.012
Interaction	2.06E+12	49	4.21E+10	2.778	0.000	1.358
Within	4.75E+13	3,136	1.52E+10			
Total	8.82E+13	3,199.00				

Figure 4-6 shows the average best feasible cost between the penalty cost and the tournament size. From this figure it can be suggested that as the penalty cost is increased the effective range of the tournament size is decreased resulting in the only lower tournament sizes being effective at higher penalty costs.

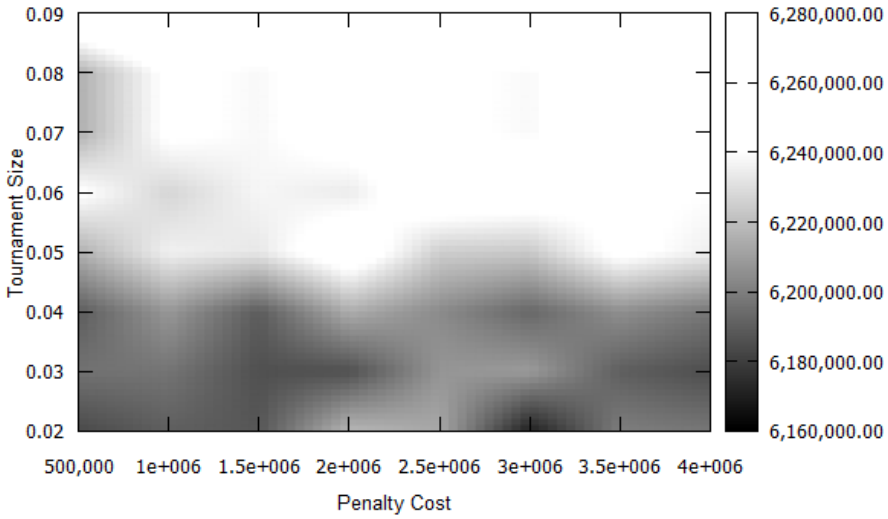


Figure 4-6 Heat map showing average best feasible cost between penalty cost and tournament size for the Hanoi problem

The results from the ANOVA test between the tournament size and the penalty cost is reported below at the best performing pipe mutation rate (P0.147). It can be seen that the tournament size has a statistically significant effect on the performance of the algorithm at this fixed mutation rate, however the P-value (0.17) for the penalty cost thus it can be stated that the penalty cost has no significant effect on algorithm performance. It can also be observed that there is no significant interaction between the two tested parameters (interaction P-value of 1.00) therefore it can be said that the interaction between tournament size and algorithm performance is not dependant on penalty cost.

Table 4-6 Two-way analysis of variance (ANOVA) between tournament size & penalty cost for the Hanoi problem with P0.147 pipe mutation

ANOVA - Hanoi (0.147 Mutation Rate)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Tournament Size	1.82E+12	7	2.60E+11	24.823	0.000	2.012
Penalty Cost	1.08E+11	7	1.55E+10	1.479	0.170	2.012
Interaction	2.07E+11	49	4.23E+09	0.404	1.000	1.358
Within	3.28E+13	3136	1.05E+10			
Total	3.50E+13	3199				

Figure 4-7 details the mean best feasible cost between the penalty cost and the mutation rate. The heat map does not show any clear relationship between the two parameters with the algorithm performing similarly at constant mutation rates at all penalty cost values and vice versa.

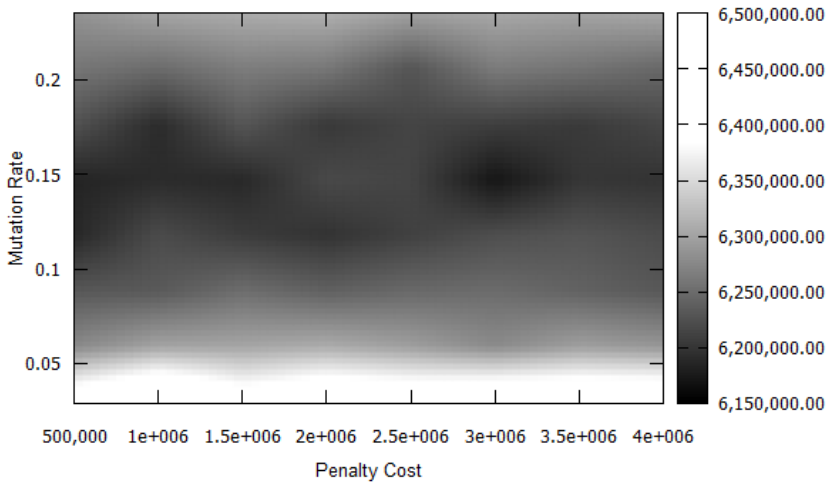


Figure 4-7 Heat map showing average best feasible cost between penalty cost and mutation rate for the Hanoi problem

The table below show the results from the ANOVA test conducted on the SSGA best feasible cost results at the best performing tournament size (0.02N). As with the pervious ANOVA test the penalty cost is not statistically associated with algorithm performance at this tournament size. However algorithm performance is dependent on variations in pipe mutation rate. In addition it can be stated that there is a lack of statistically significant interaction between the two parameters, which indicates that the relationship between mutation rate and algorithm performance is not dependent on penalty cost.

Table 4-7 Two-way analysis of variance (ANOVA) between penalty cost & mutation rate for the Hanoi problem with 0.02N tournament size

ANOVA - Hanoi (0.02 Tournament Size)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Penalty Cost	5.12E+10	7	7.32E+09	0.658	0.708	2.012
Mutation Rate	1.87E+13	7	2.66E+12	239.637	0.000	2.012
Interaction	4.15E+11	49	8.48E+09	0.762	0.887	1.358
Within	3.49E+13	3136	1.11E+10			
Total	5.40E+13	3199				

It can be said based on these results that the penalty cost does not significantly impact the performance of the algorithm, at least for the penalty values tested. The algorithm performance is much more sensitive to changes in tournament size and mutation rate. ANOVA tests conducted on the remaining problems resulted in similar outcomes; the penalty cost did not have a statistically significant impact of the performance of the SSGA for any of the other problems tested in this section. Therefore the penalty cost is not considered for the remainder the problems tested in this section.

4.1.2.1.1.2 Two Loop

The average best feasible cost between the tournament size and mutation rate for the Two Loop problem is displayed in Figure 4-8. It can be observed that as the tournament size is increased so does the effective range of the pipe mutation rate parameter. This suggests that as selection pressure increases the algorithm will perform better at higher mutation rates, this is somewhat expected because a high selection pressure increases solution exploitation and a high mutation rate encourages exploration. These results seem to support the assumption that algorithm performance is linked to the balance between exploration and exploitation of the solution space.

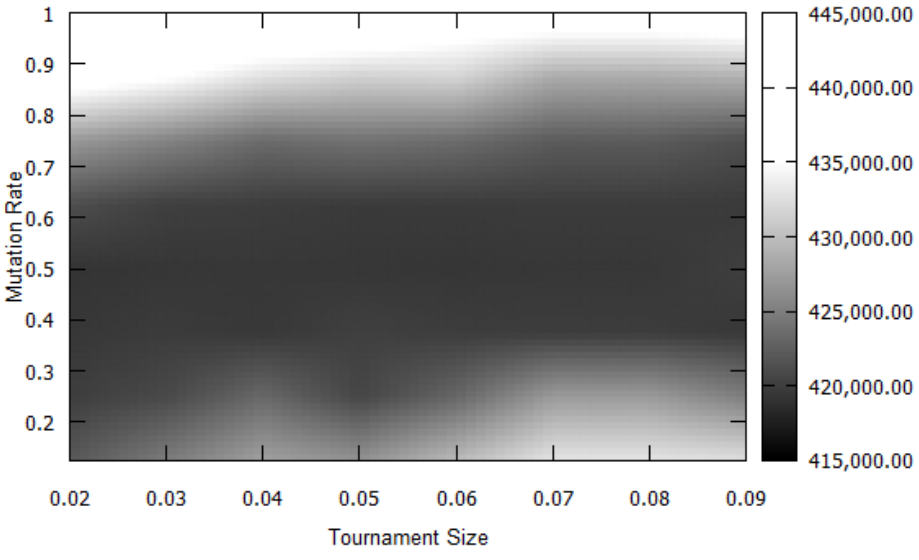


Figure 4-8 Heat map showing average best feasible cost between tournament size and mutation rate for the Two Loop problem

The two-way ANOVA shows that in the case of the two loop problem the tournament size has no statistically significant impact on the performance of the algorithm, however, mutation rate does have an impact on performance. It can also be said that the relationship between algorithm performance and mutation rate depends on the tournament size. This can be seen in Figure 4-8 which suggests that as tournament size is increased the best performing mutation rate also increases.

Table 4-8 Two-way analysis of variance (ANOVA) between tournament size & mutation rate for the Two Loop problem with $10k penalty cost

ANOVA - Two Loop (10k Penalty Cost)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Tournament Size	2.07E+08	7	2.96E+07	0.497	0.837	2.012
Mutation Rate	1.41E+11	7	2.02E+10	338.779	0.000	2.012
Interaction	1.48E+10	49	3.01E+08	5.058	0.000	1.358
Within	1.87E+11	3136	5.95E+07			
Total	3.43E+11	3199				

4.1.2.1.1.3 New York Tunnels

Figure 4-9 displays the average best feasible cost between the tournament size and mutation rates for the New York Tunnels problem. It can be seen that pipe mutation sweet spot lies between a rate of approximately P0.1 & P0.2 for all tournament sizes, however there is still a slight observable trend suggesting that as tournament size is increased so does the optimal pipe mutation rate.

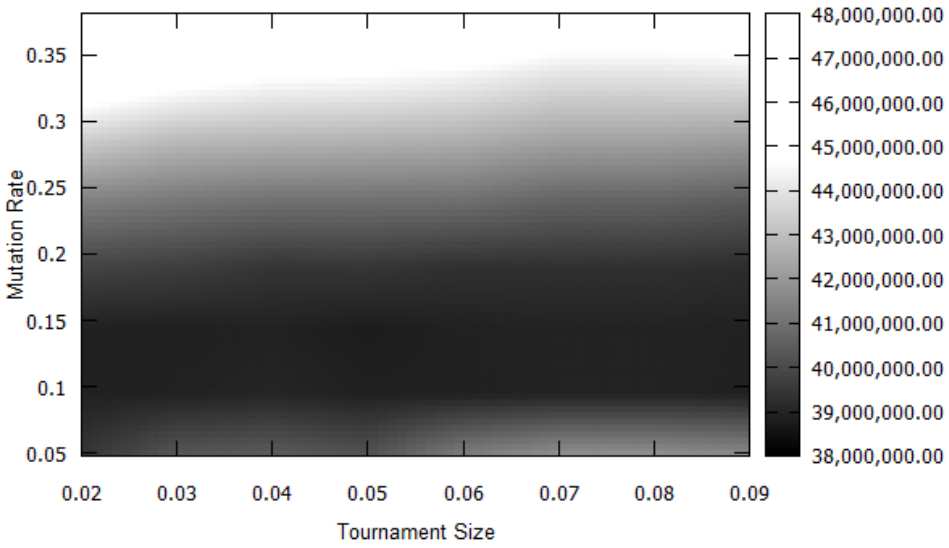


Figure 4-9 Heat map showing average best feasible cost between tournament size and mutation rate for the New York Tunnels problem

The ANOVA test results between the tournament size and mutation rate for the New York Tunnels problem are shown in Table 4-9. It can be observed that the P-values for both the tournament size and mutation rate indicate an association between parameters and algorithm performance. The interaction between both parameters is also found to be statistically significant indicating a relationship between the tournament size and algorithm performance is dependant of mutation rate and vice versa.

Table 4-9 Two-way analysis of variance (ANOVA) between tournament size & mutation rate for the New York Tunnels problem with $4000k penalty cost

ANOVA - New York Tunnels (4000k Penalty Cost)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Tournament Size	7.41E+13	7	1.06E+13	6.56	0.00	2.012
Mutation Rate	2.15E+16	7	3.08E+15	1908.17	0.00	2.012
Interaction	6.05E+14	49	1.23E+13	7.65	0.00	1.358
Within	5.06E+15	3136	1.61E+12			
Total	2.72865E+16	3199				

4.1.2.1.1.4Foss Poly 1

Figure 4-10 displays the average best cost between the tournament size and mutation rate for the Foss Poly 1 problem. It can be observed that the best performing combination of parameter values occur where both tournament size and mutation rate are lower. However there is an observable trend where, as tournament size increases the viable range of mutation rates also increases slightly.

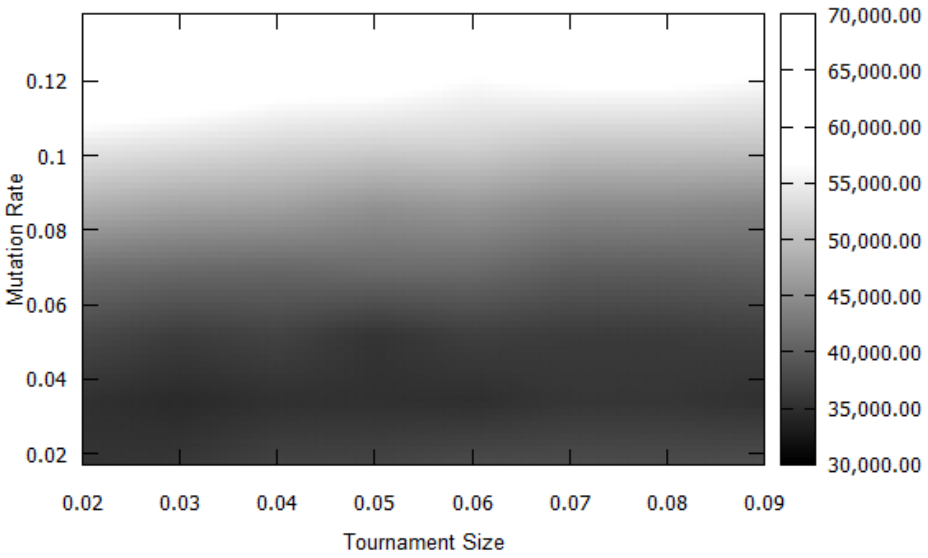


Figure 4-10 Heat map showing average best feasible cost between tournament size and mutation rate for the Foss Poly 1 problem

The ANOVA test for the Foss Poly 1 problem at a penalty cost of $16,000 between tournament size and mutation rate can be found in Table 4-10. Both the tournament size and mutation rate significantly impact the performance of the SSGA for this problem. In addition, there is a significant variation between group means for the mutation rate which is indicated by the large F-value (3221.043), much higher than the tournament size variation, suggesting that for this problem the mutation rate has a much higher impact on performance than the tournament. The test also shows that there is a statistically significant interaction between the two parameter values implying that the relationship between the mutation rate and SSGA performance is governed by the tournament size.

Table 4-10 Two-way analysis of variance (ANOVA) between tournament size & mutation rate for the Foss Poly 1 problem with $16k penalty cost

ANOVA - Foss Poly 1 (16k Penalty Cost)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Tournament Size	1.65E+09	7	2.35E+08	14.739	0.00	2.012
Mutation Rate	3.60E+11	7	5.14E+10	3221.043	0.00	2.012
Interaction	3.13E+09	49	6.40E+07	4.009	0.00	1.358
Within	5.00E+10	3136	1.60E+07			
Total	4.15E+11	3199				

4.1.2.1.1.5 Modena

Figure 4-11 shows the average best feasible solution cost between the tournament size and pipe mutation rate for the Modena problem. Only a small amount of variation in optimal mutation rate can be observed with a change in tournament size with the figure suggesting only a slight increase in effective mutation rate with an increase in tournament size.

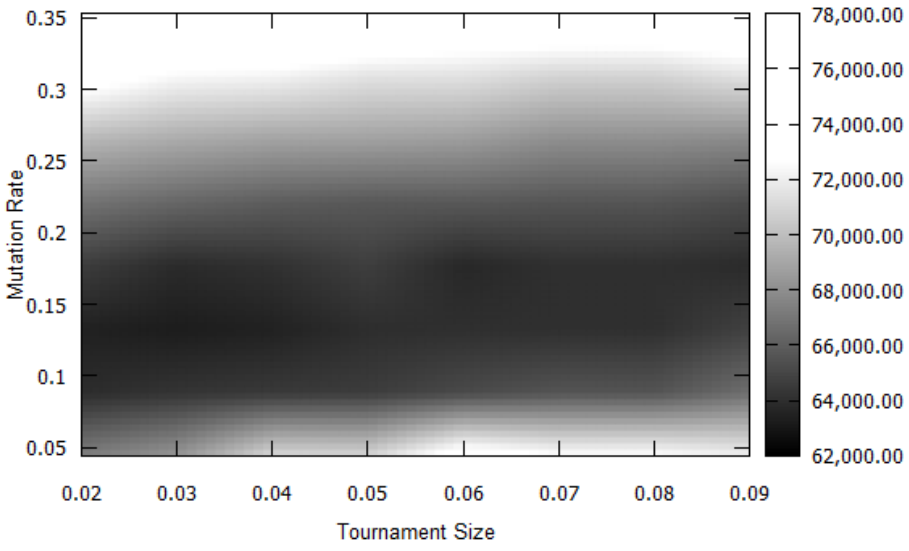


Figure 4-11 Heat map showing average best feasible cost between tournament size and mutation rate for the Modena problem

The observations of the heat map are supported by the results from the ANOVA test where it is seen that the tournament size is not significantly associated with the performance of the algorithm whereas the mutation rate

produces a statistically significant variance between the group means. This is an interesting finding as tournament size has had a significant association with the performance of the algorithm for every other problem on test. The lack of selection pressure sensitivity could be due to configuration of the problem; a feasible solution is easily found partially due to the four water sources which may in turn shape the fitness landscape in such a way that selection pressure has a minimal effect on the search of the algorithm.

Table 4-11 Two-way analysis of variance (ANOVA) between tournament size & mutation rate for the Modena problem with $20k penalty cost

ANOVA - Modena (20k Penalty Cost)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Tournament Size	5.68E+07	7	8.11E+06	0.548	0.799	2.012
Mutation Rate	5.05E+10	7	7.21E+09	486.650	0.000	2.012
Interaction	2.91E+09	49	5.95E+07	4.012	0.000	1.358
Within	4.65E+10	3136	1.48E+07			
Total	9.99E+10	3199				

4.1.2.2 Replacement Strategy

This section details the results from the replacement strategy experimentation for each of the test problems. The first part of this section shows the detailed replacement strategy results for the Hanoi problem. For this experimentation the optimal parameter values for the Hanoi problem were used; penalty cost of $500,000, tournament size of 2% and a pipe mutation rate of 0.147. The table below shows the mean best feasible network cost and standard deviation for each of the replacement strategies. From initial inspection there is a clear difference of performance between the oldest and random unconditional replacement methods and the rest of the strategies. From the table of figures it is apparent that conditional replacement has a large effect of the performance of the algorithm when paired with all of the individual selection methods (worst, oldest and random). While the oldest and random unconditional strategies perform drastically worse that of their conditional counterparts, the worst unconditional method interestingly performs well.

Table 4-12. Mean replacement strategy results for the Hanoi problem – SSGA parameter tuning

	Mean Best Feasible Cost		Mean Best Feasible Cost SD	
Worst / Conditional	**$**	**6,183,407.60**	$	99,800.06
Oldest / Conditional	$	6,195,220.00	$	**90,184.37**
Random / Conditional	$	6,206,049.80	$	104,525.92
Worst / Unconditional	$	6,205,100.00	$	95,292.54
Oldest / Unconditional	$	7,520,530.00	$	118,098.27
Random / Unconditional	$	7,486,498.20	$	153,245.31

The results show that the oldest unconditional replacement strategy performs the poorest out of all of the tested replacement methods. The poor performance of this replacement approach could be due to the fact that individuals in the population do not survive past 50 fitness evaluations due to the unconditional replacement strategy. The random unconditional method fares slightly better, although again due to the unconditional replacement approach there is a good chance that high performing solutions in the population can be replaced with poor solutions.

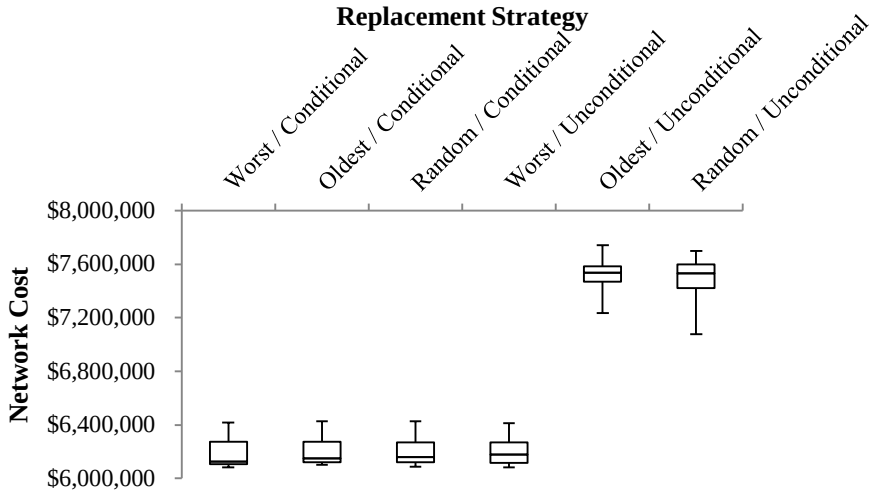


Figure 4-12 Box plot showing replacement strategy results for the Hanoi problem – SSGA parameter tuning

From these results it is clear that conditional replacement consistently performs well regardless of the method used to select the individual being replaced. This behaviour is somewhat expected as it ensures the overall fitness

of the population is never decreased unlike unconditional replacement where the population fitness can potentially decrease with the addition of newly created poor quality individuals.

The following table shows the best performing results for all of the small to medium network problem benchmarks based of mean best feasible network cost. For each experiment the best know parameters values were used based on the results in section 4.1.2.1. The detailed results for each benchmark presented in this section are available in the appendix.

Table 4-13. Best replacement strategy parameters for each benchmark problem

Problem	Individual Replaced	Replacement Condition
Two Loop	Worst	Conditional
New York Tunnels	Worst	Unconditional
Hanoi	Worst	Conditional
Foss Poly 1	Worst	Unconditional
Modena	Worst	Unconditional

From these results it is clear that replacement of the worst individual in the population results in the best performance of the algorithm for all benchmark problems. The smallest test problem, Two Loop and the Hanoi problem both perform best using a conditional replacement strategy. What is interesting however is that three of the five test problems perform better utilizing an unconditional replacement strategy. This is mainly the case for the larger problems (Foss Poly 1 & Modena) but also includes the New York Tunnels parallel expansion problem. Using a unconditional replacement strategy will tend to increase the variety of solutions in the population allowing the algorithm to potentially conduct a wider search of the solution space which in the case of the larger test problems has proven to be beneficial. It is important to point out that the unconditional replacement strategy is heavily reliant on the replacement of the worst individual in the population as this helps prevent the population from diverging from the optimal solution space and becoming unstable.

4.1.3 Conclusion

This section set out to explore the effect that various parameters have on the performance of the SSGA with the view to not only observe the behaviour of the algorithm and how various parameter configurations affect the algorithm but also ensure the parameters of the SSGA are optimally tuned to each benchmark problem with the aim to provide a competitive baseline for later comparison with the heuristic based GAs.

The first batch of experiments concentrated on the effect penalty cost has on the performance of a SSGA with a static penalty. Although the presence of a penalty cost is essential in the case of most single objective problems to ensure the constraints of the problem are satisfied, it is apparent that, at least with the penalty cost ranges utilized for these tests that the penalty cost has little to no significant effect on the performance of the algorithm except for very low penalty costs. Further testing on a larger range of penalty costs would be required to better understand the relationship between penalty cost and the performance of the SSGA, although conducting a more extensive study of this type falls outside the reach of this work. The second set of experiments explored the effect tournament size has on the performance of the algorithm. From the results gained there is a potential correlation between problem complexity and tournament size; the larger the problem the greater the optimal tournament size. The third set of experiments investigated the effect mutation rate has on the performance of the algorithm and found that as the problem complexity grows the optimal mutation rate generally falls. The final set of experiments focused on the replacement method of the SSGA. A range of different replacement strategies were tested on the benchmark problems and it was observed that the algorithm performed best when the worst individual in the population was selected for replacement. It was also observed that the use of conditional replacement worked best for the smaller problems from the problem set and an unconditional replacement approach performed best for the larger problems. The knowledge derived from this section can be used to tune the performance of the GA for use in experiments with the heuristic variants of EAs described in the later sections.

4.2 Locally Constrained Genetic Algorithm

This section describes the formulation and development of a heuristic based SSGA for the least-cost design of water distribution networks. The motivation behind the development of this algorithm was to see if the search of the algorithm could be softly constrained in the local space utilizing water system engineering knowledge normally applied during the classical design process. The core idea behind the formulation of such an algorithm was to target and eliminate hydraulically infeasible elements within a water network solution whilst retaining algorithm performance with the view of obtaining good quality feasible solutions faster than a highly tuned SSGA. The Locally Constrained Genetic Algorithm (LCO-GA) presented here applies the hydraulic deficit heuristic detailed in section 3.4.1 to the mutation operator of the Steady State GA (SSGA) presented in section 3.1. The aim of the hydraulic deficit heuristic is to target pipes within the network that are causing junctions to be in head deficit. This is achieved through its application to the mutation operator of the SSGA.

The Heuristic based Mutation Operation (HMO) is designed to allow the EA to locate feasible network designs earlier in the optimisation. It can be configured for use with any appropriate constraints, but here the application to network hydraulic performance only is considered. LCO-GA employs the hydraulic deficit heuristic that is designed to primarily target pipes which are causing head constraint violations. At initialisation, the algorithm runs a single hydraulic simulation of the WDN and logs the directional flow information of each pipe. Using this data, each junction and its immediate upstream pipe and junction are logged making it possible to identify pipes that are limiting junction head down-stream. The HMO first selects a junction through the use of a roulette wheel method which assigns wheel segment sizes using head deficit information obtained during the fitness evaluation of the solution, resulting in junctions with a high pressure head deficit having a greater probability of being selected. Once a junction is selected the heuristic searches upstream of the selected junction until a junction is found which is in pressure head excess. The pipe immediately downstream of the discovered junction is then mutated to a greater available diameter. If the pressure head at every junction satisfies the

problem constraints, the HMO employs a slightly different method. The roulette wheel method is employed again; however, wheel segment size and therefore junction selection probability is now proportional to junction pressure head excess. This results in junctions with high pressure head excess having a greater probability of selection. Once selected the pipe immediately upstream of the initially selected junction is mutated to a smaller size. As stated previously due to the dependency of the HMO on a solution's pressure head information, mutation cannot be applied post crossover without the need to re-evaluate the hydraulic network of resultant solutions.

4.2.1 Experimental Setup

During early experimentation with LCO-GA it was found that if the HMO was employed exclusively (i.e., without random pipe mutation) throughout the evolutionary process, the population would become stagnant and prematurely converge on a sub optimal solution. This lead to the idea of combining heuristic based mutation and random pipe mutation (employed by the SSGA) to help prevent premature convergence. Therefore this first set of experiments explores the performance of the algorithm with varying applications of the HMO vs. standard pipe mutation. The probability of heuristic based mutation being applied for each pipe selected for mutation was set at a constant 25%, 50%, 75% and 100%. In the case where the heuristic is not selected for use the mutation operator employs the standard pipe mutation method as used by the SSGA resulting in a constant number of mutations regardless of the mutation type being used.

These initial experiments were conducted on the Hanoi problem and the optimal parameter values obtained from the single-objective genetic algorithm parameter tuning section of this chapter were used for all instances of the SSGA and LCO-GA in this set of experiments. Each configuration of LCO-GA and the SSGA were run 50 times, each for 200,000 objective function evaluations. To ensure a fair comparison between the SSGA and the different LCO-GA configurations a list of 50 randomly generated integers were generated for the purpose of seeding the random number function at the beginning of each of the 50 runs. The same list of random number seeds were used for each algorithm configuration experiment ensuring the same starting populations for

each run, for example, LCO-GA 1st run would have the same starting population as the 1st run of the SSGA. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size of 100 individuals, single-point crossover, penalty cost of $500,000, tournament size of 2% and a pipe mutation rate of 0.147. Conditional worst individual replacement strategy was used as per the previous experimentation. As with the SSGA, LCO-GA was implemented in C++ and all of the following experiments were run on an Intel Core i7-4770K 3.5GHz PC.

4.2.2 Results

This section presents the results for the initial LCO-GA experiments and compares them to the performance of the SSGA. The table below shows the best results obtained for each algorithm from the best performing run for the Hanoi problem. Both the best fitness and best feasible network cost are presented along with the least number of fitness evaluations taken for that algorithm to reach a feasible solution. From these results it can be observed that the SSGA and LCO-GA with 25% and 50% heuristic mutation probability achieved the best known feasible solution for the Hanoi problem ($6,081,220) within the allotted 200,000 fitness evaluations. LCO-GA with 75% heuristic mutation probability performs slightly worse than the previously mentioned algorithms however it is the version of the algorithm with solely heuristic mutation (LCO-GA (100%)) that performs drastically worse than the other algorithms. Looking at the number of fitness evaluations it takes the algorithms to achieve a hydraulically feasible solution shows that the more the heuristic mutation operator is applied the faster the algorithm finds the feasible solution space, taking LCO-GA (100%) only 4 fitness evaluations to achieve a feasible solution compared to the SSGA which took 64.

Table 4-14. LCO-GA best single run results for the Hanoi problem - SSGA & LCO-GA variants comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**1.644E-07**	**6,081,220**	64
LCO-GA (25%)	**1.644E-07**	**6,081,220**	36
LCO-GA (50%)	**1.644E-07**	**6,081,220**	20
LCO-GA (75%)	1.640E-07	6,097,450	10
LCO-GA (100%)	1.609E-07	6,224,050	**4**

The following table presents the mean results from the best solutions from the 50 runs of each algorithm variant. Included in these results is the best average fitness, feasible cost and evaluations taken to achieve a feasible solution each accompanied by the standard deviation for the best results from the 50 runs. From these results it can be seen that the SSGA outperforms the LCO-GA variants in terms of average best solution fitness and network cost after 200,000 fitness evaluations. We also see that as the proportion of heuristic mutation is increased the average solution quality decreases. However as with the previous best single run results, increasing the application of heuristic mutation allows the algorithm to find the feasible solution space in less fitness evaluations; where LCO-GA (100%) takes on average 68.3 fitness evaluation to find a feasible solution compared to the SSGA which take on average 415.2 evaluations.

Table 4-15. LCO-GA 50 run best solution average results for the Hanoi problem - SSGA & LCO-GA variants comparison

	50 Run Best Solution Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	**1.618E-07**	**2.571E-09**	**6,183,410**	**99,800**	415.2	152.3
LCO-GA (25%)	1.614E-07	2.648E-09	6,197,340	102,431	164.2	61.6
LCO-GA (50%)	1.603E-07	2.634E-09	6,241,470	102,754	110.4	33.8
LCO-GA (75%)	1.599E-07	2.965E-09	6,255,600	116,537	86.5	33.9
LCO-GA (100%)	1.548E-07	3.342E-09	6,463,910	141,243	**68.3**	**29.3**

Figure 4-13 presents the average fitness from the best solution in each run over the 200,000 fitness evaluations for the Hanoi problem. From this graph it can be observed that all variants of the LCO-GA exhibit better performance in the early stages of the search compared to the SSGA. However the LCO-GA variants with very high applications of the heuristic mutation do not perform very well in the later stages of the search. For example, LCO-GA (100%) can be seen to rapidly converge on a sub-optimal solution at approximately 30,000 evaluations and does not improve significantly throughout the remaining ~170,000 evaluations. As the application of the heuristic mutation is decreased the final solution quality of the algorithm is increased all be it with a slight decrease in the early stage performance of the search. The best performing LCO-GA variant is LCO-GA (25%) which achieves better solutions than the SSGA up until approximately 140,000 solution evaluations where the SSGA slightly out performs LCO-GA (25%) in the remainder of the search.

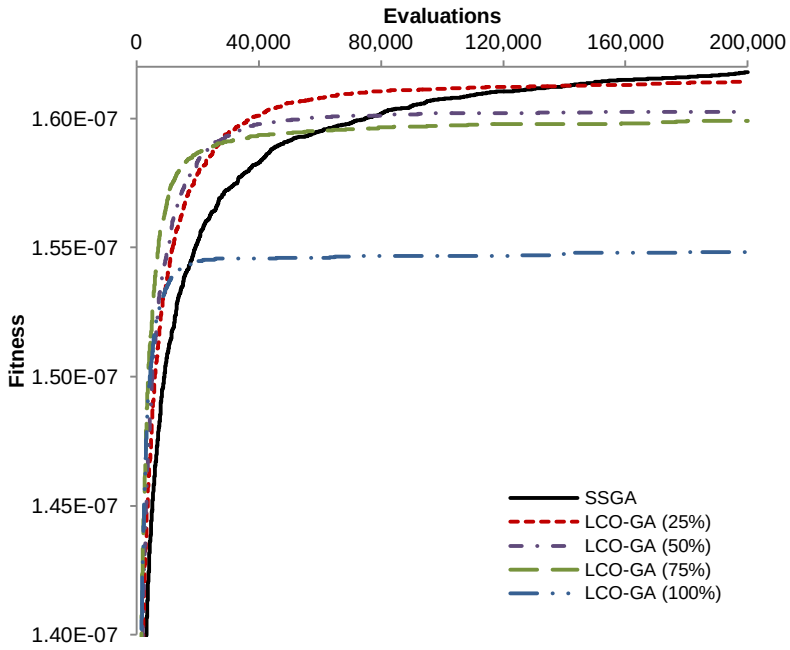


Figure 4-13. Graph showing the average best solution fitness over evaluations for the Hanoi problem - SSGA & LCO-GA variants comparison

Figure 4-14 shows the average percentage of feasible solutions present in the population during the search. The first plot (left) spans the entire 200,000 fitness evaluations and the second (right) gives a detailed look at the first 2,000 evaluations. It can be seen from this figure that LCO-GA (100%) promotes initial population feasibility more than the other algorithms, however following the first

few thousand evaluations the feasibility of the population drops from approximately 90% to 80% within 25,000 evaluations. This behaviour can be attributed to the heuristic which in the case of a hydraulically feasible solution the HMO targets areas of hydraulic head excess and starts to constrict pipe diameters with a view to improving infrastructure cost. However aggressive use of this component of the heuristic seems to decrease population feasibility without having the desired effect on the overall quality of the solutions.

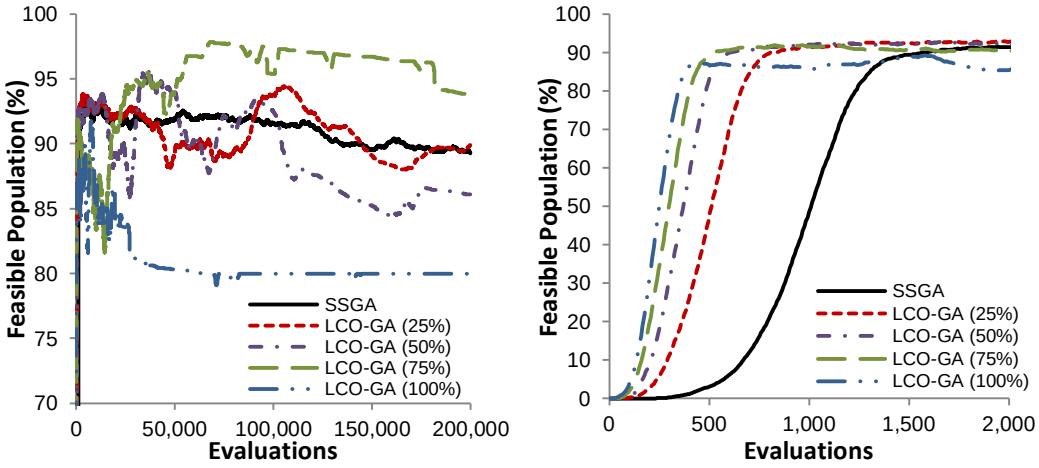


Figure 4-14. Graphs showing average percentage of feasible solutions present in the population for the Hanoi problem – SSGA & LCO-GA variants comparison

LCO-GA (75%) achieves a population feasibility of approximately 92% after around 400 evaluations and after an initial drop in population feasibility in the first ~20,000 evaluations increases sharply and sustains a very high feasibility level mostly staying above 95% population feasibility for the remainder of the search. This behaviour is quite interesting as one would expect a lower number of feasible solutions to be present in the population during the later stages of the search due to the heavy (75%) application of the heuristic which actively attempts to reduce excess feasibility (excess head) in a solution. However it appears that in combination with the standard random pipe mutation the algorithm performs well in terms of solution feasibility. It emerges that as the probability of the application of the heuristic is decreased the more evaluations the algorithm needs to achieve a feasible solution and hence it takes longer to accumulate hydraulically feasible solutions in the population. This is enforced further by the results for LCO-GA (50%) and LCO-GA (25%); both of which find the feasible search space much faster than the SSGA however as the influence

of the heuristic mutation is decreased the evaluations needed to find a feasible solution and achieve a ~90% population feasibility is increased. Both of the aforementioned LCO-GA configurations are seen to settle in the area of population feasibility between ~85% - 90% for the duration of the run of the algorithm, although both do fluctuate more compared with the SSGA.

4.2.3 Conclusion

This section presented the results from the initial Locally Constrained Genetic Algorithm (LCO-GA) experiment set on the Hanoi problem with the view of investigating how the application strength of the Heuristic Mutation Operator (HMO) impacts the performance of the algorithm both in terms of solution fitness but also solution feasibility. It was found that the higher the probability of heuristic based mutation the faster the algorithm found the feasible search space thus increasing the initial convergence of the algorithm. However at these high application rates of the heuristic based mutation the overall performance of the algorithm was diminished in the later stages of the search compared with the standard Steady State Genetic Algorithm (SSGA) often prematurely converging on a sub optimal solution. The LCO-GA configuration found to perform best was LCO-GA (25%) (25% probability of applying the heuristic during mutation). LCO-GA (25%) displayed faster convergence over that of the SSGA producing higher fitness solutions than the SSGA up until approximately 130,000 solution evaluations where the SSGA is able to find slightly better solutions in the remainder of the search. One of the key performance behaviours of LCO-GA is the algorithm's ability to find the feasible solution space in less solution evaluations than the SSGA, this trait is very desirable from a design engineer's point of view, especially in the case where time and hence solution evaluations are limited. For relatively small networks such as the Hanoi network this is not a big consideration as a solution evaluation can be performed in less than a millisecond on a high-end consumer grade CPU. However, when tackling a much larger, more complex real world problem solution evolution compute time increases dramatically to over 35 milliseconds (Network B). This by itself does not seem very excessive until you consider the fact that a single run of the algorithm could require a few hundred thousand solution evaluations to achieve convergence on a near optimal solution. For

example a single run on the real world Network B problem with 200,000 solution evaluations would take approximately 2 hours to compute. This of course does not factor in the compute time to execute the remainder of the algorithm although this is very negligible compared with the fitness evaluation CPU time. An additional consideration on this point is that these experiments are only conducting single period simulations, however a great deal of WDN design problems require extended period simulation where, for example, the daily demand cycle has to be taken into account commonly resulting in the network having to be solved for each hour of a 24 hour period which greatly increases the amount of CPU time needed to fully evaluate a single solution.

It is clear from this set of experiments that the hydraulic heuristic based mutation aids the search of the algorithm in the initial stages of the search in terms of fitness value convergence rate and also the number of solution evaluations needed to find the feasible solution space. However as has been observed, the sole use of the heuristic based mutation hinders the search of the algorithm in the later stages of the search often resulting in premature convergence and hence suboptimal solutions; requiring the standard pipe mutation to improve the solution quality in the later stages of the search.

4.3 Adaptive Locally Constrained Genetic Algorithm

It has been observed that strong applications of the heuristic mutation operator (HMO) are very effective in the early stages of the algorithm run, finding the feasible solution space quickly. However the random pipe mutation operator was found to be more effective in the latter generations of the algorithm. Therefore, it was necessary to employ a device to influence the usage of the HMO. The Fitness Gradient Monitor (FGM) controls the probability that the HMO is applied to the current generation's child solutions, through the monitoring of the population's current best solution's fitness. The probability of the application of the HMO is decreased from 1 as the fitness curve of the best solution's fitness tends to zero.

$$P_{gm} = \frac{g_c}{g_i} \qquad (24)$$

Where g_i is the initial gradient, g_c is the current gradient and P_{gm} is the probability of the HMO being employed. If HMO is not utilised then the random pipe mutation operator is used instead. This method ensures a smooth transition between the use of HMO and the random pipe mutation operator as the algorithm's search progresses. This additional process ensures that the HMO is applied aggressively at the start of the algorithm's search, improving solution feasibility, but is able to smoothly reduce the influence of the hydraulic heuristic as the search progresses and the rate of solution fitness increase reduces. The resultant algorithm will be known as Adaptive Locally Constrained Genetic Algorithm (ALCO-GA).

4.3.1 Experimental Setup

This set of experiments explores the performance of the Adaptive Locally Constrained Genetic Algorithm (ALCO-GA) with varying applications of the adaptive HMO vs. standard pipe mutation. The probability of the adaptive heuristic based mutation being applied for each pipe selected for mutation was set at a constant 25%, 50%, 75% and 100%. In the case where the heuristic is not selected for use, either due to the FGM or the applied constant probability the mutation operator employs the standard pipe mutation method as used by the SSGA resulting in a constant number of mutations regardless of the mutation type being used.

As with the previous LCO-GA experiments this set of experiments were conducted on the Hanoi problem and the optimal parameter values obtained from the single-objective genetic algorithm parameter tuning section of this chapter were used for all instances of the SSGA and ALCO-GA in this set of experiments. Each configuration of ALCO-GA and the SSGA were run 50 times, each for 200,000 objective function evaluations. To ensure a fair comparison between the SSGA and the different ALCO-GA configurations a list of 50 randomly generated integers were generated for the purpose of seeding the random number function at the beginning of each of the 50 runs. The same list of random number seeds were used for each algorithm configuration experiment ensuring the same starting populations for each run. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size of 100 individuals, single-point crossover, penalty

cost of $500,000, tournament size of 2% and a pipe mutation rate of 0.147. Conditional worst individual replacement strategy was used as per the previous experimentation. As with the SSGA, ALCO-GA was implemented in C++ and all of the following experiments were run on an Intel Core i7-4770K 3.5GHz PC.

4.3.2 Results

Table 4-16 presents the best results obtained for each algorithm from the best performing run for the Hanoi problem. Both the best fitness and best feasible network cost are presented along with the least number of fitness evaluations taken for that algorithm to reach a feasible solution. It can be seen that all of the algorithms find the best known solution for the Hanoi problem ($6,081,220) however all of the ALCO-GA variants find a feasible solution in less than half the evaluations it takes the SSGA to do, with the ALCO-GA (50%) variant performing the best in this area.

Table 4-16. ALCO-GA best single run results for the Hanoi problem

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**1.644E-07**	**6,081,220**	64
ALCO-GA (25%)	**1.644E-07**	**6,081,220**	30
ALCO-GA (50%)	**1.644E-07**	**6,081,220**	**22**
ALCO-GA (75%)	**1.644E-07**	**6,081,220**	24
ALCO-GA (100%)	**1.644E-07**	**6,081,220**	24

Table 4-17 presents the mean results from the best solutions from the 50 runs of each algorithm variant. Included in these results is the best average fitness, feasible cost and evaluations taken to achieve a feasible solution each accompanied by the standard deviation for the best results from the 50 runs. From these results it can be seen that the SSGA outperforms the ALCO-GA variants in terms of fitness and feasible solution cost however all ALCO-GA variants find the feasible solution space in less fitness evaluations than the SSGA. The best performing ALCO-GA variant is ALCO-GA (100%) achieving a mean best feasible cost of $6,192,960, $9,550 more than the SSGA. However, ALCO-GA does achieve the lowest standard deviation for the fitness and

feasible network cost suggesting that this ALCO-GA variant has a better chance of producing more consistent results than the SSGA and other ALCO-GA variants. ALCO-GA (100%) also achieves a feasible solution in an average of 67.8 solution evaluations, over 6 times faster than the SSGA and also with more consistency shown by the relatively low evaluations to feasible standard deviation.

Table 4-17. ALCO-GA best solution average results for 50 runs for the Hanoi problem

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	**1.618E-07**	2.571E-09	**6,183,410**	99,800	415.2	152.3
ALCO-GA (25%)	1.612E-07	2.480E-09	6,204,560	96,439	162.0	56.4
ALCO-GA (50%)	1.610E-07	2.692E-09	6,215,230	104,339	99.6	42.5
ALCO-GA (75%)	1.615E-07	2.575E-09	6,193,610	100,013	80.6	30.6
ALCO-GA (100%)	1.615E-07	**2.347E-09**	6,192,960	**90,537**	**67.8**	**25.9**

Figure 4-15 presents the average fitness from the best solution in each run over the 200,000 fitness evaluations for the Hanoi problem. Unlike the previous experimentation with the LCO-GA variants the ALCO-GA variants are very closely matched throughout the search of the algorithm, all displaying faster convergence than the SSGA in the first stages of the search. The best performing ALCO-GA variants are ALCO-GA (75%) and ALCO-GA (100%) achieving the best solution fitness for the majority of the search; only being surpassed by the SSGA after approximately 170,000 solution evaluations. Interestingly it is ALCO-GA (50%) that performs the worst out of all the algorithms; at approximately 20,000 evaluations the convergence rate of ALCO-GA (50%) decreases drastically and is surpassed by the SSGA at approximately 80,000 evaluations. This behaviour can be attributed to a small number of outlying runs which seem to get caught in a local minima; this is reinforced by the relatively high standard deviation presented in Table 4-17.

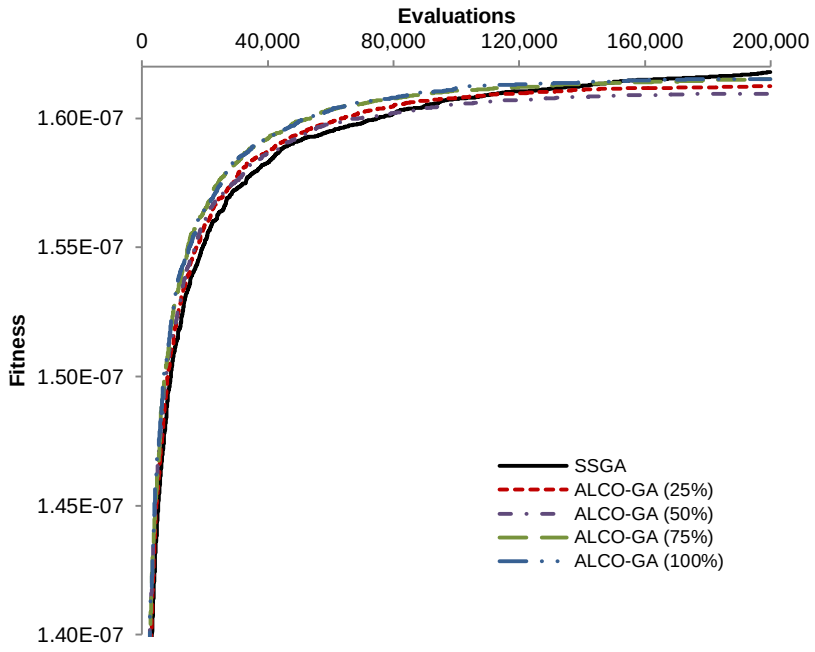


Figure 4-15. Graph showing the average best solution fitness over evaluations for the Hanoi problem - SSGA & ALCO-GA variants comparison

Figure 4-16 presents the average percentage of feasible solution present in the population during the search. The first plot (left) spans the entire 200,000 fitness evaluations and the second (right) gives a detailed look at the first 2,000 evaluations. The first thing to note is that all of the algorithms have very similar population feasibility levels in the first approximately 40,000 evaluations of the search. As observed previously, the population feasibility of the SSGA remains almost constant for the entire run following the initial search into the feasible search space (~0 – 2000 evaluations). The first 3 ALCO-GA variants ALCO-GA (25%), (50%), (75%) sustain roughly the same feasibility levels as the SSGA up until approximately 125,000 evaluations, where they proceed to drop. This behaviour is interesting because at this point in the search the probability of the heuristic being applied is very low, although it still seems to have an effect on the population feasibility possibly attributing to the decrease in feasible solutions. Finally looking at the second plot (right) in Figure 4-16 it can be seen that as the influence of the adaptive heuristic mutation operator is increased, fewer evaluations are needed for the algorithm to reach the feasible solution space.

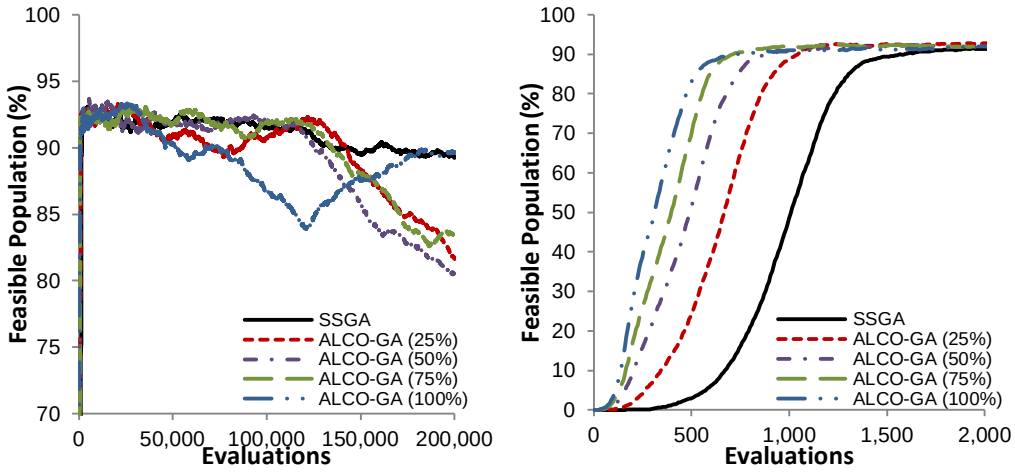


Figure 4-16. Graphs showing average percentage of feasible solutions present in the population for the Hanoi problem – SSGA & ALCO-GA variants comparison

The final part of this results section presents and compares the SSGA and the best performing LCO-GA and ALCO-GA variants (LCO-GA (25%) & ALCO-GA (100%)) on the Hanoi problem using the results gain in the previous sections.

Table 4-18 presents the best results obtained for each algorithm from the best performing run for the Hanoi problem. All of the algorithms achieve the best known solution for the problem in question, however it is clear that ALCO-GA (100%) achieves a feasible solution in less evaluations than LCO-GA (25%) and the SSGA. This behaviour is expected as the hydraulic heuristic is primarily designed to improve the hydraulic feasibility of solutions and since ALCO-GA (100%) is solely utilizing the heuristic in the early stage of the search it finds the feasible search space faster than the other algorithms.

Table 4-18. SSGA, LCO-GA & ALCO-GA best single run results for the Hanoi problem

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**1.644E-07**	**6,081,220**	64
LCO-GA (25%)	**1.644E-07**	**6,081,220**	36
ALCO-GA (100%)	**1.644E-07**	**6,081,220**	**24**

The following table shows the mean results from the best solutions from the 50 runs of each algorithm. From the previous experiments it has been shown that the SSGA achieves a better average solution than the heuristic based mutation algorithms, but these results show that ALCO-GA (100%) achieves a better solution quality, standard deviation and evaluations to feasibility than LCO-GA (25%). Thus suggesting that the implementation of the adaptive approach has improved the resultant solution quality of the algorithm over a long run but also maintained the algorithm's ability to find the feasible search space in a small number of solution evaluations.

Table 4-19. SSGA, LCO-GA & ALCO-GA best solution average results for 50 runs for the Hanoi problem

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	**1.618E-07**	2.571E-09	**6,183,410**	99,800	415.2	152.3
LCO-GA (25%)	1.614E-07	2.648E-09	6,197,340	102,431	164.2	61.6
ALCO-GA (100%)	1.615E-07	**2.347E-09**	6,192,960	**90,537**	**67.8**	**25.9**

Figure 4-17 shows the average fitness for each algorithm over the 200,000 solution evaluations for the Hanoi problem. It is clear from these results and results from previous experiments (presented in earlier sections of this chapter) that both of the heuristic based mutation algorithms perform better than the SSGA in the early stages of the search. It can now be seen that LCO-GA (25%) converges faster than ALCO-GA (100%); outperforming the adaptive variant in the first 100,000 evaluations. However, ALCO-GA is able to achieve slightly higher mean fitness values than the LCO-GA post 100,000 evaluations.

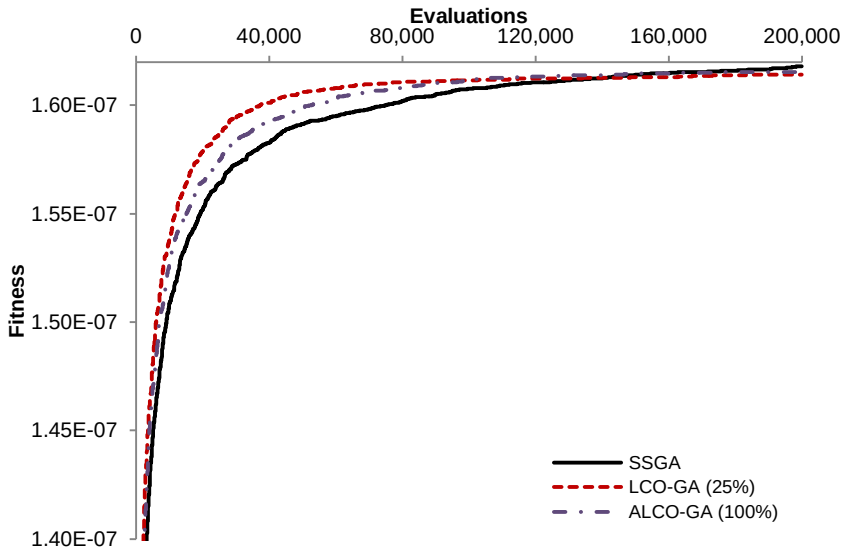


Figure 4-17. Graph showing the average best solution fitness over evaluations for the Hanoi problem – SSGA, LCO-GA (25%) & ALCO-GA (100%) comparison

Figure 4-18 shows the average percentage of feasible solutions present in the population for each of the algorithms for the Hanoi problem. The previous experimentations with LCO-GA and ALCO-GA have already confirmed that both algorithms reach the feasible solution space and achieve high population feasibility in fewer evaluations than the SSGA. It can be seen in these results that ALCO-GA (100%) reaches high population feasibility (> 90%) in less solution evaluations than LCO-GA (25%).

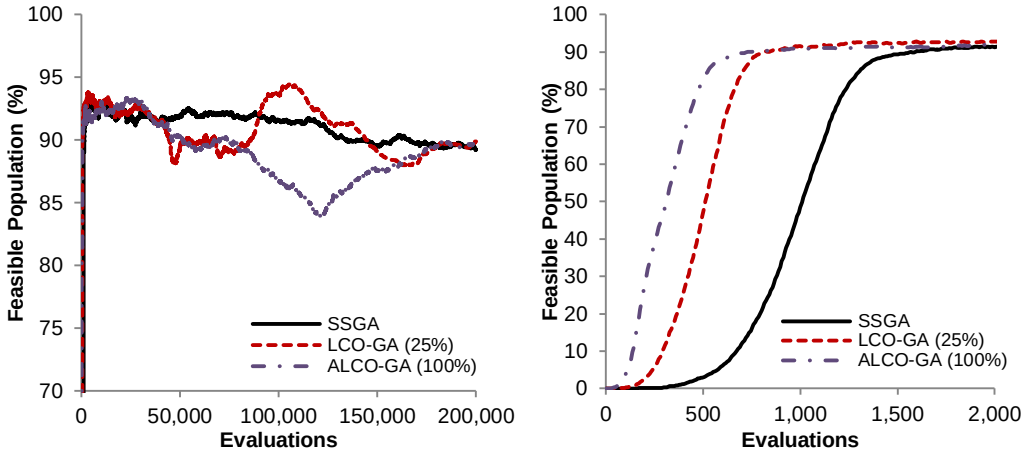


Figure 4-18. Graphs showing average percentage of feasible solutions present in the population for the Hanoi problem – SSGA, LCO-GA (25%) & ALCO-GA (100%) comparison

4.3.3 Conclusion

This section introduced the concept of the Fitness Gradient Monitor (FGM), an additional component added to LCO-GA which was designed to monitor the fitness of the best individual in the current population and calculate the resultant gradient over time. Utilizing this information, the application rate of the Heuristic Mutation Operator (HMO) is controlled; with the aim being to employ the HMO heavily in the initial stages of the search and less as the algorithm starts to converge on a solution. The resultant Adaptive Locally Constrained Genetic Algorithm (ALCO-GA) was applied to the Hanoi problem with varying amounts of adaptive HMO application to gauge what performance impact the operator has on the algorithm. It was found that decreasing the application of the adaptive HMO had a detrimental effect on the performance of ALCO-GA both in terms of mean fitness but also evaluations to the feasible solution space.

Finally the best performing LCO-GA variant (LCO-GA (25%)) from the previous results section and the best performing ALCO-GA variant (ALCO-GA (100%)) were compared. It has been observed that ALCO-GA performs better than LCO-GA in terms of solution quality after 200,000 evaluations and also promotes solution feasibility more than the LCO-GA variant. These experiments have shown that the application of the HMO in the early stages of an algorithm's search is beneficial not only in terms of early solution fitness but also in promoting solution feasibility; however as the search continues, sustained use of the heuristic has a detrimental effect of the overall performance of the algorithm. Applying the adaptive fitness gradient component to the algorithm has corrected the performance of the algorithm by focusing the application of the HMO in the early stages of the search whilst letting the standard pipe mutation take over in the later stages resulting in a more stable and consistent performance.

4.4 Pipe Smoothing Genetic Algorithm

The Pipe Smoothing Genetic Algorithm (PSGA) is based around the principle that in a gravity fed WDN the diameter of any pipe is never greater than the sum of the diameter(s) of the directly upstream pipes. Networks that adhere to this rule can be seen to 'smoothly' transition from large to small diameters from source to the extremities of the network. This rule is routinely and implicitly applied by engineers when designing such networks as it makes little sense to follow a smaller diameter pipe with a larger one in the majority of circumstances. The larger pipe will cost more to install and will not add to the hydraulic capability of the system as it will be constrained by the smaller diameter pipe upstream. One further negative aspect of this arrangement is that velocities will be lower in the larger pipe and high water age can become an issue. A standard GA of course will mutate some of these inconsistent pipe selections from the final solution as they have a corresponding improvement in the cost function and no hydraulic penalty. However extensive experimentation has shown that even well-optimised solutions after hundreds of thousands of generations of a standard EA still contain significant numbers of incorrectly sized pipes in larger networks.

PSGA applies the rule described in detail in section 3.4.2 directly to the genotype without evaluating the effect this process has on the phenotype. The heuristic employed by PSGA is developed from the network topology of a specific problem and remains constant throughout the evolutionary process. The heuristic is applied to a solution through the mutation operator; where the probability of the heuristic being applied is defined by a preset algorithm parameter. It is the aim of the heuristic to guide the algorithm's search to the engineering feasible solution space to locate smoother WDN designs whilst maintaining the performance of a standard genetic algorithm. The PSGA mutation operator does not perform any additional partial or full fitness evaluations, except a single hydraulic simulation at initialisation to determine flow directions. This was an important consideration when developing PSGA as additional fitness evaluations would require further hydraulic evaluations, increasing algorithm run time.

PSGA is essentially a standard GA (SGA) which incorporates an additional feature; a pipe smoothing heuristic based mutation operator. The standard GA used was the steady-state GA (section 3.1.2) with tournament selection with tournament size *t* and single-point crossover. A binary string comprising of *N* sub-strings was employed where each sub-string represents the diameter of each pipe in the WDN. Mutation was conducted as a random pipewise mutation with probability *m*.

The pipe smoothing mutation operator randomly selects a pipe to be mutated. The sum of all the diameters of the directly upstream pipes is set as the maximum allowable diameter the current pipe can take. This operator employs a skewed roulette wheel approach to the random selection of the pipe diameter. This is achieved be weighting the larger pipe diameters that fall within the maximum allowable size so that the larger the diameter, the higher the probability there is of selection. Upon selection the pipe being mutated is changed to the selected diameter.

To function correctly both the pipe smoothing initialiser and mutation operator require each pipe in the network to be 'aware' of the pipes directly up and down stream of their location. When changes are made to a WDN there is a possibility that flow direction could change in some pipes hence swapping up & down stream pipes relative to the pipe in question. The flow direction is logged at each hydraulic evaluation of the network, therefore to preserve this hydraulic data the pipe smoothing mutation operator precedes the crossover operator.

4.4.1 Experimental Setup

During early experimentation with PSGA it was found that if the pipe smoothing mutation operator was employed exclusively (i.e., without random pipe mutation) throughout the evolutionary process, the performance of the algorithm would be diminished compared with the SSGA. As with ALCO-GA it was decided that combining heuristic based mutation and random pipe mutation (employed by the SSGA) would help to prevent premature convergence and sub-optimal performance. Therefore this first set of experiments explores the performance of the algorithm with varying applications of the pipe smoothing

mutation operator vs. standard pipe mutation. The probability of pipe smoothing heuristic based mutation being applied for each pipe selected for mutation was set at a constant 25%, 50%, 75% and 100%. In the case where the heuristic is not selected for use the mutation operator employs the standard pipe mutation method as used by the SSGA resulting in a constant number of mutations regardless of the mutation type being used.

These initial experiments were conducted on the Hanoi problem and the optimal parameter values obtained from the single-objective genetic algorithm parameter tuning section of this chapter were used for all instances of the SSGA and PSGA in this set of experiments. Each configuration of PSGA and the SSGA were run 50 times, each for 200,000 objective function evaluations. To ensure a fair comparison between the SSGA and the different PSGA configurations a list of 50 randomly generated integers were generated for the purpose of seeding the random number function at the beginning of each of the 50 runs. The same list of random number seeds were used for each algorithm configuration experiment ensuring the same starting populations for each run, for example, PSGA 1st run would have the same starting population as the 1st run of the SSGA. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size of 100 individuals, single-point crossover, penalty cost of $500,000, tournament size of 2% and a pipe mutation rate of 0.147. Conditional worst individual replacement strategy was used as per the previous experimentation. As with the SSGA, PSGA was implemented in C++ and all of the following experiments were run on the same system as the previous experiments.

4.4.2 Results

This section presents the results for the initial PSGA experiments and compares them to the performance of the SSGA. The table below shows the best results obtained for each algorithm from the best performing run for the Hanoi problem. Both the best fitness and best feasible network cost are presented along with the least number of fitness evaluations taken for that algorithm to reach a feasible solution. From these results it can be observed that the SSGA and all PSGA configurations achieved the best known feasible solution for the Hanoi problem ($6,081,220) within the allotted 200,000 fitness

evaluations. Looking at the number of fitness evaluations it takes the algorithms to achieve a hydraulically feasible solution shows that the more the heuristic mutation operator is applied the slower the algorithm finds the feasible solution space, however PSGA (100%) does achieve a feasible solution faster than the other PSGA configurations but still falls short of the SSGA which took 64 evaluations.

Table 4-20. PSGA best single run results for the Hanoi problem - SSGA & PSGA variants comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**1.644E-07**	**6,081,220**	**64**
PSGA (25%)	**1.644E-07**	**6,081,220**	236
PSGA (50%)	**1.644E-07**	**6,081,220**	264
PSGA (75%)	**1.644E-07**	**6,081,220**	314
PSGA (100%)	**1.644E-07**	**6,081,220**	190

The following table presents the mean results from the best solutions from the 50 runs of each algorithm variant. Included in these results is the best average fitness, feasible cost, evaluations taken to achieve a feasible solution and pipe smoothing violations each accompanied by the standard deviation for the best results from the 50 runs. Pipe smoothing violations are a count of how many times the pipe smoothing heuristic is violated in a solution and hence can be used as a metric of network smoothness where the lower the violations the smoother the network.

Table 4-21. PSGA 50 run best solution average results for the Hanoi problem - SSGA & PSGA variants comparison

	50 Run Best Solution Average							
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**	**Pipe Smoothing Violations**	**Pipe Smoothing Violations Standard Deviation**
SSGA	1.618E-07	**2.571E-09**	6,183,410	**99,800**	**415.2**	152.3	0.00	0.00
PSGA (25%)	**1.621E-07**	2.585E-09	**6,170,850**	100,200	496.6	**140.8**	0.00	0.00
PSGA (50%)	1.615E-07	2.714E-09	6,194,280	105,406	545.4	187.4	0.00	0.00
PSGA (75%)	1.613E-07	2.850E-09	6,204,390	110,457	685.2	232.0	0.00	0.00
PSGA (100%)	1.616E-07	3.074E-09	6,192,390	119,208	700.5	237.0	0.00	0.00

From these results it can be seen that PSGA (25%) outperforms the SSGA in terms of average best solution fitness and network cost after 200,000 fitness evaluations. As with the previous best single run results, increasing the application of heuristic mutation allows the algorithm to find the feasible solution space in more fitness evaluations. It was found that for the Hanoi problem that as solution fitness increased so does the smoothness of the network, this resulted in all algorithms tested finding perfectly smooth networks (0 smoothing violations).

Figure 4-19 presents the average fitness from the best solution in each run over the 200,000 fitness evaluations for the Hanoi problem. From this graph it can be observed that all variants of the PSGA exhibit better performance in the early stages of the search compared to the SSGA. However the PSGA variants with very high applications of the heuristic mutation do not perform as well in the later stages of the search.

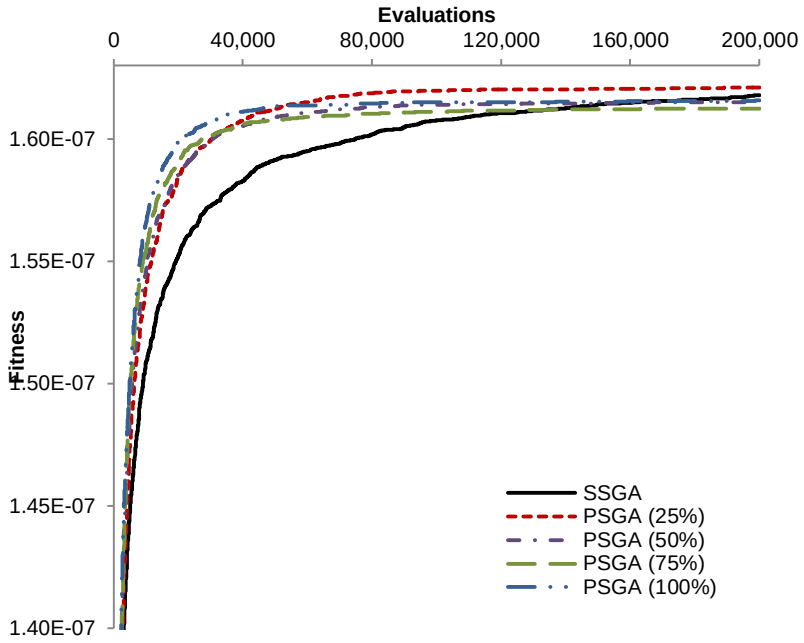


Figure 4-19. Graph showing the average best solution fitness over evaluations for the Hanoi problem - SSGA & PSGA variants comparison

For example, PSGA (100%) can be seen to rapidly converge on a slightly sub-optimal solution at approximately 70,000 evaluations and does not improve throughout the remaining ~130,000 evaluations, this is the case for the 50% & 75% also: These algorithms display very fast convergence to near optimal

solutions but are slightly outperformed eventually by the SSGA. As the application of the heuristic mutation is decreased the final solution quality of the algorithm in general is increased all be it with a slight decrease in the early stage performance of the search. The best performing PSGA variant is PSGA (25%) which achieves better solution quality throughout the entirety of the search. In short all PSGA variants outperform the SSGA in the first 100,000 fitness evaluations of the search.

Figure 4-20 shows the average percentage of feasible solutions present in the population during the search (left) and the average best solution's pipe smoothing violations (right). It can be seen from this figure that there is minimal difference in population feasibility between all algorithms in the first ~40,000 evaluations of the search, however the population feasibility of the PSGA variants decrease slightly as the search progresses compared to the SSGA. Looking at the pipe smoothing violations produced by the algorithms it can be seen that as the pipe smoothing heuristic is applied to a greater extent the average number of violations decreases at a higher rate. This of course is an expected behaviour of the PSGA.

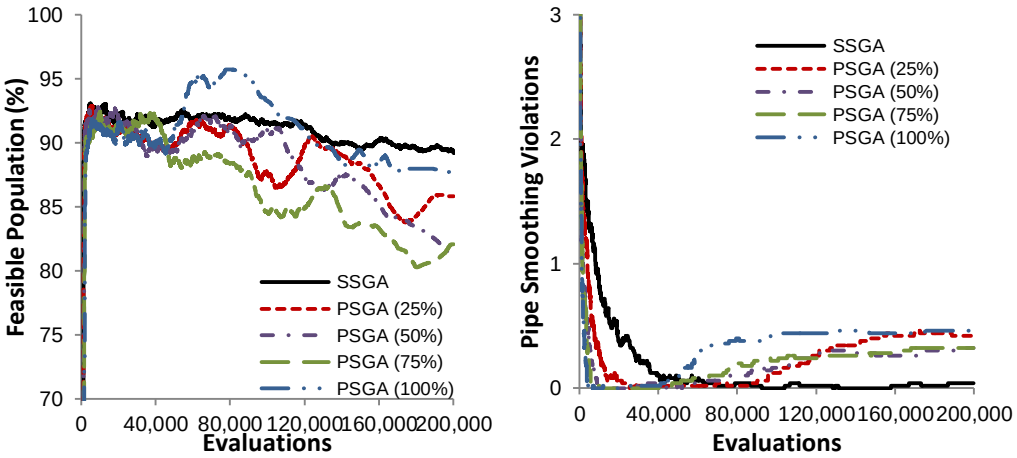


Figure 4-20. Graphs showing average percentage of feasible solutions present in the population for the Hanoi problem – SSGA & PSGA variants comparison

4.4.3 Conclusion

This section presented the results from the initial Pipe Smoothing Genetic Algorithm (PSGA) experiment set on the Hanoi problem with the view of investigating how the application strength of the pipe smoothing heuristic

mutation operator affects the performance of the algorithm both in terms of solution fitness but also solution feasibility and network smoothness. It was found that the higher the probability of heuristic based mutation the faster the algorithm converged. However at high application rates of the heuristic based mutation the overall performance of the algorithm was diminished in the later stages of the search compared with the standard Steady State Genetic Algorithm (SSGA). The PSGA configuration found to perform best was PSGA (25%) (25% probability of applying the heuristic during mutation). PSGA (25%) displayed faster convergence over that of the SSGA producing higher fitness solutions than the SSGA throughout the entire 200,000 evaluation search. A key performance behaviour of PSGA is the algorithm's ability to find smoother network solutions in less solution evaluations than the SSGA, as mentioned earlier in this chapter, this trait is very desirable from a design engineer's point of view, especially in the case where time and hence solution evaluations are limited.

It is clear from this set of experiments that the pipe smoothing heuristic based mutation aids the search of the algorithm throughout all stages of the search in terms of fitness value convergence rate and also the smoothness of the resultant network solutions. However as has been observed, the sole use of the heuristic based mutation is sub-optimal; requiring the standard pipe mutation to improve the overall performance of the Pipe Smoothing Genetic Algorithm. From this stage on PSGA (25%) will be referred to as the Pipe Smoothing Genetic Algorithm (PSGA).

4.5 Conclusion

The first section of this chapter explored the effect of varying the different parameters of a single-objective steady state genetic algorithm on a number of small to medium size WDN design problems from the literature. The experiments were designed to give insight into the effect that different parameters have on the performance of the algorithm but also ensure the algorithm is performing at near peak performance to provide a fair comparison to the heuristic based algorithms. It was found that the penalty cost has little to no significant effect on the performance of the algorithm except for very low penalty costs. The results also show a potential correlation between problem

complexity and tournament size; the larger the problem the greater the optimal tournament size. It was also observed that as the problem complexity grows the optimal mutation rate generally falls and that the mutation rate has a much larger impact of the performance of the algorithm than any other parameter tested. Finally it was observed that the algorithm performed best when the worst individual in the population was selected for replacement. It was also observed that the use of conditional replacement worked best for the smaller problems from the problem set and an unconditional replacement approach performed best for the larger problems. This work goes some way to understanding algorithm parameter sensitivity and parameter interactions to aid in addressing of the second research question posed in this thesis.

Following the parameter investigation of the SSGA, the development of the Locally Constrained Genetic Algorithm (LCO-GA) is presented. The experiment was designed to investigate how the application strength of the bottleneck eliminating mutation operator (3.4) impacts the performance of the algorithm both in terms of solution fitness but also solution feasibility. It was found that the higher the probability of heuristic based mutation the faster the algorithm found the feasible search space thus increasing the initial convergence of the algorithm. However at these high application rates of the heuristic based mutation the overall performance of the algorithm was diminished in the later stages of the search compared with the standard Steady State Genetic Algorithm (SSGA) often prematurely converging on a sub optimal solution. LCO-GA exhibited the ability to find the feasible solution space in less solution evaluations than the SSGA, this trait is very desirable from a design engineer's point of view, especially in the case where time and hence solution evaluations are limited. Therefore it was necessary to vary the application rate of the heuristic over the duration of the search. An additional component added to LCO-GA which was designed to monitor the fitness of the best individual in the current population and calculate the resultant gradient over time. Utilizing this information, the application rate of the Heuristic Mutation Operator (HMO) is controlled; with the aim being to employ the HMO heavily in the initial stages of the search and less as the algorithm starts to converge on a solution. It has been observed that the resultant Adaptive Locally Constrained Genetic

Algorithm (ALCO-GA) performs better than LCO-GA in terms of solution quality and also promotes solution feasibility more than the LCO-GA variant.

The final section of this chapter presents the development of the Pipe Smoothing Genetic Algorithm (PSGA) an algorithm based off of the pipe smoothing heuristic presented in chapter 3 (3.5). A set of experiments aimed at investigating how the application strength of the pipe smoothing heuristic mutation operator impacts the performance of the algorithm both in terms of solution fitness but also solution feasibility and network smoothness was presented. It was found that the higher the probability of heuristic based mutation the more susceptible the algorithm was to premature convergence compared with the standard Steady State Genetic Algorithm (SSGA). It was found that the pipe smoothing heuristic based mutation aids the search of the algorithm throughout all stages of the search in terms of fitness value, convergence rate and also the smoothness of the resultant network solutions.

It has been demonstrated in this chapter that the incorporation of water systems knowledge to the mutation operator of an EA has the potential to improve overall algorithm performance in terms of both mathematical optimality and engineering feasibility, paving the way to addressing the primary research question posed by this thesis.

Chapter 5: Heuristic Based Algorithm Experimentation

During the development of the engineering inspired genetic algorithms (chapter 4) it was observed that both algorithms (ALCO-GA & PSGA) demonstrated increased performance when compared to a highly tuned Steady State Genetic Algorithm (SSGA) on the Hanoi water distribution network design problem. The integration of specific water systems knowledge into the mutation operator was successful and demonstrated promising performance on a medium sized benchmark problem from the literature. However, to fully address the primary research question posed by this work, it is necessary to expand the problem set to include problems of differing types and increased complexity. In this chapter both ALCO-GA and PSGA are directly compared with the SSGA on a wide range of water distribution network design problems presented in section 3.3 with the view of fully analyzing the performance of the engineering inspired algorithms. In addition, the performance of the SSGA, ALCO-GA and PSGA are compared over a number of different parameter configurations involving the tournament size and mutation rate in order to gauge the sensitivity each algorithm has to variation of these parameters.

5.1 Adaptive Locally Constrained GA Experimentation

Following the development of the Locally Constrained Genetic Algorithm (LCO-GA) (4.2) and the Adaptive Locally Constrained Genetic Algorithm (ALCO-GA) (4.3) this section aims to explore the performance of ALCO-GA on a range of water distribution network design problems of differing structure, complexity and size. The initial experimentation presented in this section compares ALCO-GA and the SSGA in terms of overall solution quality achieved and also solution feasibility. The final section explores the robustness of ALCO-GA and SSGA when different parameters are varied.

5.1.1 ALCO-GA Tuned Performance Experiments

5.1.1.1 Experimental Setup

The set of experiments presented in this section compare ALCO-GA and the SSGA, applying them to a variety of different water distribution network design problems, a list and brief description of each of the networks is available

in section 3.3. As stated before the collection of water distribution network design problems were selected for their structure, complexity and size variations, providing a diverse benchmark set to enable extensive evaluation of ALCO-GA. The optimal parameter values obtained from the single-objective genetic algorithm parameter tuning section (0) of this chapter were used for all instances of the SSGA and ALCO-GA in this set of experiments. ALCO-GA and the SSGA were run 50 times with the exception of the Network A problem which was only run 10 times, each for 100,000 objective function evaluations. To ensure a fair comparison between the SSGA and ALCO-GA a list of 50 (10 for Network A) randomly generated integers were generated for the purpose of seeding the random number function at the beginning of each of the 50 runs. The same list of random number seeds was used for both ALCO-GA and the SSGA ensuring the same starting populations for each run. The SSGA and ALCO-GA were implemented in C++ and all of the following experiments were run on an Intel Core i7-4770K 3.5GHz PC.

5.1.1.2 Results

5.1.1.2.1 Two Loop

The following results show the performance comparison between SSGA and ALCO-GA for the Two Loop water distribution network design problem. Table 5-1 presents the best performing solutions for each algorithm from the 50 runs. It can be seen that both algorithms find the best known feasible solution for this problem ($419,000). Both algorithms produce a feasible solution in the initial population, an important thing to note in regard to this is that due to the small size of the problem and the pipe diameter decision set a randomly generated solution has a very high probability to produce a solution which satisfies the hydraulic constraints of the problem, also as stated previously the starting populations of both algorithms are identical as they are generated using the same random number seed set.

Table 5-1. Best single run results for the Two Loop problem - SSGA & ALCO-GA comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**2.387E-06**	**419,000**	0
ALCO-GA	**2.387E-06**	**419,000**	0

Table 5-2 shows the mean results for the algorithms for the Two Loop problem. As can be seen, ALCO-GA achieves a better fitness and feasible solution cost than the SSGA also obtaining a lower standard deviation for the best solutions from the 50 runs over the 100,000 solution evaluations. In this set of experiments all of the solutions in the initial population were hydraulically feasible, resulting in the 0 evaluations to feasible result for both algorithms.

Table 5-2. Mean results for the Two Loop problem – SSGA & ALCO-GA comparison

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	2.383E-06	6.301E-09	419,740	1,426	0.0	0.0
ALCO-GA	**2.384E-06**	**2.832E-09**	**419,420**	**499**	0.0	0.0

As can be seen from Figure 5-1, ALCO-GA displays faster convergence than the SSGA, achieving a near optimal solution at approximately 60,000 solution evaluations. In terms of solution feasibility, both algorithms start at an average of 10% population feasibility and quickly rise in the first 200 evaluations with ALCO-GA rising to approximately 50% feasibility and SSGA to around 40%. Interestingly population feasibility becomes reduced for both algorithms in the next 10,000 evaluations, however at this point population feasibility starts to rise significantly for ALCO-GA leaving the SSGA to remain in the 25%-30% range.

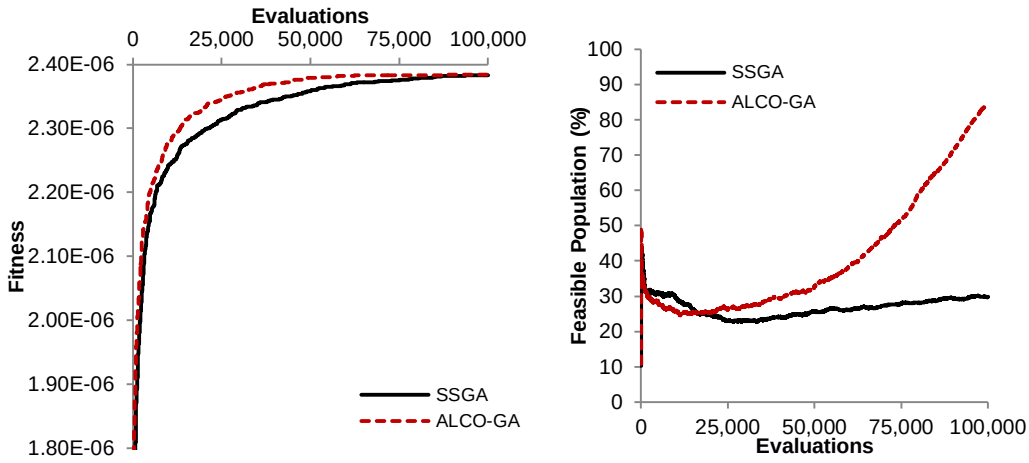


Figure 5-1. Graphs showing the average best solution fitness (left) & average percentage of feasible solutions present in the population (right) for the Two Loop problem – SSGA & ALCO-GA comparison

5.1.1.2.2 Foss Poly 1

The following results presented here are for the Foss Poly 1 water distribution network design problem. It can be seen from Table 5-3 that following the allotted 100,000 solution evaluations ALCO-GA finds a better solution for the 50 runs than the SSGA, achieving a feasible network cost $4,805 less than the best solution found by the SSGA. As with the previous Two Loop experiment, this problem also offers a high probability that a solution in the randomly generated starting population is feasible, and this being the case in this instance where on average approximately 30% of feasible solutions were generated in the starting population.

Table 5-3. Best single run results for the Foss Poly 1 problem - SSGA & ALCO-GA comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	2.496E-05	40,063	0
ALCO-GA	**2.844E-05**	**35,258**	0

The average results for the 50 runs of both algorithms (Table 5-4) shows that ALCO-GA achieves a better average solution fitness and feasible network

cost than the SSGA. Again, due to the presence of feasible solutions in the starting population of all of the runs, the average evaluations to the feasible solution space is 0.

Table 5-4. Mean results for the Foss Poly 1 problem – SSGA & ALCO-GA comparison

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	2.218E-05	**1.184E-06**	47,030	3,001	0.0	0.0
ALCO-GA	**2.603E-05**	1.299E-06	**39,662**	**2,394**	0.0	0.0

Figure 5-2 displays (left) the extent at which ALCO-GA outperforms the SSGA in terms of average solution fitness during the search of both algorithms for the Foss Poly 1 problem. From these results it is clear that ALCO-GA achieves better average solution fitness than the SSGA at every stage of the 100,000 evaluation search. Both algorithms achieve similar population feasibility in the early stages of the search; however the population feasibility of ALCO-GA is lower than that of the SSGA in the first 30,000 evaluations, dropping to around 5% feasible solutions. This suggests that the hydraulic excess limiting component of the modified mutation operator is playing a more significant role in the early stages of the search that it has in other problems such as Hanoi.

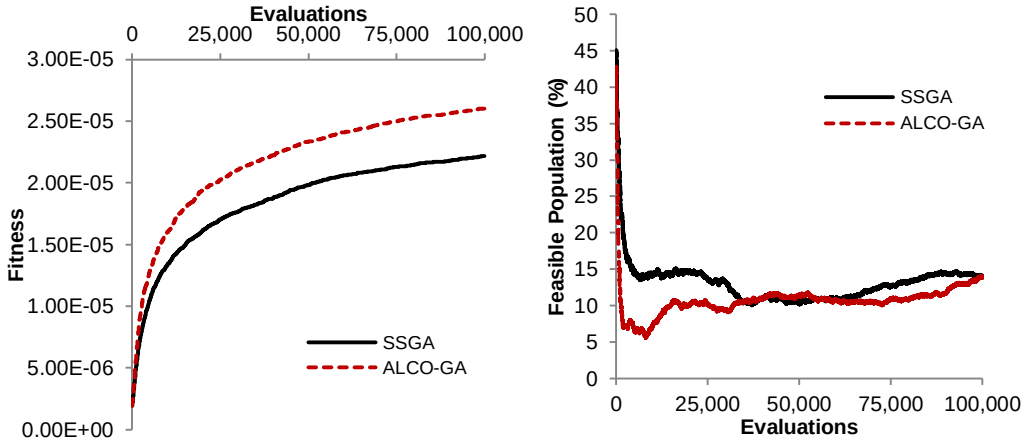


Figure 5-2. Graphs showing the average best solution fitness (left) & average percentage of feasible solutions present in the population (right) for the Two Loop problem – SSGA & ALCO-GA comparison

5.1.1.2.3 Modena

The following results are for the Modena water distribution network design problem, a large real-world network with multiple reservoir sources. Table 5-5 show the best single run results for both algorithms after the allotted 100,000 fitness evaluations. As can be seen from the table, the SSGA finds a solution just less than $10,000 cheaper than ALCO-GA. As with the previous two problems presented in this results section, the makeup of this problem gives a high probability that a feasible solution can be randomly generated hence the 0 evaluations taken to find a feasible solution for both algorithms.

Table 5-5. Best single run results for the Modena problem - SSGA & ALCO-GA comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**1.762E-07**	**5,898,900**	0
ALCO-GA	1.710E-07	5,908,630	0

Table 5-6 presents the average results for the Modena problem and shows that ALCO-GA outperforms the SSGA in terms of solution quality and solution variation for the 50 runs; ALCO-GA achieves a mean feasible network cost just under $100,000 less than the SSGA and a smaller feasible cost standard deviation. As stated previously the Modena problem lends itself to producing feasible solutions, this is demonstrated by the 0 mean evaluations to feasible result presented in the table below, thus implying that at least one feasible solution was generated in each of the 50 runs performed in this experiment set.

Table 5-6. Mean results for the Modena problem – SSGA & ALCO-GA comparison

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	1.617E-07	4.253E-09	6,385,570	167,052	0.0	0.0
ALCO-GA	**1.639E-07**	**3.176E-09**	**6,290,200**	**151,066**	0.0	0.0

Figure 5-3 presents the mean best solution fitness for the 50 runs over the 100,000 evaluations for both algorithms (left) and the average percentage of feasible solutions present in the population (right) for the Modena problem. From this figure it can be seen that ALCO-GA outperforms the SSGA throughout the entire search of the algorithms. It can also be seen that the mean population feasibility for ALCO-GA drops immediately from 8% to 1% at the beginning of the search and then proceeds to climb back to the feasibility levels of the SSGA at around 6-7%. The overall population feasibility levels generated by both algorithms are quite low for this problem suggesting that in this case the search of a GA benefits from having solutions close to the feasible solution space boundary. This is a trait that is encouraged by the heuristic based mutation operator in ALCO-GA and hence why there are performance gains to be made in the early stages of the search.

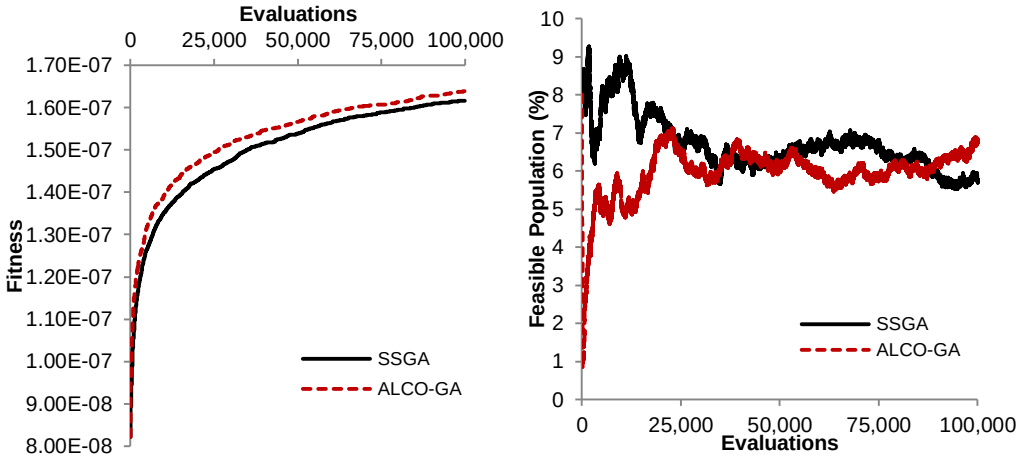


Figure 5-3. Graphs showing the average best solution fitness (left) & average percentage of feasible solutions present in the population (right) for the Modena problem – SSGA & ALCO-GA comparison

5.1.1.2.4 New York Tunnels

The following section presents the results for both algorithms for the New York Tunnels network, a parallel expansion problem. Table 5-7 shows that both algorithms achieve the same best single run results, reaching the best know solution for this formulation of the problem ($38,637,600).

Table 5-7. Best single run results for the New York Tunnels problem - SSGA & ALCO-GA comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**2.615E-08**	**38,637,600**	0
ALCO-GA	**2.615E-08**	**38,637,600**	0

The following mean results table shows the SSGA slightly outperforms ALCO-GA in terms of solution fitness and feasible network cost although the gains in performance the SSGA has over ALCO-GA are minimal compared with the total cost of the network. As with some of the previous problems, the nature of the New York Tunnels parallel expansion problem highly promotes feasibility which in turn increases the probability that randomly generated solutions will satisfy the hydraulic constraints of the problem. In this case we see from the results that there are solutions present in each of the 50 starting populations that are feasible, hence resulting in the 0 evaluations to feasible result for each algorithm.

Table 5-8. Mean results for the New York Tunnels problem – SSGA & ALCO-GA comparison

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	**2.597E-08**	2.271E-10	**38,919,500**	**355,019**	0.0	0.0
ALCO-GA	2.594E-08	**2.253E-10**	38,952,400	367,342	0.0	0.0

Figure 5-4 shows that ALCO-GA converges considerably faster than the SSGA in the early stages of the search, outperforming the SSGA for the first ~30,000 evaluations whilst sustaining a completive solution quality throughout the remainder of the search. The second graph indicates that the average starting population percent feasibility is just under 90% however as is shown in the figure, the population feasibility of both algorithms decreases rapidly in the early stages of the search. This behaviour is somewhat expected as it adheres to the observed performance of the algorithms when applied to problems with

high initial solution feasibility. It can be seen that ALCO-GA reduces solution feasibility very aggressively at the beginning of the search whilst reducing the cost of the network and increasing solution fitness.

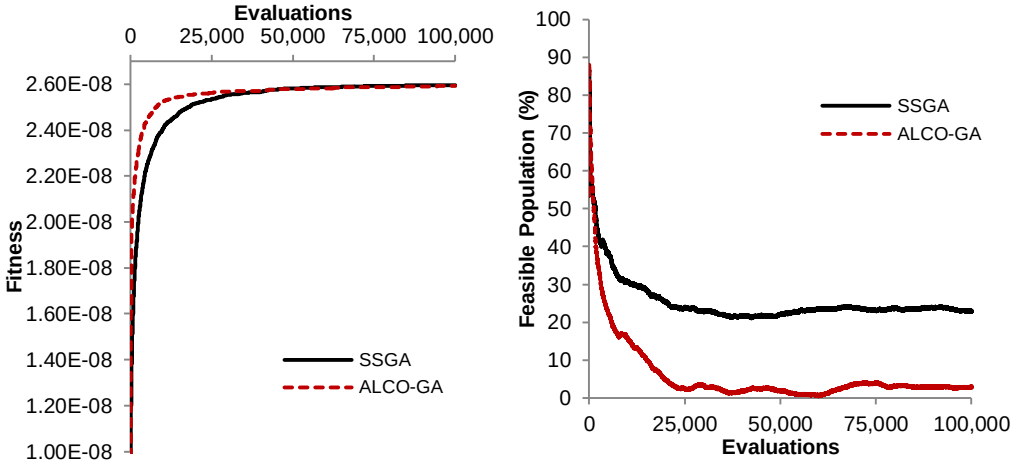


Figure 5-4. Graphs showing the average best solution fitness (left) & average percentage of feasible solutions present in the population (right) for the New York Tunnels problem – SSGA & ALCO-GA comparison

5.1.1.2.5 Network A

The following set of results is for the Network A water distribution network design problem. This problem presents a real-world challenge for optimization algorithms due to the network complexity and hydraulic constraints which are difficult to satisfy whilst maintaining competitive solution quality. Table 5-9 presents the best single run results for the Network A problem for the SSGA and ALCO-GA after 100,000 solution evaluations. It can be seen that the SSGA finds a better feasible cost solution than ALCO-GA from the 10 runs after 100,000 fitness evaluations, however ALCO-GA finds a feasible solution in only 60 evaluations compared with the 1,934 evaluations it takes the SSGA to do so.

Table 5-9. Best single run results for the Network A problem - SSGA & ALCO-GA comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**2.723E-07**	**3,672,300**	1934
ALCO-GA	2.661E-07	3,758,000	**60**

The following table shows the average results from all the runs for this problem. As can be seen ALCO-GA achieves a slightly better average feasible cost however has a slightly lower average fitness value than the SSGA. This result is due to having a relatively low penalty factor where the reduction in infrastructure cost is more than the resultant penalty cost produced by a minimal head deficit. This becomes pronounced in problems such as these where producing competitive hydraulically feasible solutions is more difficult. ALCO-GA greatly outperforms the SSGA in terms of evaluations taken to achieve a feasible solution, finding the feasible solution space on average in less than 2% of the evaluations that the SSGA takes to achieve a feasible solution.

Table 5-10. Mean results for the Network A problem – SSGA & ALCO-GA comparison

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	**2.530E-07**	1.519E-08	4,024,560	321,490	6333.2	4814.9
ALCO-GA	2.528E-07	**1.213E-08**	**3,976,580**	**221,773**	**120.2**	**35.1**

The following figure shows the mean best solution performances of both algorithms are very similar in terms of solution fitness throughout the entire search. Following further statistical testing (Mann-Whitney U test) it was found that there was no significant difference between each algorithm's set of final solutions in terms of fitness and feasible solution cost. The only clear difference in performance between the algorithms is the ability for ALCO-GA to promote highly feasible solutions in the population achieving over 90% population feasibility in less solution evaluations than the SSGA takes to find a single feasible solution.

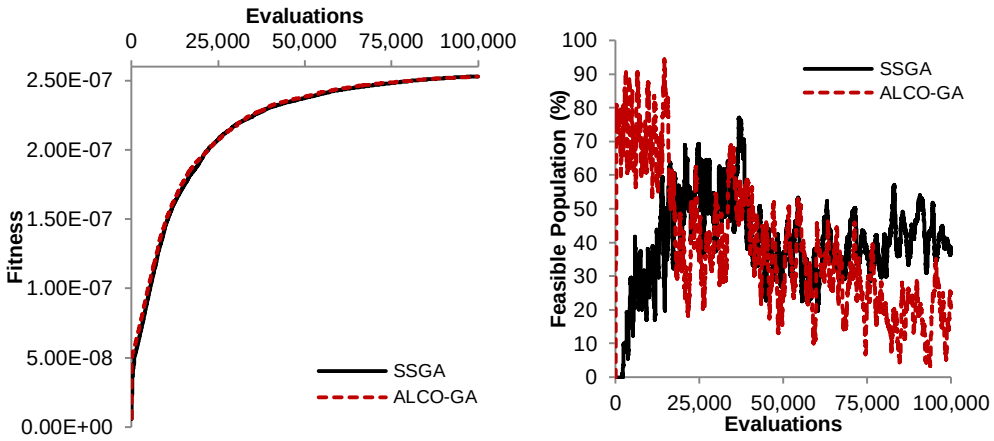


Figure 5-5. Graphs showing the average best solution fitness (left) & average percentage of feasible solutions present in the population (right) for the Network A problem – SSGA & ALCO-GA comparison

5.1.1.2.6 Network B

The following section presents the results for the Network B problem, a real-world water distribution network and the largest benchmark problem presented in this work. The following table shows the best solutions found by both ALCO-GA and the SSGA after 100,000 solution evaluations from ten randomly seeded separate runs. In terms of solution fitness and feasible network cost, ALCO-GA finds a slightly better solution however as with the previous real world large scale network (Network A) ALCO-GA finds a feasible solution in a lot less fitness evaluations than the SSGA.

Table 5-11. Best single run results for the Network B problem - SSGA & ALCO-GA comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	1.125E-07	8,887,670	10238
ALCO-GA	**1.133E-07**	**8,835,080**	**304**

Table 5-12 presents the mean results for the two competing algorithms for this problem. The results show that the SSGA slightly outperforms ALCO-GA in terms of solution fitness and feasible network cost after 100,000 fitness

evaluations. However as with the Network A experiment set, these solution sets also do not provide statistically significant fitness or feasible network cost results. However there is statistically significant difference between the two sets of results when comparing the evaluations it takes to reach a feasible solution. With ALCO-GA on average reaching the feasible solution space in approximately a tenth of the time of the SSGA.

Table 5-12. Mean results for the Network B problem – SSGA & ALCO-GA comparison

	Average					
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**
SSGA	**1.049E-07**	**4.863E-09**	**9,827,230**	**768,935**	17027.4	6555.6
ALCO-GA	1.022E-07	6.816E-09	10,024,500	900,713	**1708.4**	**2822.1**

Figure 5-6 presents the average fitness and population feasibility results for the two algorithms on the Network B problem. The first observation is that ALCO-GA seems to perform better than the SSGA in the first stages of the search, however the SSGA goes on to overtake ALCO-GA in terms of fitness after approximately 25,000 solution evaluations. As with the previous problem (Network A), ALCO-GA exhibits a strong tendency to produce feasible solutions in the early stages of the search. This behaviour is shown in the figure below with the mean population feasibility reaching around 45% in the first 2,000 fitness evaluations compared with the SSGA which only finds its first feasible solution at approximately 10,000 solution evaluations.

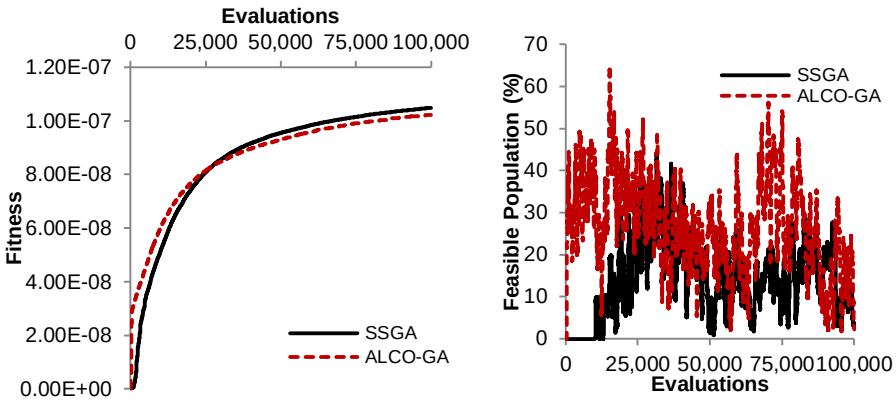


Figure 5-6. Graphs showing the average best solution fitness (left) & average percentage of feasible solutions present in the population (right) for the Network B problem – SSGA & ALCO-GA comparison

5.1.1.3 Conclusion

The primary aim of this section was to compare the performance of the Adaptive Locally Constrained Genetic Algorithm (ALCO-GA) with the Steady State Genetic Algorithm (SSGA). The problems used in this set of experiments were selected for their range of size, complexity and layout, hence offering a diverse set of challenges for both algorithms with the view of providing an insight into the performance and behaviour of ALCO-GA. Both algorithms utilized the same parameters obtained from the parameter tuning experiments presented earlier in this chapter (section 0). This was done to ensure that the SSGA was not placed at a disadvantage when being compared to ALCO-GA and both algorithms could be fairly compared.

ALCO-GA generally displayed increased performed over the SSGA on the smaller single source network design problems (Two Loop, Hanoi, Foss Poly 1). It was found that ALCO-GA displayed heightened performance in the early stages of the search compared with the SSGA and commonly went on to produce competitive final solutions, often better than the highly tuned SSGA. The results from the Modena problem suggest that ALCO-GA would work well for large, real world problems with multiple sources.

Integrating a relatively simple problem specific heuristic into the mutation operator of a standard steady state genetic algorithm has shown to improve algorithm performance at least in the early stages of the search but also in

many cases improve overall solution quality. Reaching quality feasible solutions in less fitness evaluations would allow an engineer to utilize such a method to aid in the initial design of a water distribution system.

5.1.2 Parameter Robustness

Identifying the optimal GA parameter set can involve a large number of algorithm runs using a number of different parameter configurations. This can be a time consuming operation and in the case of complex large real-world problems is near infeasible. The sensitivity of an algorithm to its parameter configuration (parameter robustness) is therefore an important consideration, and in this section, the performance of both the SSGA and ALCO-GA are compared over a number of different parameter configurations involving the tournament size and mutation rate.

5.1.2.1 Experimental Setup

As with the parameter tuning experiments conducted earlier in this chapter, for each individual network problem eight evenly distributed parameter values were selected for each of the test parameters (tournament size & mutation probability). The parameter values used in this set of experiments are the same as those used in the initial parameter tuning runs presented in section 0, however in this experiment set the penalty factor was used as a variable and was fixed to the best performing value found during the initial parameter tuning experiments for each problem. Using an exhaustive approach, each of the possible parameter value combinations were tested on the selected network problem. For each parameter value combination SSGA and ALCO-GA were run 50 times, each for 100,000 objective function evaluations utilizing a different randomly seeded starting population for each run, however for fairness the two algorithms utilized the same random number seed list, as has been the case in previous comparison experiments. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size was fixed at 100 individuals, single-point crossover was utilized and a conditional worst individual replacement strategy was implemented.

5.1.2.2 Results

This section presents the results from the parameter robustness experiments for the SSGA and ALCO-GA. To enable the comparison between the two algorithms the mean results for each parameter value were plotted for each algorithm. The area under/above (maximization/minimization) the resultant curve is highlighted to aid in the comparison of the two algorithms' performance over the selected range of parameter values. The resultant areas are then normalized using the area produced by the two most extreme data points produced by the experiments, thus resulting in a proportional representation for each algorithm where the greater the proportion the greater the parameter robustness of the algorithm and the less susceptible it is to parameter variance.

5.1.2.2.1 Hanoi

The following set of figures present a detailed look at the performance of the SSGA and ALCO-GA for a range of tournament sizes on the Hanoi problem. The first two figures compare the mean best solution fitness and feasible network cost for tournament sizes ranging between 0.02N – 0.09N (where N is the population size). It can be seen from these results that ALCO-GA achieves better average fitness and feasible network costs than the SSGA for the majority of tournament sizes, with the only exception being at a tournament size of 0.04.

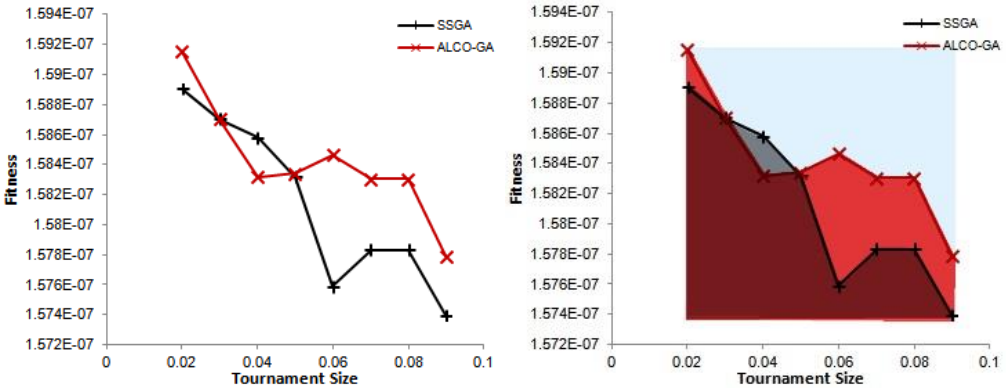


Figure 5-7. Average fitness at varying tournament sizes for the Hanoi problem – SSGA & ALCO-GA

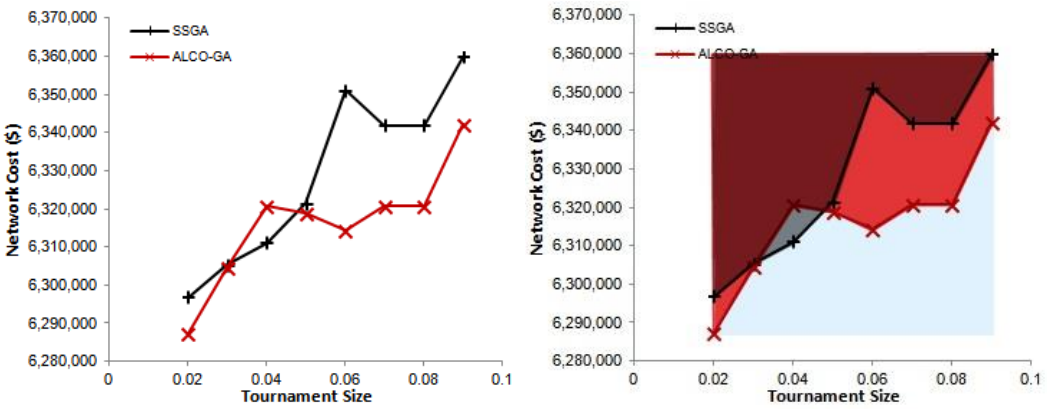


Figure 5-8. Average feasible network cost at varying tournament sizes for the Hanoi problem - SSGA & ALCO-GA

The following figure displays the mean percentage of runs which achieve the best known solution for the Hanoi problem ($6,081,220) for the stated range of tournament sizes. It can be seen that ALCO-GA finds the best known solution more frequently than the SSGA for small tournament sizes under 0.04N and tournament sizes above 0.06N.

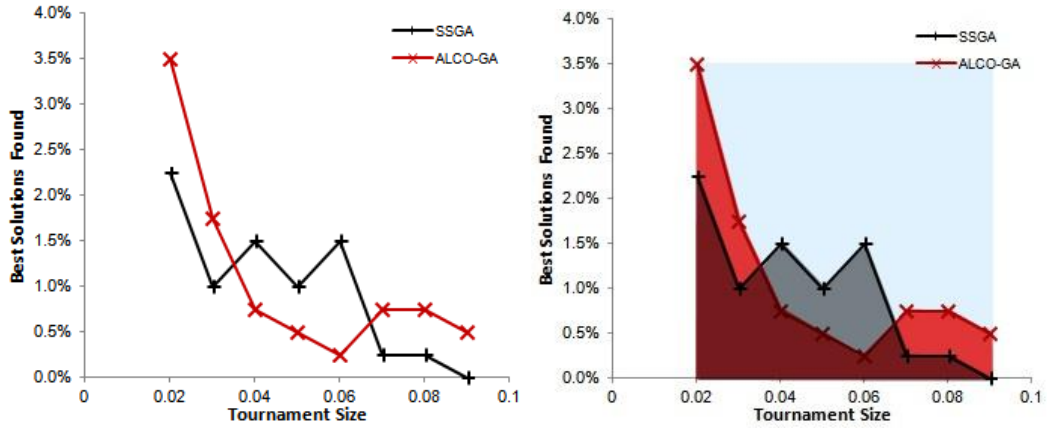


Figure 5-9. Average percentage of best known solutions found at varying tournament sizes for the Hanoi problem - SSGA & ALCO-GA

The following figure shows the comparison between the two algorithms in terms of the average evaluations taken for the each algorithm to achieve a feasible solution at each tournament size. It is clear from these results that ALCO-GA finds the feasible solution space in many fewer evaluations than the SSGA for every tournament size. As observed previously, as the tournament size is increased the number of evaluations taken for the SSGA to achieve a

feasible solution decrease, however it is observed that although ALCO-GA displays this same behaviour it is much less pronounced.

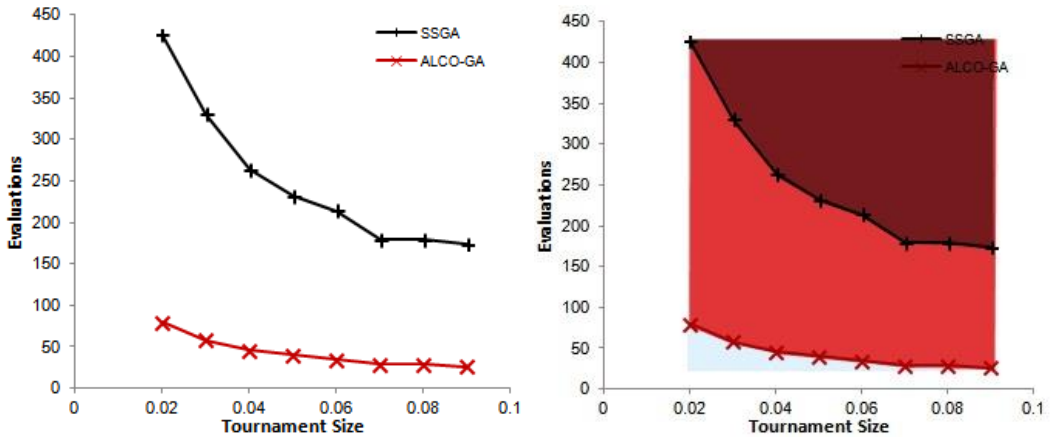


Figure 5-10. Average evaluations taken to achieve a feasible solution at varying tournament sizes for the Hanoi problem - SSGA & ALCO-GA

The figure below presents the normalised area under/above the curve to enable to direct comparison between the SSGA and ALCO-GA for the range of tournament sizes tested. As can be seen, the performance of ALCO-GA is less sensitive to tournament size changes than the SSGA especially in terms of fitness, network cost and the number of evaluations taken to achieve a feasible solution.

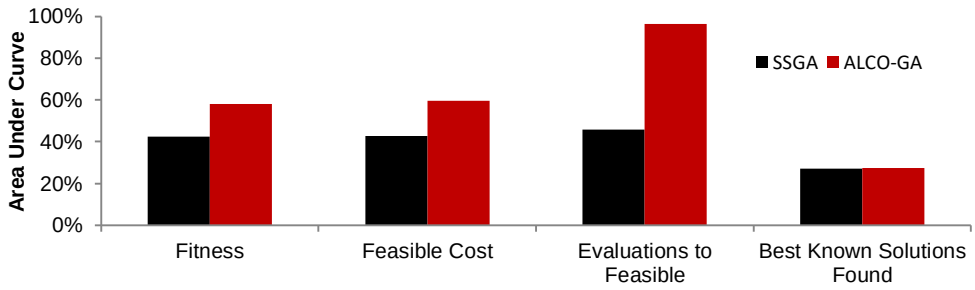


Figure 5-11. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Hanoi problem - SSGA & ALCO-GA

The following set of figures present the performance of the SSGA and ALCO-GA for a range of pipe mutation rates on the Hanoi problem. The following figure displays the average best fitness and feasible network costs for a set of mutation probabilities ranging from 0.029 to 0.235. It can be seen from

these results that the SSGA achieves better average results than ALCO-GA between ~0.06 and ~0.017 however at rates of mutation above and below this range the performance of the SSGA falls off more aggressively than that of ALCO-GA.

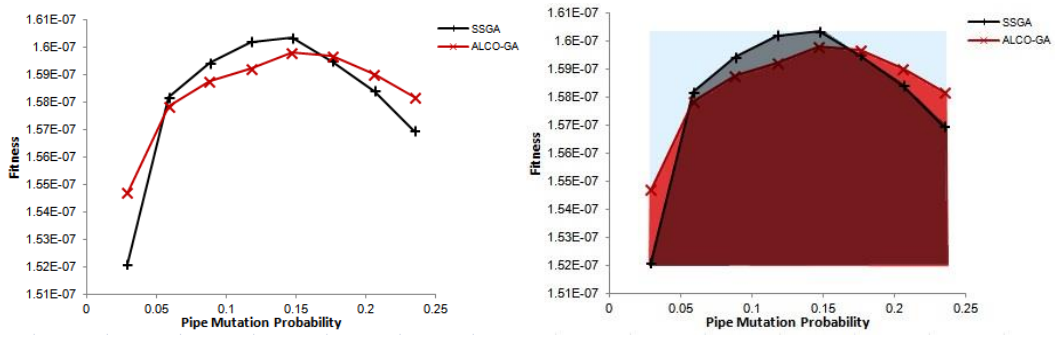


Figure 5-12. Average fitness at varying mutation rates for the Hanoi problem – SSGA & ALCO-GA

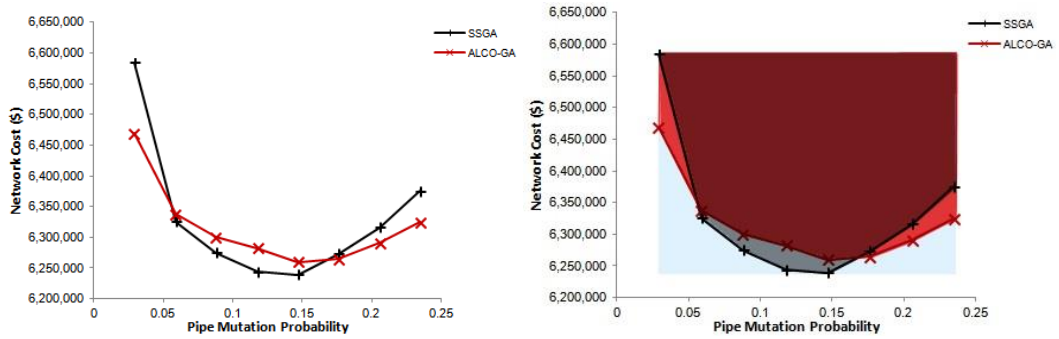


Figure 5-13. Average feasible network cost at varying mutation rates for the Hanoi problem - SSGA & ALCO-GA

The figure below shows the average percentage of solutions which achieve the best known solution for the Hanoi problem ($6,081,220). It can be seen that the performance of both algorithms is relatively similar in this instance however it can be observed that although the SSGA achieves the best known solution more at lower mutation rates, ALCO-GA generally outperforms the SSGA to a greater extent at higher mutation rates.

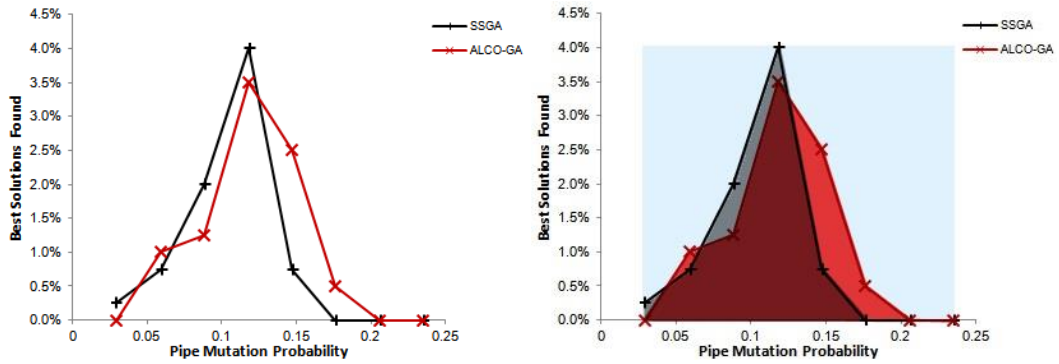


Figure 5-14. Average percentage of best known solutions found at varying mutation rates for the Hanoi problem - SSGA & ALCO-GA

The following figure presents the comparison between the two algorithms in terms of the average evaluations taken for the each algorithm to achieve a feasible solution at each mutation probability. As was observed previously from the SSGA parameter tuning experiments as the mutation rate is increased the number of evaluations required to achieve a feasible solution also increases for the SSGA. This however is not the case with ALCO-GA where as the mutation rate increases the number of evaluations taken to find the feasible solution space decreases slightly. This behaviour is expected as the heuristic which primarily promotes feasible solutions is being applied more at higher rates of mutation.

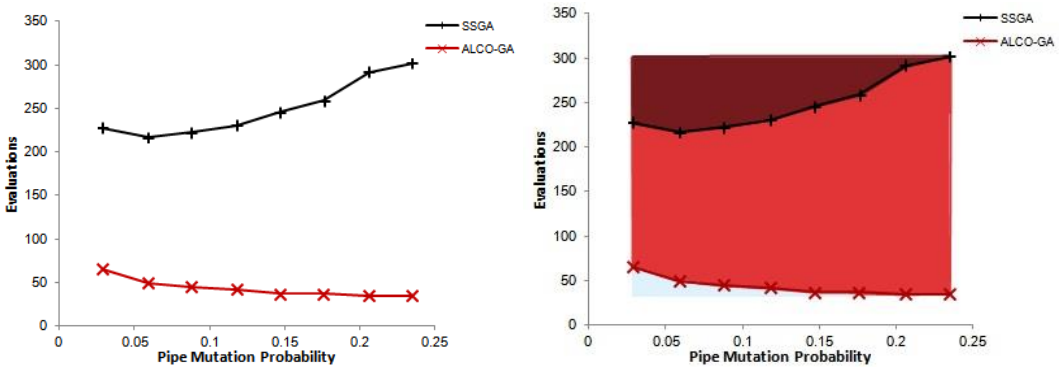


Figure 5-15. Average evaluations taken to achieve a feasible solution at varying mutation rates for the Hanoi problem - SSGA & ALCO-GA

The figure below presents the normalised area under/over the curves presented above to enable to direct comparison between the SSGA and ALCO-GA for the range of mutation probabilities tested. In this case the sensitivity to the mutation rate is very similar for fitness, feasible network cost and best

known solution achieved with ALCO-GA performing slightly better. However as pointed out in the previous figure, ALCO-GA drastically out performs the SSGA in terms of evaluations taken to achieve a feasible solution.

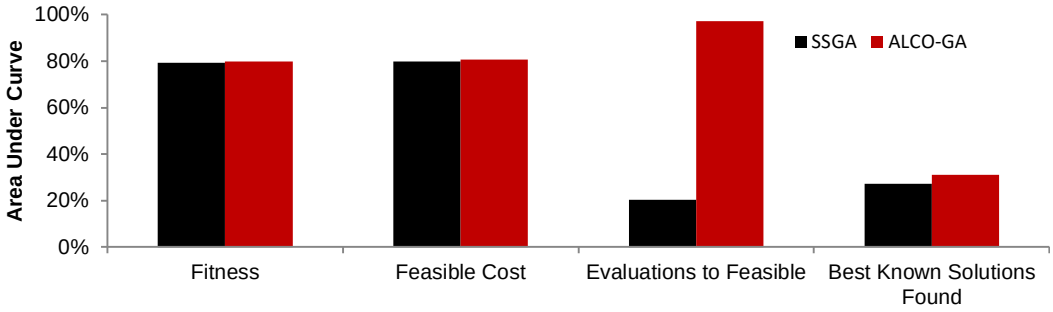


Figure 5-16. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Hanoi problem - SSGA & ALCO-GA

5.1.2.2.2 Two Loop

The following figures present the normalized area under the curve for the tournament size and mutation probability on the Two Loop problem. The results for the tournament size show that ALCO-GA is far less sensitive to changes in the tournament size than the SSGA in the case of solution quality and number of best known solutions found. The difference in sensitivity to mutation rate changes between the two algorithms is diminished compared with tournament size changes; however ALCO-GA still outperforms the SSGA in terms of solution quality and the number of best know solutions found. It should be noted that the initial population (identical for both ALCO-GA & SSGA) contained feasible solutions therefore both algorithms achieved feasibility in 0 fitness evaluations.

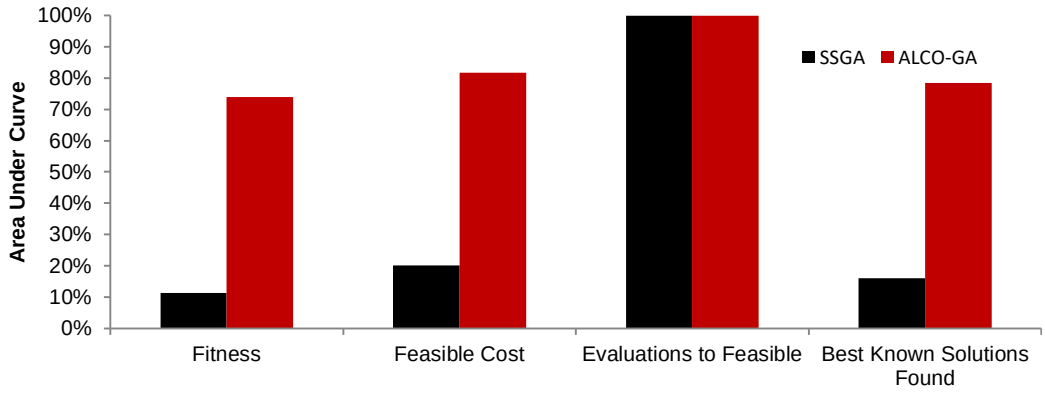


Figure 5-17. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Two Loop problem - SSGA & ALCO-GA

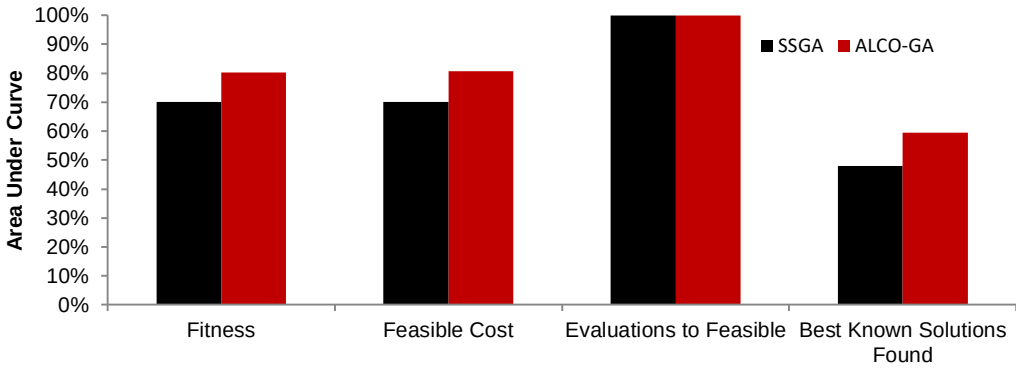


Figure 5-18. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Two Loop problem - SSGA & ALCO-GA

5.1.2.2.3 New York Tunnels

The results for the New York Tunnels problem are presented in the following figures. ALCO-GA shows to be a lot less sensitive to tournament size changes than the SSGA when it comes to solution fitness, feasible network cost and the number of best known solutions obtained. The mutation rate results reflect the same behaviour although the difference between the algorithms is much less pronounced. The parallel expansion nature of the New York Tunnels problem lends itself to the production of feasible solutions and in this case feasible solutions are found in the randomly generated starting populations. This results in the evaluations taken to achieve a feasible solution to 0 for both methods and is therefore reflected in the 100% area displayed in the figures.

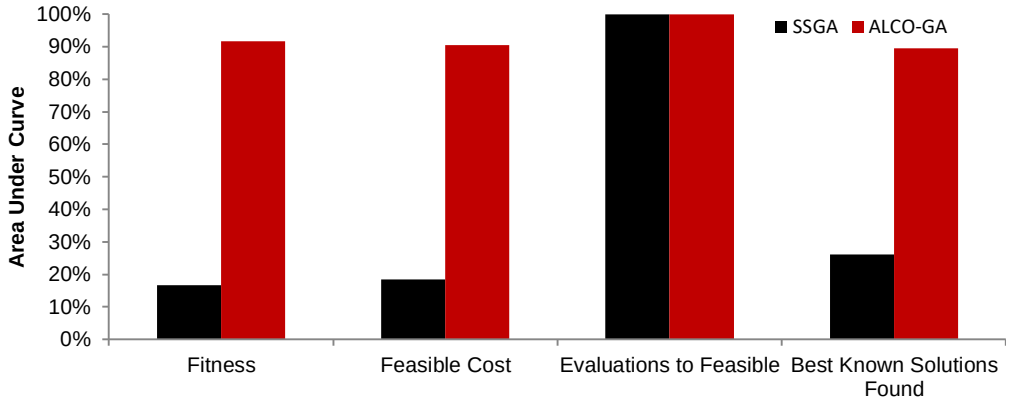


Figure 5-19. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the New York Tunnels problem - SSGA & ALCO-GA

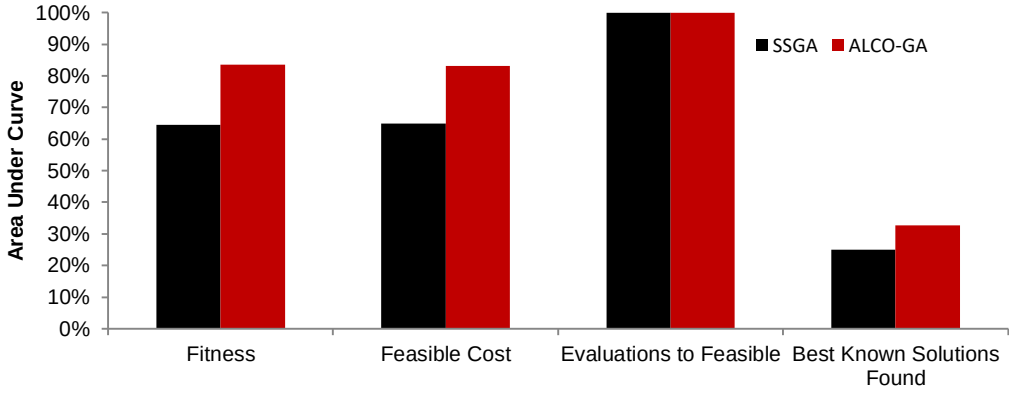


Figure 5-20. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the New York Tunnels problem - SSGA & ALCO-GA

5.1.2.2.4 Foss Poly 1

The robustness results for the Foss Poly network design problem are displayed below. As with the previous experiments the sensitivity to variations in tournament size of ALCO-GA are less than that of the SSGA. The difference in sensitivity between the two algorithms for the mutation rate results is similar to that of the tournament size experiment however as was found with previous problems the difference in sensitivity to mutation rate is diminished compared with tournament size.

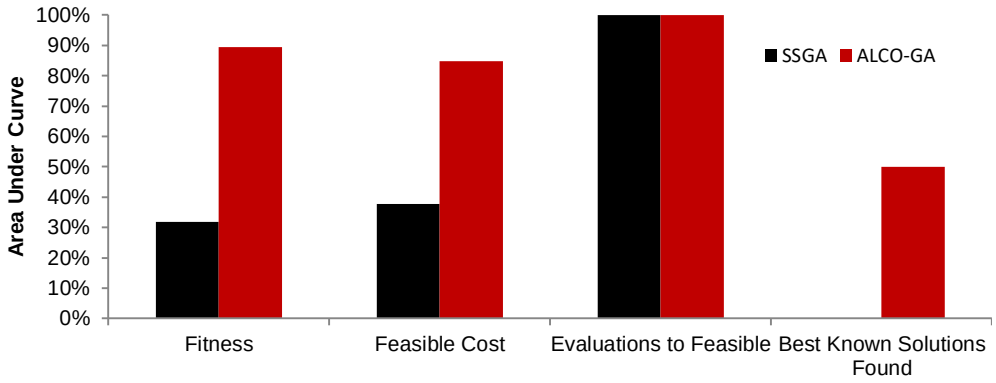


Figure 5-21. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Foss Poly 1 problem - SSGA & ALCO-GA

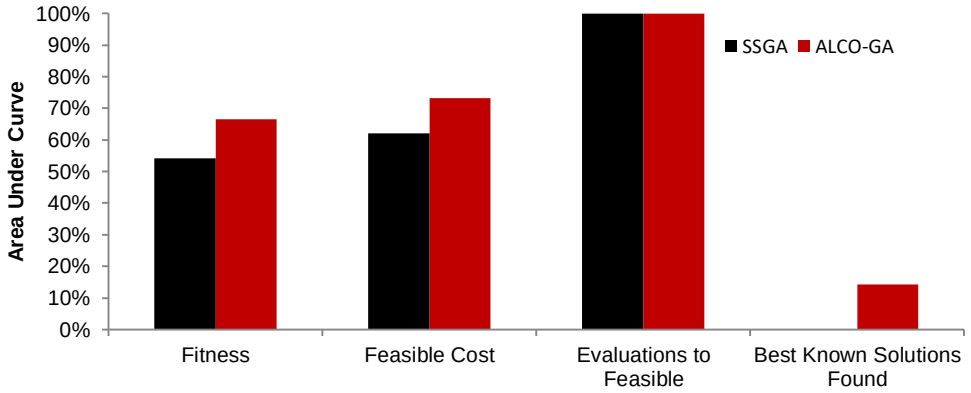


Figure 5-22. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Foss Poly 1 problem - SSGA & ALCO-GA

5.1.2.2.5 Modena

The following figures present the robustness results for the Modena network design problem. As can be seen, the SSGA tends to be more sensitive to changes in the tournament size than ALCO-GA. ALCO-GA on average achieves a better result in terms of solution fitness, feasible network cost and the number of best known solutions found than the SSGA. However, the mutation rate sensitivity results show the two algorithms perform very similarly with ALCO-GA only marginally outperforming the SSGA in terms of overall solution quality.

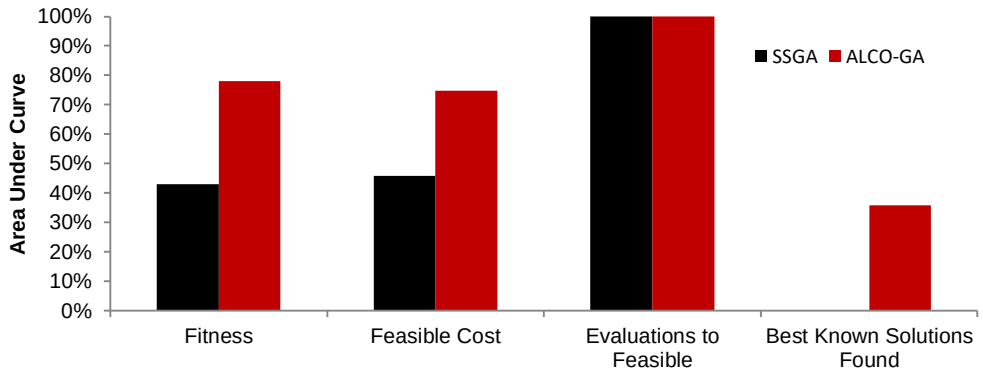


Figure 5-23. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Modena problem - SSGA & ALCO-GA

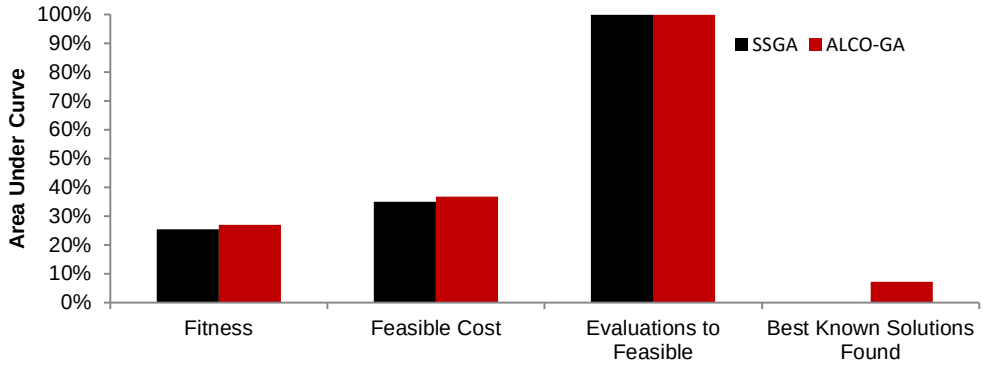


Figure 5-24. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Modena problem - SSGA & ALCO-GA

5.1.2.2.6 Network A

Due to the complexity of the Network A water distribution network design problem and the resultant run time this set of experiments were scaled down compared with the previously presented problems. For this experiment the tournament size ranged between 2% and 8% in 2% intervals and mutation probability ranged between 0.0016 and 0.0128 in 0.0016 intervals. For each parameter value combination SSGA and ALCO-GA were run 10 times, each for 100,000 objective function evaluations utilizing a different randomly seeded starting population for each run. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size was fixed at 100 individuals, single-point crossover was utilized and a conditional worst individual replacement strategy was implemented.

Figure 5-25 shows the average fitness, feasible network cost and evaluations to feasible results at each of the 4 selected tournament sizes for the Network A problem. Looking at the fitness and feasible network cost plots, ALCO-GA is seen to achieve better results than the SSGA at all tournament sizes. The results suggest that the SSGA achieves best performance at a tournament size of 0.06N whilst ALCO-GA requires a larger tournament size to perform optimally. The most striking plot from this figure is the evaluations to feasible chart which shows ALCO-GA achieving the feasible solution space in far fewer evaluations than the SSGA at all tournament sizes.

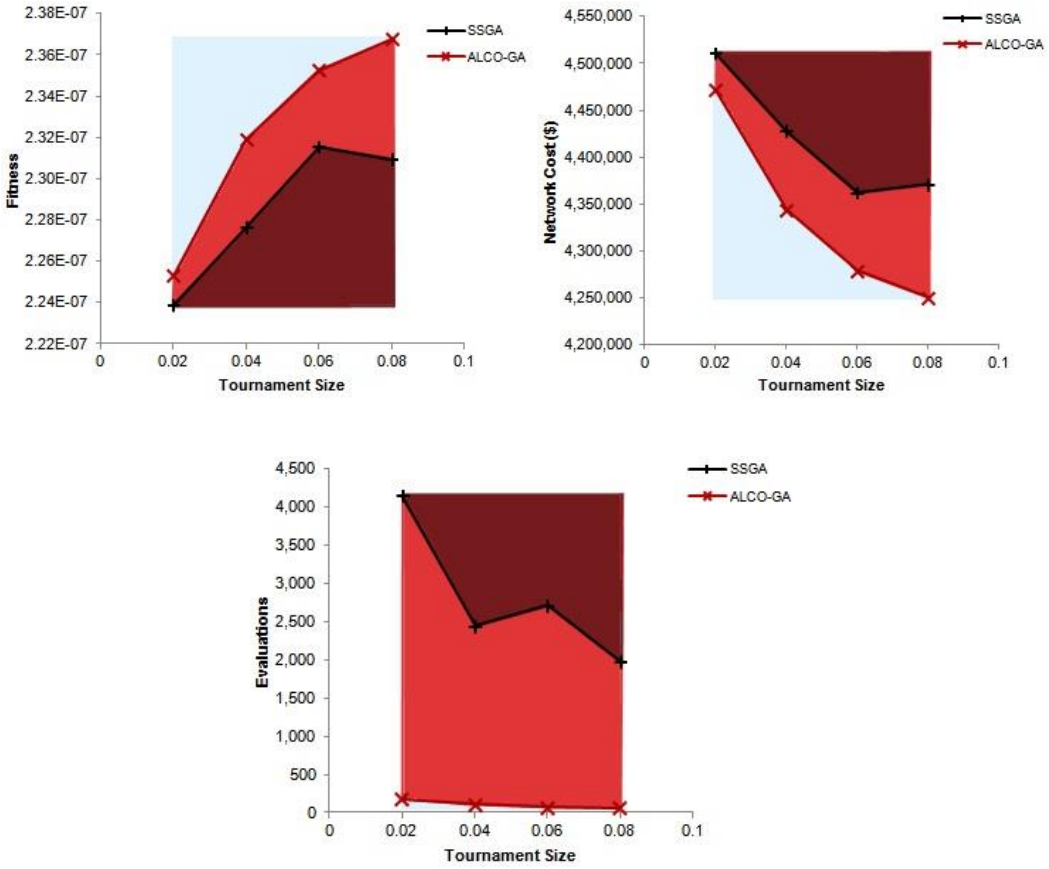


Figure 5-25. Average fitness (top left), feasible network cost (top right) & evaluations taken to achieve a feasible solution (bottom) at varying tournament sizes for the Network A problem – SSGA & ALCO-GA

Both algorithms are seen to be affected by varying tournament size; evaluations to a feasible solution decreases as the tournament size increases however it is clear that the performance of the SSGA is influenced to a much greater degree than ALCO-GA. This is reflected in Figure 5-26, which displays the normalized

resultant area for the fitness, network cost and evaluation taken to reach a feasible solution.

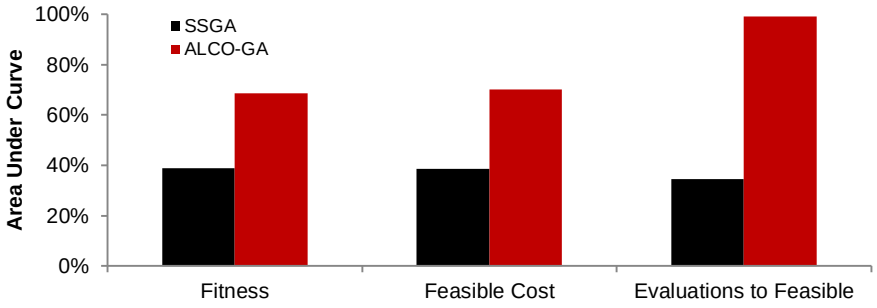


Figure 5-26. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying tournament sizes for the Network A problem - SSGA & ALCO-GA

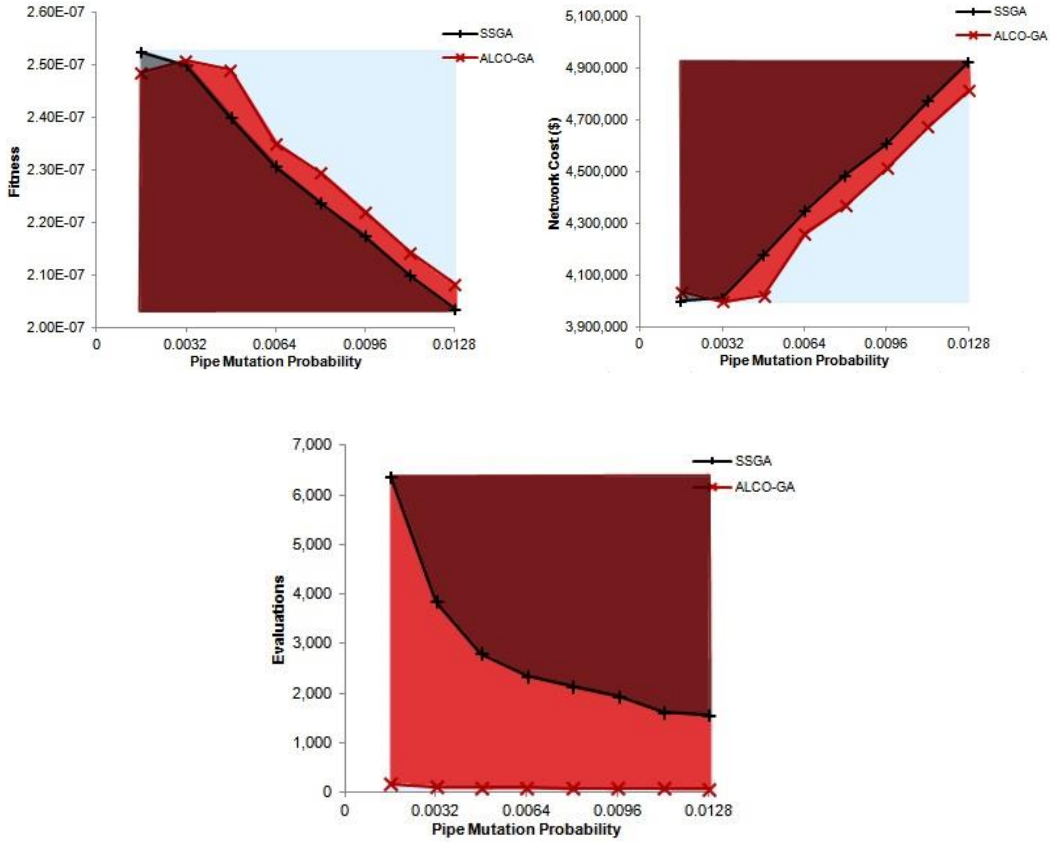


Figure 5-27. Average fitness (top left), feasible network cost (top right) & evaluations taken to achieve a feasible solution (bottom) at varying mutation rates for the Network A problem – SSGA & ALCO-GA

Figure 5-27 shows the average fitness, feasible network cost and evaluations to feasible results at each of the 8 selected mutation rates for the Network A problem. As can be seen, ALCO-GA achieves an average solution quality higher than that of the SSGA at all mutation rates tested apart from the lowest mutation probability of 0.0016. Both algorithms display a general decrease in performance as the mutation rate is increased however the inclusion of the heuristic based mutation seems to almost offset this decline in performance resulting in a decreased sensitivity to mutation rate compared to the SSGA. There is a drastic difference in performance between the two algorithms in terms of evaluations taken to find the feasible solution space with ALCO-GA, especially at lower mutation rates.

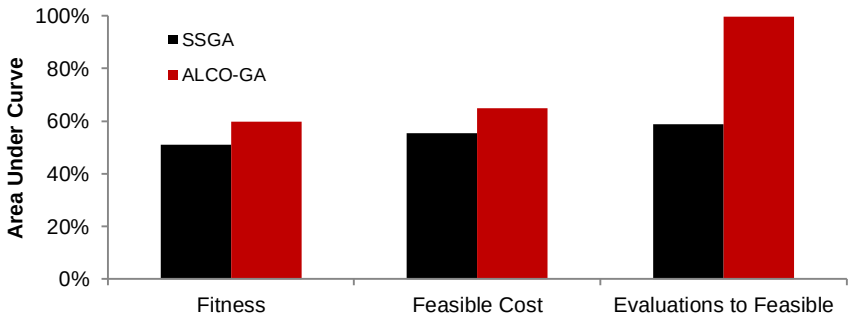


Figure 5-28. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying mutation rates for the Network A problem - SSGA & ALCO-GA

5.1.2.2.7 Network B

As with the previous problem (Network A), due to the complexity of the Network B problem and the resultant run time this set of experiments were scaled down compared with the previously presented problems. For this experiment the tournament size ranged between 2% and 8% in 2% intervals and mutation probability ranged between 0.0008 and 0.0064 in 0.0008 intervals. For each parameter value combination SSGA and PSGA were run 10 times, each for 100,000 objective function evaluations utilizing a different randomly seeded starting population for each run. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size was fixed at 100 individuals, single-point crossover was utilized and a conditional worst individual replacement strategy was implemented.

Figure 5-29 shows the average fitness, feasible network cost and evaluations to feasible results at each of the 4 selected tournament sizes for the Network B problem. Looking at the fitness and feasible network cost plots, ALCO-GA is seen to achieve better results than the SSGA at all tournament sizes in terms of feasible network cost. The results suggest that the SSGA achieves best performance at a tournament size of 0.08N whilst PSGA achieves optimal performance at a tournament size of 0.06N. Both algorithms are seen to be affected by varying tournament size; evaluations to a feasible solution decreases as the tournament size increases. The extent at which ALCO-GA outperforms the SSGA in terms of evaluation to achieve a feasible solution is clear from this figure.

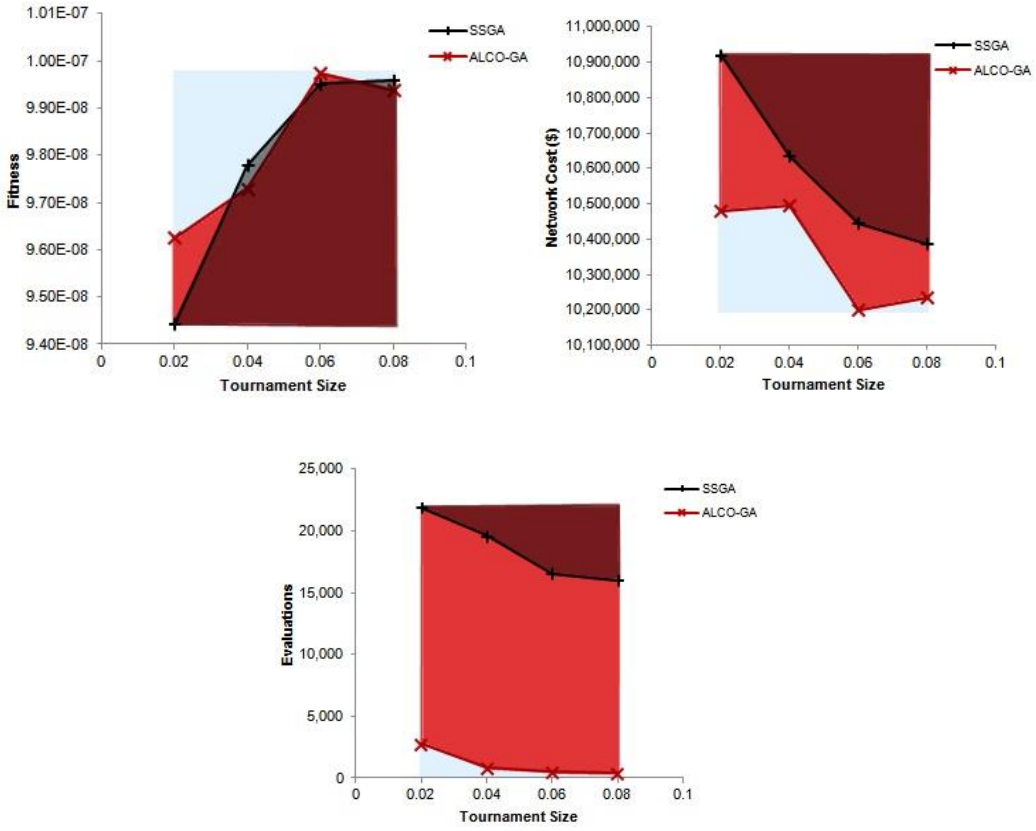


Figure 5-29. Average fitness (top left), feasible network cost (top right) & evaluations taken to achieve a feasible solution (bottom) at varying tournament sizes for the Network B problem – SSGA & ALCO-GA

These results are reflected in Figure 5-30, which displays the normalized resultant area for the fitness, network cost, evaluation taken to reach a feasible solution and pipe smoothing violations.

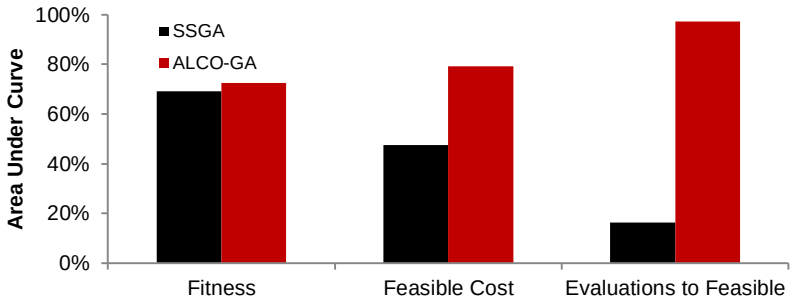


Figure 5-30. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying tournament sizes for the Network B problem - SSGA & ALCO-GA

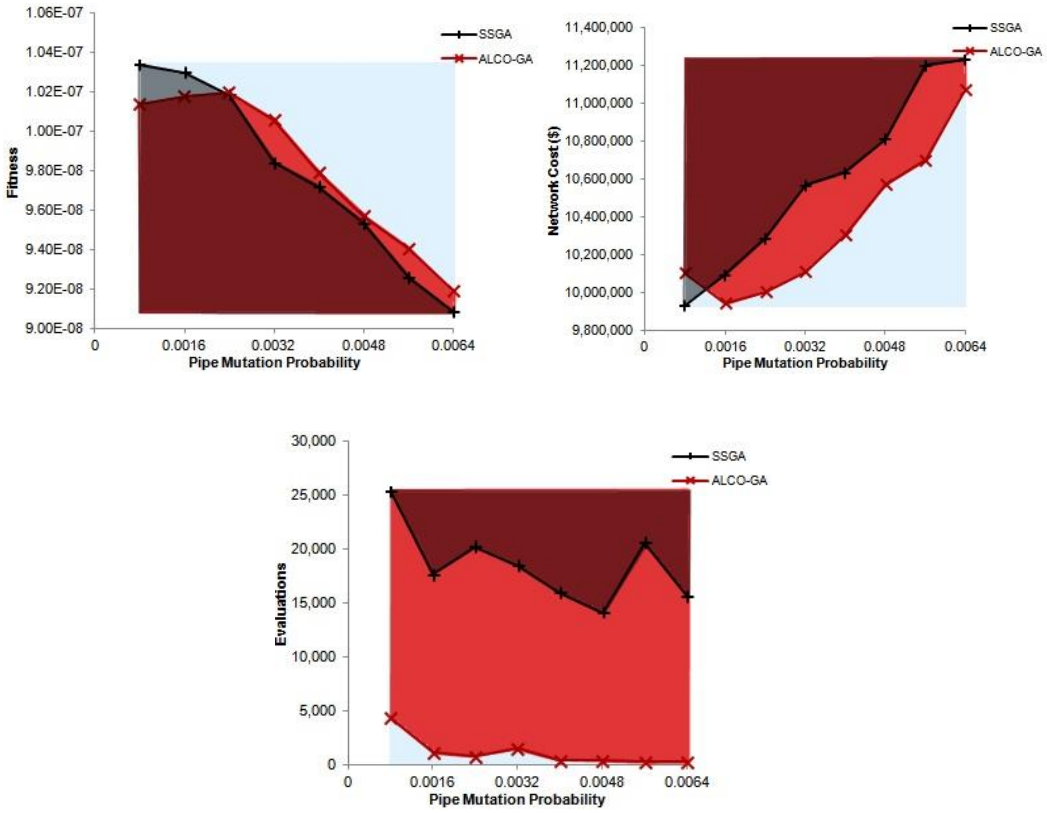


Figure 5-31. Average fitness (top left), feasible network cost (top right) & evaluations taken to achieve a feasible solution (bottom) at varying mutation rates for the Network B problem – SSGA & ALCO-GA

Figure 5-31 shows the average fitness, feasible network cost and evaluations to feasible results at each of the 8 selected mutation rates for the Network B problem. As can be seen, ALCO-GA achieves an average solution quality higher than that of the SSGA at all mutation rates tested apart from the lowest mutation probability of 0.0008.

Both algorithms display a general decrease in performance as mutation rate is increased however the inclusion of the pipe smoothing heuristic based mutation seems to diminish this decline in performance resulting in a decreased sensitivity to mutation rate compared to the SSGA. There is a significant difference in performance between the two algorithms in terms of pipe evaluations taken to achieve a feasible solution. It is also observed that as the mutation rate is increased so does the number of evaluations required to reach the feasible solution space.

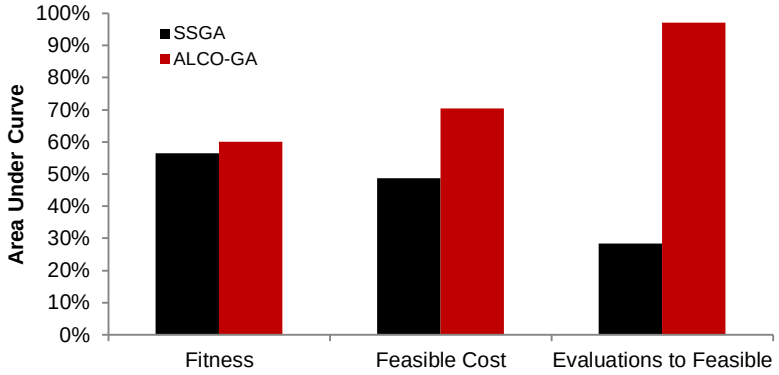


Figure 5-32. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying mutation rates for the Network B problem - SSGA & ALCO-GA

5.1.3 Conclusion

An Adaptive Locally Constrained Genetic Algorithm for WDN design optimisation problems has been described. The algorithm is based on a standard Steady State Genetic Algorithm (SSGA) with the addition of an adaptive heuristic based mutation operator which utilises hydraulic head information to guide the search of the algorithm into the feasible solution space. The algorithm has been found to perform well on a range of WDNs of varying complexity compared to the SSGA; sometimes not only finding a better solution but achieving it in less solution evaluations. It should be noted that the performance gains of ALCO-GA are largely located towards the initial stages of the search. The algorithm is able to generate feasible solutions more quickly than the SSGA and so in applications where the number of function evaluations is limited, i.e. where large real-world networks are optimised, ALCO-GA can provide valid designs much earlier in the evolutionary process. In particular,

very complex networks with large objective functions will require an algorithm that is able to generate feasible solutions early in the optimisation process. ALCO-GA has demonstrated this property, and whilst these feasible solutions in the early stages of the search are reasonably far removed from being near-optimal, they do at least meet the engineering criteria. Additional experimentation has shown that the ALCO-GA is more robust to parameter settings than the SSGA meaning that if extensive parameter tuning is not feasible due to the complexity of the network, the rule-of-thumb parameters are more likely to function well with ALCO-GA than the SSGA. In short, ALCO-GA would be the algorithm of choice for an engineer wanting to optimise a large complex real-world network for the first time.

Although ALCO-GA has shown to improve the performance of these benchmark networks, real world networks will include further complications above and beyond the single period gravity-fed nature of the benchmark problems presented here, notably the inclusion of extended period simulations, pumps, pressure reducing valves, flow control valves, water treatment works and other network infrastructure. The ALCO-GA heuristic could be incorporated into any optimisation that sizes pipes amongst other infrastructure in water distribution network design. However, the heuristic is currently restricted to pipe sizing and whilst pipes are among the most numerous assets represented by decision variables in a water distribution network, the influence of these additional infrastructure types is likely to weaken the effect of the heuristic. That said, there remains a good deal of scope in implementing ALCO-like rules within an EA for other elements of the network. Tanks for instance could be upsized if they were found to be overtopping and downsized if levels did not change sufficiently; pressure reducing valve settings could be modified according to pressure at critical downstream nodes and pump settings could be incremented/decremented according to downstream head. ALCO-GA demonstrates that the principle of localised modification within a water distribution network can be a powerful addition to the global search of the evolutionary algorithm and future work could focus on developing a bespoke 'intelligent' operator for each asset type to achieve similar results on real-world networks.

5.2 Pipe Smoothing GA Experimentation

Following the development of the Pipe Smoothing Genetic Algorithm (PSGA) (4.4) this section sets out to explore the performance of PSGA on a range of water distribution network design problems of differing structure, complexity and size. The initial experimentation presented in this section compares PSGA and the SSGA in terms of overall solution quality achieved and also solution feasibility. The final section explores the robustness of PSGA and SSGA when different parameters are varied.

5.2.1 PSGA Tuned Performance Experiments

5.2.1.1 Experimental Setup

The set of experiments presented in this section compare PSGA and the SSGA, applying them to a variety of different water distribution network design problems, a list and brief description of each of the networks is available in section 3.3. As stated before the collection of water distribution network design problems were selected for their structure, complexity and size variations, providing a diverse benchmark set to enable extensive evaluation of PSGA. The optimal parameter values obtained from the single-objective genetic algorithm parameter tuning section (0) of this chapter were used for all instances of the SSGA and PSGA in this set of experiments. PSGA and the SSGA were run 50 times with the exception of the Network A & B problems which were only run 10 times, each for 100,000 objective function evaluations. To ensure a fair comparison between the SSGA and PSGA a list of 50 (10 for Network A) randomly generated integers were generated for the purpose of seeding the random number function at the beginning of each of the 50 runs. The same list of random number seeds was used for both PSGA and the SSGA ensuring the same starting populations for each run. The SSGA and PSGA were implemented in C++ and all of the following experiments were run on an Intel Core i7-4770K 3.5GHz PC.

5.2.1.2 Results

5.2.1.2.1 Two Loop

The following results show the performance comparison between SSGA and PSGA for the Two Loop water distribution network design problem. Table 5-13 presents the best performing solutions for each algorithm from the 50 runs. It can be seen that both algorithms find the best known feasible solution for this problem ($419,000). Both algorithms produce a feasible solution in the initial population, an important thing to note in regard to this is that due to the small size of the problem and the pipe diameter decision set a randomly generated solution has a very high probability to produce a solution which satisfies the hydraulic constraints of the problem, also as stated previously the starting populations of both algorithms are identical as they are generated using the same random number seed set.

Table 5-13. Best single run results for the Two Loop problem - SSGA & PSGA comparison

	Best Single Run		
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**
SSGA	**2.387E-06**	**419,000**	0
PSGA	**2.387E-06**	**419,000**	0

Table 5-14 shows the mean results for the algorithms for the Two Loop problem. As can be seen, PSGA achieves a better fitness and feasible solution cost than the SSGA also obtaining a lower standard deviation for the best solutions from the 50 runs over the 100,000 solution evaluations. In this set of experiments all of the solutions in the initial population were hydraulically feasible, resulting in the 0 evaluations to feasible result for both algorithms.

Table 5-14. Mean results for the Two Loop problem – SSGA & PSGA comparison

	Average							
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**	**Pipe Smoothing Violations**	**Pipe Smoothing Violations Standard Deviation**
SSGA	2.383E-06	6.301E-09	419,740	1,426	**0.0**	**0.0**	**0.00**	**0.00**
PSGA	**2.385E-06**	**2.517E-09**	**419,260**	**443**	**0.0**	**0.0**	**0.00**	**0.00**

As can be seen from Figure 5-33, PSGA displays faster convergence than the SSGA, achieving a near optimal solution at approximately 60,000 solution evaluations. In terms of solution feasibility, both algorithms start at an average of 10% population feasibility and quickly rise in the first 200 evaluations with PSGA and SSGA rising to approximately 40% feasibility. Interestingly population feasibility becomes reduced for both algorithms in the next 10,000 evaluations, however at this point population feasibility starts to rise significantly for ALCO-GA leaving the SSGA to remain in the 25%-30% range.

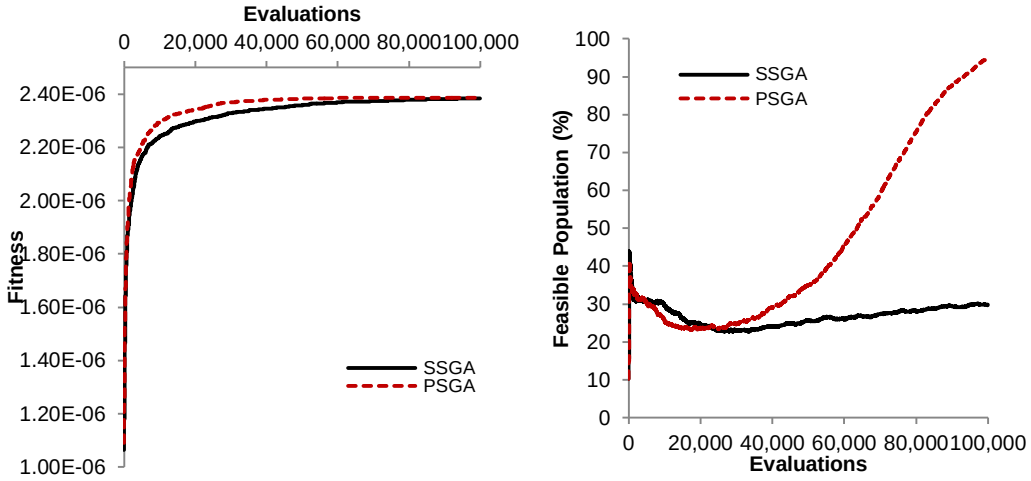


Figure 5-33. Graphs showing the average best solution fitness (left) & average percentage of feasible solutions present in the population (right) for the Two Loop problem – SSGA & PSGA comparison

5.2.1.2.2 Foss Poly 1

The following results presented here are for the Foss Poly 1 water distribution network design problem. It can be seen from Table 5-15 that following the allotted 100,000 solution evaluations PSGA finds a better solution for the 50 runs than the SSGA, achieving a feasible network cost $7,737 less than the best solution found by the SSGA. As with the previous Two Loop experiment, this problem also offers a high probability that a solution in the randomly generated starting population is feasible, and this being the case in this instance where on average approximately 30% of feasible solutions were generated in the starting population. Both algorithms also achieved perfectly smooth networks.

Table 5-15. Best single run results for the Foss Poly 1 problem - SSGA & PSGA comparison

	Best Single Run			
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**	**Pipe Smoothing Violations**
SSGA	2.496E-05	40,063	0	0
PSGA	**3.105E-05**	**32,326**	0	0

The average results for the 50 runs of both algorithms (Table 5-16) shows that PSGA achieves a better average solution fitness and feasible network cost than the SSGA. Again, due to the presents of feasible solution in the starting population of all of the runs, the average evaluations to the feasible search space is 0. PSGA also achieves a smoother network solution on average compared with the SSGA.

Table 5-16. Mean results for the Foss Poly 1 problem – SSGA & PSGA comparison

	Average							
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**	**Pipe Smoothing Violations**	**Pipe Smoothing Violations Standard Deviation**
SSGA	2.218E-05	**1.184E-06**	47,030	3,001	0.0	0.0	2.60	1.40
PSGA	**2.827E-05**	1.690E-06	**36,361**	**2,391**	0.0	0.0	**0.82**	**0.98**

Figure 5-34 displays (top) the extent at which PSGA outperforms the SSGA in terms of average solution fitness during the search of both algorithms for the Foss Poly 1 problem. From these results it is clear that PSGA achieves better average solution fitness than the SSGA at every stage of the 100,000 evaluation search. Both algorithms achieve similar population feasibility throughout all stages of the search. On average PSGA produces smoother networks than the SSGA throughout the entire search of the algorithms.

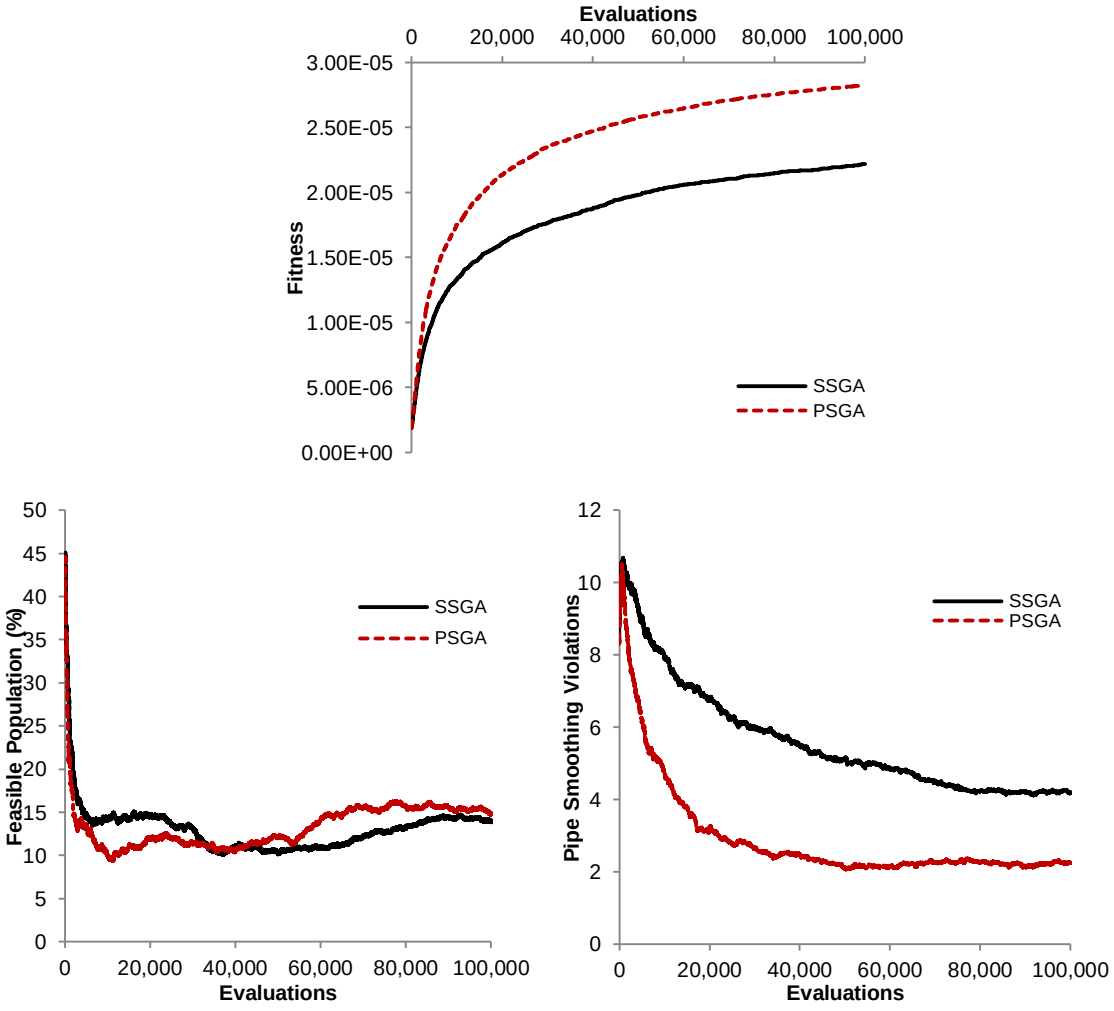


Figure 5-34. Graphs showing the average best solution fitness (top), average percentage of feasible solutions present in the population (left) & average pipe smoothing violations for the Foss Poly 1 problem – SSGA & PSGA comparison

5.2.1.2.3 New York Tunnels

The following section presents the results for both algorithms for the New York Tunnels network, a parallel expansion problem. Table 5-17 shows that both algorithms achieve the same best single run results, reaching the best know solution for this formulation of the problem ($38,637,600). Also PSGA produces a slightly smoother network solution to that of the SSGA.

Table 5-17. Best single run results for the New York Tunnels problem - SSGA & PSGA comparison

	Best Single Run			
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**	**Pipe Smoothing Violations**
SSGA	**2.615E-08**	**38,637,600**	**0**	4
PSGA	**2.615E-08**	**38,637,600**	**0**	**2**

The following mean results table shows the SSGA slightly outperforms PSGA in terms of solution fitness and feasible network cost although the gains in performance the SSGA has over PSGA are minimal compared with the total cost of the network. As with some of the previous problems, the nature of the New York Tunnels parallel expansion problem highly promotes feasibility which in turn increases the probability that randomly generated solutions will satisfy the hydraulic constraints of the problem. In this case we see from the results that there are solutions present in each of the 50 starting populations that are feasible, hence resulting in the 0 evaluations to feasible result for each algorithm. Also the PSGA on average produces smoother networks than the SSGA over the 100,000 evaluation search.

Table 5-18. Mean results for the New York Tunnels problem – SSGA & PSGA comparison

	Average							
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**	**Pipe Smoothing Violations**	**Pipe Smoothing Violations Standard Deviation**
SSGA	**2.597E-08**	**2.271E-10**	**38,919,500**	**355,019**	**0.0**	**0.0**	8.92	**4.77**
PSGA	2.576E-08	7.186E-10	39,214,200	1,167,922	**0.0**	**0.0**	**6.64**	5.17

Figure 5-35 shows that PSGA converges considerably faster than the SSGA in the early stages of the search, outperforming the SSGA for the first ~30,000 evaluations whilst sustaining a competitive solution quality throughout the remainder of the search. The second graph (bottom left) indicates that the average starting population percent feasibility is just under 90% however as is shown in the figure, the population feasibility of both algorithms decreases rapidly in the early stages of the search. This behaviour is somewhat expected as it adheres to the observed performance of the algorithms when applied to problems with high initial solution feasibility.

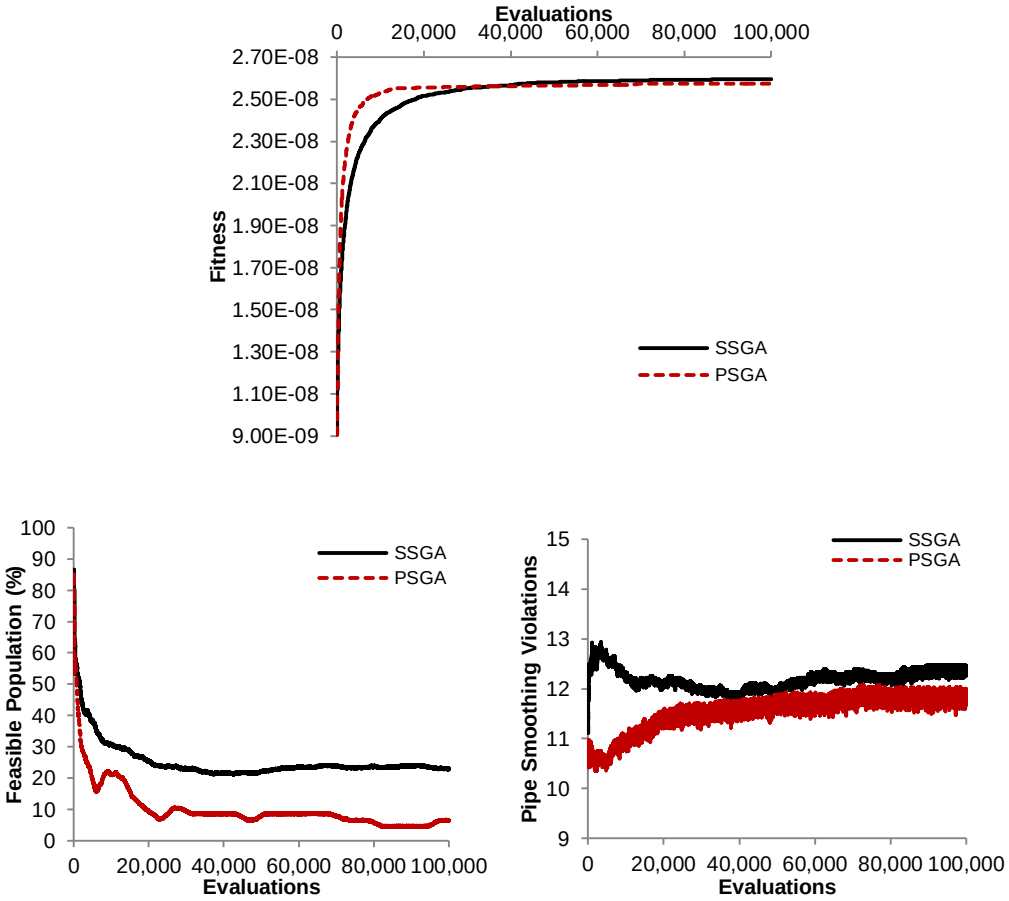


Figure 5-35. Graphs showing the average best solution fitness (top), average percentage of feasible solutions present in the population (left) & average pipe smoothing violations for the New York Tunnels problem – SSGA & PSGA comparison

It can be seen that PSGA reduces solution feasibility very aggressively at the beginning of the search whilst reducing the cost of the network and increasing solution fitness. PSGA on average produces a smoother network

than the SSGA throughout the search of the algorithms, however the PSGA displays a greater variation in smoothness than the SSGA.

5.2.1.2.4 Modena

The following results are for the Modena water distribution network design problem, a large real-world network with multiple reservoir sources. Table 5-19 shows the best single run results for both algorithms after the allotted 100,000 fitness evaluations. As can be seen from the table, the PSGA finds a solution over $1m cheaper than the SSGA. This result is surprising as the improvement is very significant. Such a drastic improvement in performance could be due to the multiple source nature of the problem which naturally lends itself to the influence of the pipe smoothing heuristic. As with the previous two problems presented in this results section, the makeup of this problem gives a high probability that a feasible solution can be randomly generated hence the 0 evaluations taken to find a feasible solution for both algorithms. PSGA also achieves a smoother network solution than the SSGA.

Table 5-19. Best single run results for the Modena problem - SSGA & PSGA comparison

	Best Single Run			
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**	**Pipe Smoothing Violations**
SSGA	1.762E-07	5,898,900	**0**	91
PSGA	**2.117E-07**	**4,874,240**	**0**	**76**

Table 5-20 presents the average results for the Modena problem and shows that PSGA outperforms the SSGA in terms of solution quality and solution variation for the 50 runs; PSGA achieves a mean feasible network cost over $1m less than the SSGA and a smaller feasible cost standard deviation. As stated previously the Modena problem lends itself to producing feasible solutions, this is demonstrated by the 0 mean evaluations to feasible result presented in the table below, thus implying that at least one feasible solution was generated in each of the 50 runs performed in this experiment set. PSGA on average produces less pipe smoothing violations than the SSGA.

Table 5-20. Mean results for the Modena problem – SSGA & PSGA comparison

	Average							
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**	**Pipe Smoothing Violations**	**Pipe Smoothing Violations Standard Deviation**
SSGA	1.617E-07	4.253E-09	6,385,570	167,052	**0.0**	**0.0**	96.48	2.21
PSGA	**2.006E-07**	**3.938E-09**	**5,171,050**	**148,699**	**0.0**	**0.0**	**81.88**	**2.14**

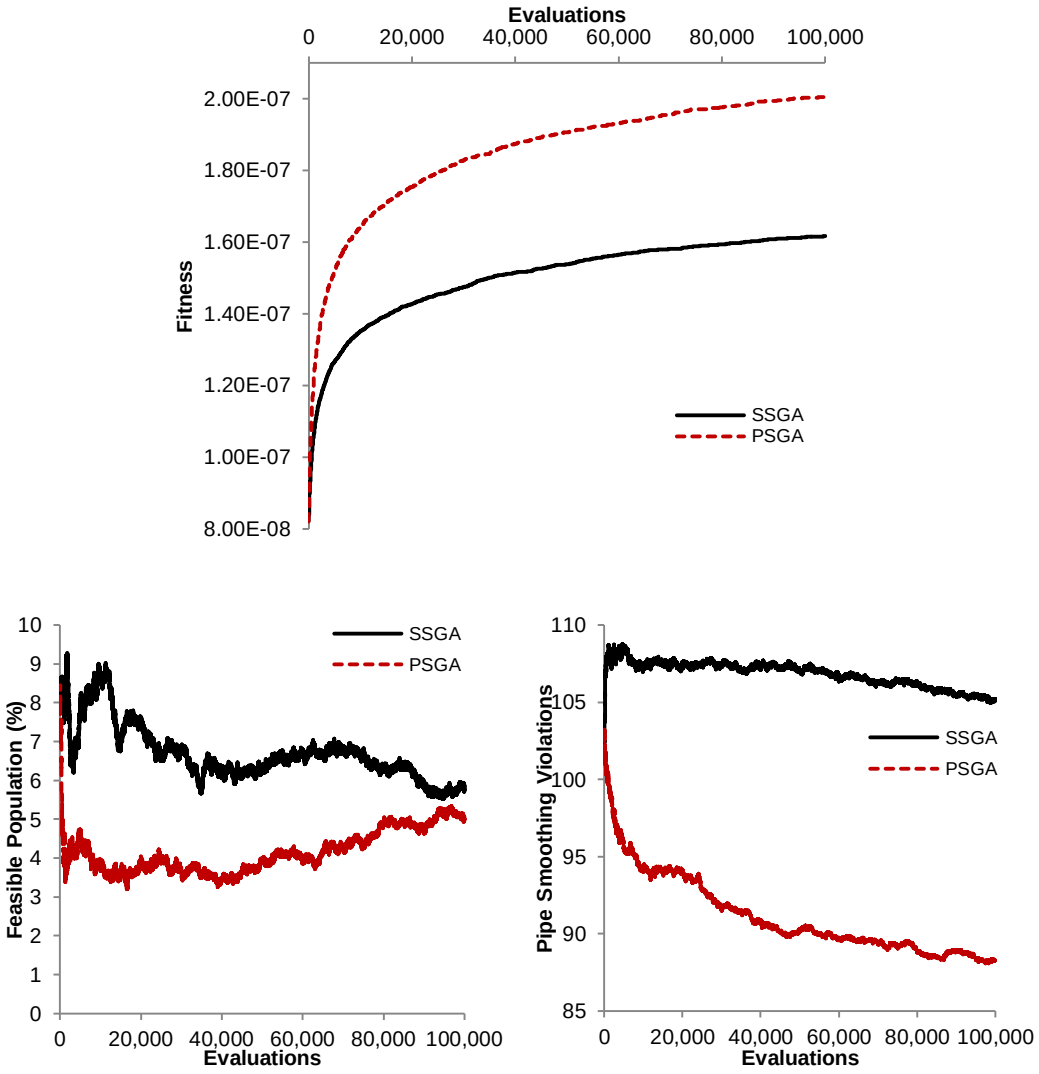


Figure 5-36. Graphs showing the average best solution fitness (top), average percentage of feasible solutions present in the population (left) & average pipe smoothing violations for the Modena problem – SSGA & PSGA comparison

Figure 5-36 presents the mean best solution fitness for the 50 runs over the 100,000 evaluations for both algorithms (top), the average percentage of feasible solutions present in the population (bottom left) and the average pipe smoothing violations (bottom right) for the Modena problem. From this figure it can be seen that PSGA outperforms the SSGA throughout the entire search of the algorithms. It can also be seen that the mean population feasibility for PSGA drops immediately from 8% to 4% at the beginning of the search and then proceeds to climb back to just under the feasibility levels of the SSGA at around 5-6%.

The overall population feasibility levels generated by both algorithms are quite low for this problem suggesting that in this case the search of a GA benefits from having solutions close to the feasible solution space boundary. This is a trait that is encouraged by the pipe smoothing heuristic in PSGA and hence why there are performance gains to be had in the early stages of the search. The average pipe smoothing violations produced by PSGA during the search is less than that produced by the SSGA.

5.2.1.2.5 Network A

The following set of results is for the Network A water distribution network design problem. This problem presents a real-world challenge for optimization algorithms due to the network complexity and hydraulic constraints which are difficult to satisfy whilst maintaining competitive solution quality. Table 5-21 presents the best single run results for the Network A problem for the SSGA and PSGA after 100,000 solution evaluations. It can be seen that the SSGA finds a better feasible cost solution than PSGA from the 10 runs after 100,000 fitness evaluations, however PSGA finds a smoother network solution compared with the SSGA.

Table 5-21. Best single run results for the Network A problem - SSGA & PSGA comparison

	Best Single Run			
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**	**Pipe Smoothing Violations**
SSGA	**2.723E-07**	**3,672,300**	**1934**	129
PSGA	2.480E-07	4,033,160	2048	**74**

The following table shows the average results from all the runs for this problem. As can be seen SSGA achieves a better average feasible cost and fitness value than the PSGA. Interestingly PSGA outperforms the SSGA in terms of evaluations taken to achieve a feasible solution, finding the feasible solution space on average in around 2000 fewer evaluations than the SSGA. The PSGA also achieves smoother networks than the SSGA on average.

Table 5-22. Mean results for the Network A problem – SSGA & PSGA comparison

	Average							
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**	**Pipe Smoothing Violations**	**Pipe Smoothing Violations Standard Deviation**
SSGA	**2.530E-07**	1.519E-08	**4,024,560**	321,490	6333.2	4814.9	139.90	**7.45**
PSGA	2.398E-07	**7.039E-09**	4,236,430	**261,929**	**4406.0**	**2595.1**	**87.20**	8.52

The following figure shows the mean best solution performances of both algorithms are very similar in terms of solution fitness throughout the entire search. Following further statistical testing (Mann-Whitney U test) it was found that there was no significant difference between each algorithm's set of final solutions in terms of fitness and feasible solution cost. The only clear difference in performance between the algorithms is the ability for PSGA to promote network smoothness over the SSGA for this complex problem.

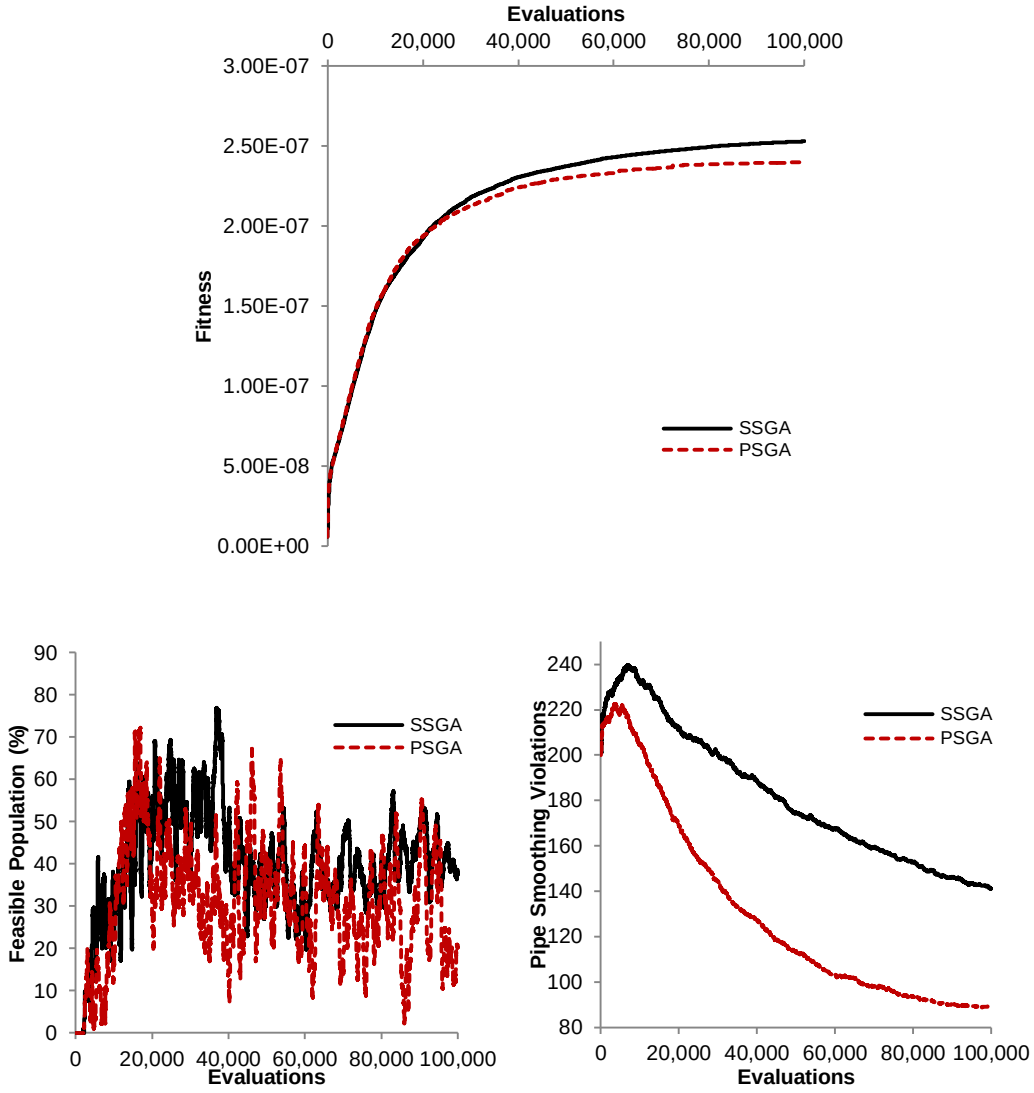


Figure 5-37. Graphs showing the average best solution fitness (top), average percentage of feasible solutions present in the population (left) & average pipe smoothing violations for the Network A problem – SSGA & PSGA comparison

5.2.1.2.6 Network B

The following section presents the results for the Network B problem, a real-world water distribution network and the largest benchmark problem presented in this work. The following table shows the best solutions found by both PSGA and the SSGA after 100,000 solution evaluations from ten randomly seeded separate runs. In terms of solution fitness and feasible network cost, PSGA finds a better solution however as with the previous real world large scale network (Network A) PSGA finds a much smoother network solution than the SSGA.

Table 5-23. Best single run results for the Network B problem - SSGA & PSGA comparison

	Best Single Run			
Algorithm	**Fitness**	**Feasible Cost ($)**	**Evaluations to Feasible**	**Pipe Smoothing Violations**
SSGA	1.125E-07	8,887,670	**10238**	346
PSGA	**1.184E-07**	**8,446,460**	12698	**183**

Table 5-24 presents the mean results for the two competing algorithms for this problem. The results show that the PSGA outperforms the SSGA in terms of solution fitness and feasible network cost after 100,000 fitness evaluations. However as with the Network A experiment set, these solution sets also do not provide statistically significant fitness or feasible network cost results. However there is a statistically significant difference between the two sets of results when comparing the pipe smoothing violations. With PSGA on average reaching a much smoother network compared to the SSGA.

Table 5-24. Mean results for the Network B problem – SSGA & PSGA comparison

	Average							
Algorithm	**Fitness**	**Fitness Standard Deviation**	**Feasible Cost ($)**	**Feasible Cost Standard Deviation ($)**	**Evaluations to Feasible**	**Evaluations to Feasible Standard Deviation**	**Pipe Smoothing Violations**	**Pipe Smoothing Violations Standard Deviation**
SSGA	1.049E-07	**4.863E-09**	9,827,230	**768,935**	**17027.4**	**6555.6**	359.20	**9.95**
PSGA	**1.101E-07**	5.873E-09	**9,398,420**	871,918	24832.2	10989.3	**208.70**	14.19

Figure 5-38 presents the average fitness and population feasibility results for the two algorithms on the Network B problem. The first observation is that both algorithms seem to perform equally in the first stages of the search, however the PSGA goes on to improve upon the SSGA in terms of fitness after approximately 15,000 solution evaluations. As with the previous problem (Network A), PSGA exhibits a strong tendency to produce much smoother solutions at all stages of the search. This behaviour is shown in the figure below with the mean pipe smoothing violations decreasing at a higher rate than the SSGA.

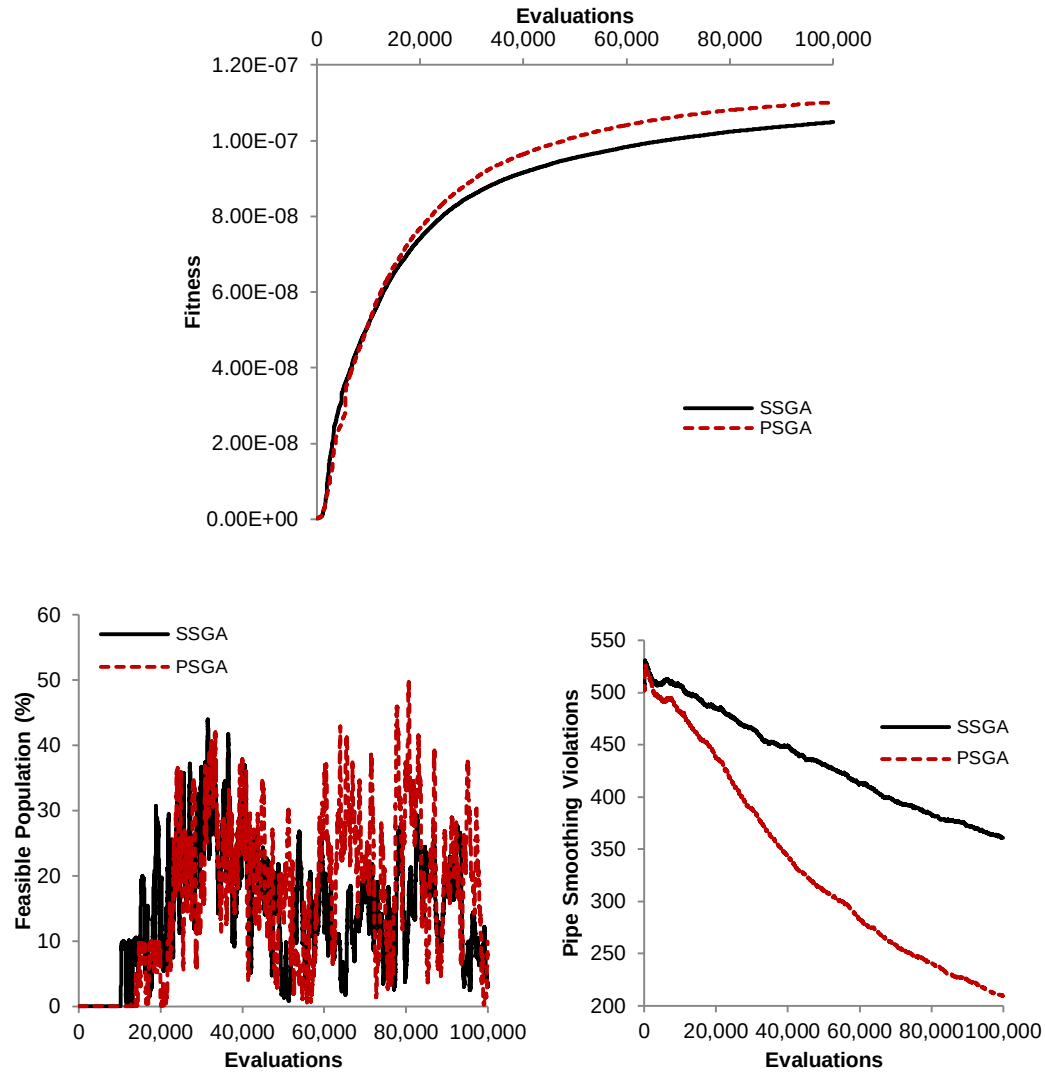


Figure 5-38. Graphs showing the average best solution fitness (top), average percentage of feasible solutions present in the population (left) & average pipe smoothing violations for the Network B problem – SSGA & PSGA comparison

5.2.1.3 Conclusion

The primary aim of this section was to compare the performance of the Pipe Smoothing Genetic Algorithm (PSGA) with the Steady State Genetic Algorithm (SSGA). The problems used in this set of experiments were selected for their range of size, complexity and layout, hence offering a diverse set of challenges for both algorithms with the view of providing an insight into the performance and behaviour of PSGA. Both algorithms utilized the same parameters obtained from the parameter tuning experiments presented earlier in this chapter (section 0). This was done to ensure that the SSGA was not

placed at a disadvantage when being compared to PSGA and both algorithms could be fairly compared.

PSGA generally displayed increased performed over the SSGA on the smaller single source network design problems (Two Loop, Hanoi, Foss Poly 1). It was found that PSGA displayed heightened performance in the early stages of the search compared with the SSGA and commonly went on to produce competitive final solutions, often better than the highly tuned SSGA. The results from the Modena problem suggest that PSGA would work extremely well for large, real world problems with multiple sources.

Integrating the pipe smoothing heuristic into the mutation operator of a standard steady state genetic algorithm has shown to improve algorithm performance in the majority of situations often improving overall solution quality significantly. Producing good quality solutions with less pipe smoothing constraints would allow an engineer to utilize such a method to aid in the initial and ongoing design of a commercial water distribution system.

5.2.2 Parameter Robustness

Identifying the optimal GA parameter set can involve a large number of algorithm runs using a number of different parameter configurations. This can be a time consuming operation and in the case of complex large real-world problems is near infeasible. The sensitivity of an algorithm to its parameter configuration (parameter robustness) is therefore an important consideration, and in this section, the performance of both the SSGA and PSGA are compared over a number of different parameter configurations involving the tournament size and mutation rate.

5.2.2.1 Experimental Setup

As with the parameter tuning experiments conducted earlier in this chapter, for each individual network problem eight evenly distributed parameter values were selected for each of the test parameters (tournament size & mutation probability). The parameter values used in this set of experiments are the same as those used in the initial parameter tuning runs presented in section 0, however in this experiment set the penalty factor was used as a variable and

was fixed to the best performing value found during the initial parameter tuning experiments for each problem. Using an exhaustive approach, each of the possible parameter value combinations were tested on the selected network problem. For each parameter value combination SSGA and PSGA were run 50 times, each for 100,000 objective function evaluations utilizing a different randomly seeded starting population for each run, however for fairness the two algorithms utilized the same random number seed list, as has been the case in previous comparison experiments. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size was fixed at 100 individuals, single-point crossover was utilized and a conditional worst individual replacement strategy was implemented.

5.2.2.2 Results

This section presents the results from the parameter robustness experiments for the SSGA and ALCO-GA. To enable the comparison between the two algorithms the mean results for each parameter value were plotted for each algorithm. The area under/above (maximization/minimization) the resultant curve is highlighted to aid in the comparison of the two algorithms' performance over the selected range of parameter values. The resultant areas are then normalized using the area produced by the two most extreme data points produced by the experiments, thus resulting in a proportional representation for each algorithm where the greater the proportion the greater the parameter robustness of the algorithm and the less susceptible it is to parameter variance.

5.2.2.2.1 Two Loop

The following figures present the normalized area under the curve for the tournament size and mutation probability on the Two Loop problem. The results for the tournament size show that PSGA is far less sensitive to changes in the tournament size than the SSGA in the case of solution quality and number of best known solutions found. The difference in sensitivity to mutation rate changes between the two algorithms is diminished compared with tournament size changes; however PSGA still outperforms the SSGA in terms of solution quality and the number of best known solutions found. It should be noted that the initial population (identical for both PSGA & SSGA) contained feasible

solutions therefore both algorithms achieved feasibility in 0 fitness evaluations. This is also the case for the pipe smoothing violations where all of the best solutions produced networks with no pipe smoothing violations; hence both algorithms produced identical smoothness results.

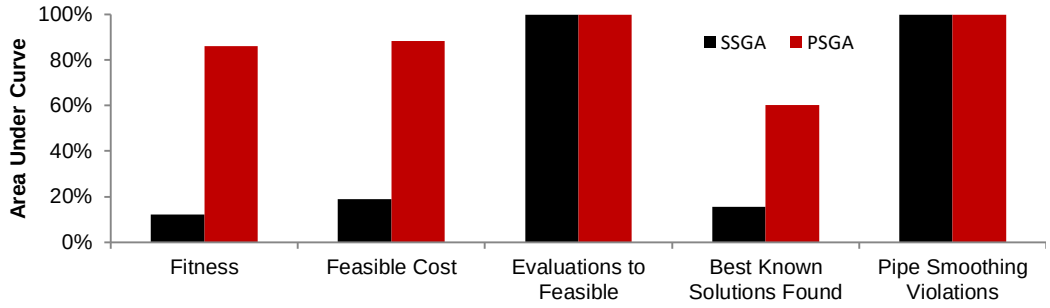


Figure 5-39. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Two Loop problem - SSGA & PSGA

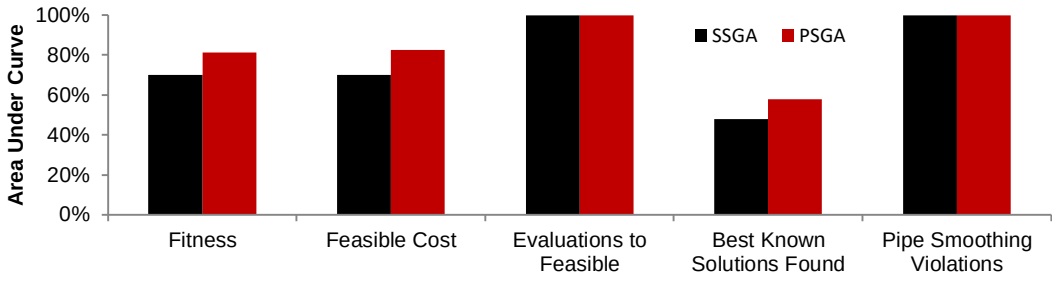


Figure 5-40. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Two Loop problem - SSGA & PSGA

5.2.2.2.2 Foss Poly 1

The robustness results for the Foss Poly network design problem are displayed below. As with the previous experiments the sensitivity to variations in tournament size of PSGA are less than that of the SSGA. The difference in sensitivity between the two algorithms for the mutation rate results is similar to that of the tournament size experiment however as was found with previous problems the difference in sensitivity to mutation rate is diminished compared with tournament size.

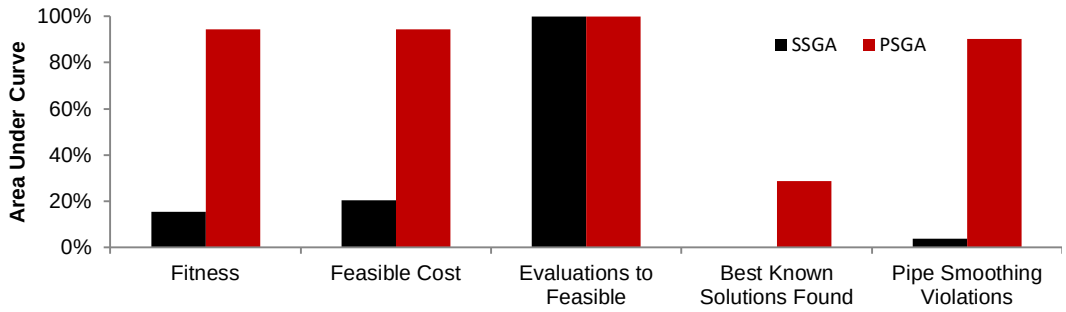


Figure 5-41. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Foss Poly 1 problem - SSGA & PSGA

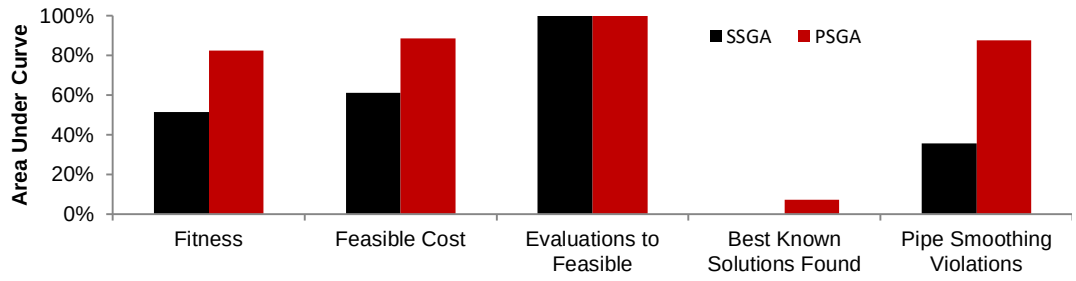


Figure 5-42. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Foss Poly 1 problem - SSGA & PSGA

5.2.2.2.3 Hanoi

The results for the tournament size show that PSGA is far less sensitive to changes in the tournament size than the SSGA in the case of solution quality, number of best known solutions found and network smoothness. The difference in sensitivity to mutation rate changes between the two algorithms is diminished compared with tournament size changes; however PSGA still outperforms the SSGA in terms of solution quality, the number of best known solutions found and pipe smoothing violations. What is clear is that the PSGA on average increases the number of evaluations required to find the feasible search space over the SSGA for the Hanoi problem.

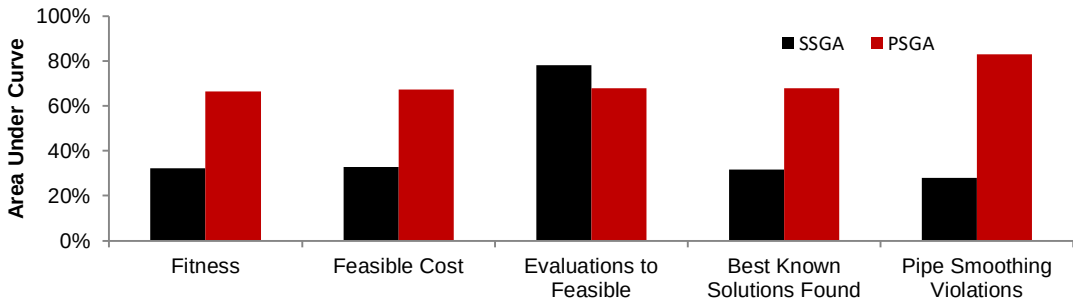


Figure 5-43. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Hanoi problem - SSGA & PSGA

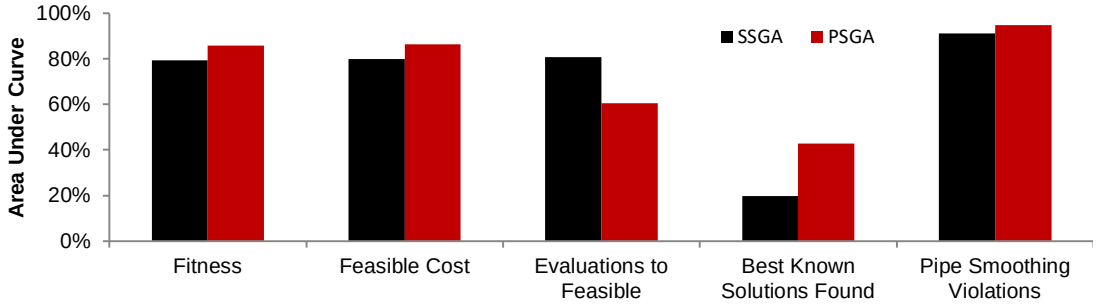


Figure 5-44. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Hanoi problem - SSGA & PSGA

5.2.2.2.4 New York Tunnels

The results for the New York Tunnels problem are presented in the following figures. PSGA is shown to be considerably less sensitive to tournament size changes than the SSGA when it comes to solution fitness, feasible network cost, the number of best known solutions obtained and network smoothness. The mutation rate results reflect the same behaviour although the difference between the algorithms is much less pronounced. The parallel expansion nature of the New York Tunnels problem lends itself to the production of feasible solutions and in this case feasible solutions are found in the randomly generated starting populations. This results in the evaluations taken to achieve a feasible solution to 0 for both methods and is therefore reflected in the 100% area displayed in the figures.

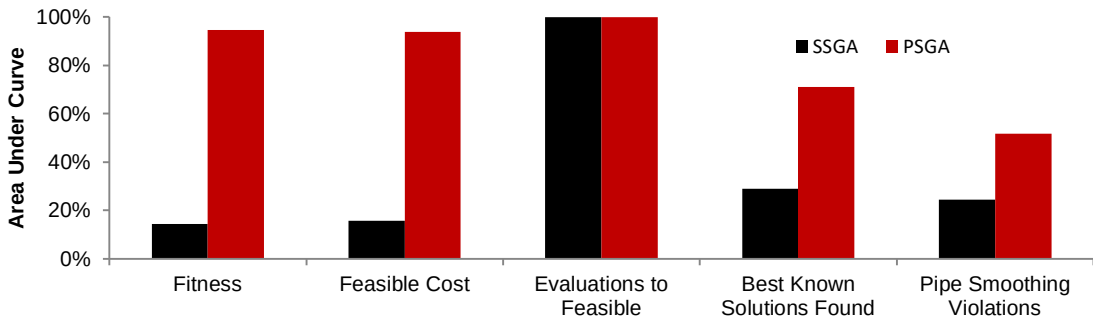


Figure 5-45. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the New York Tunnels problem - SSGA & PSGA

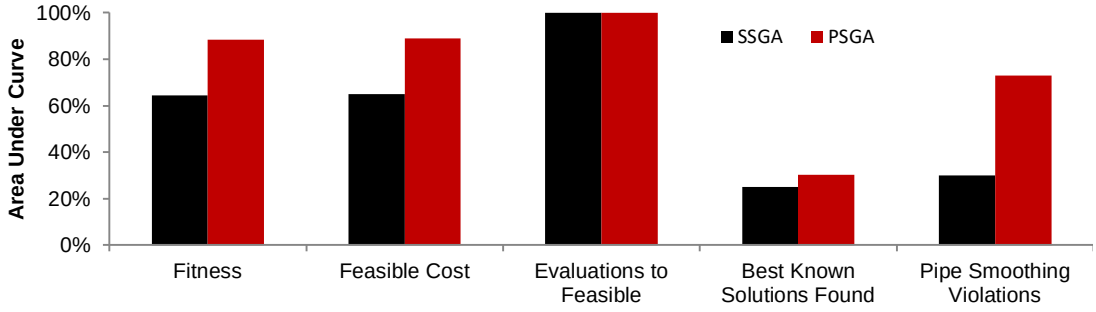


Figure 5-46. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the New York Tunnels problem - SSGA & PSGA

5.2.2.2.5 Modena

The following figures present the robustness results for the Modena network design problem. As can be seen, the SSGA tends to be a lot more sensitive to changes in the tournament size than PSGA. PSGA on average achieves a better result in terms of solution fitness, feasible network cost, the number of best known solutions found and pipe smoothing violations than the SSGA. However, the mutation rate sensitivity results show a diminished difference between the two algorithms.

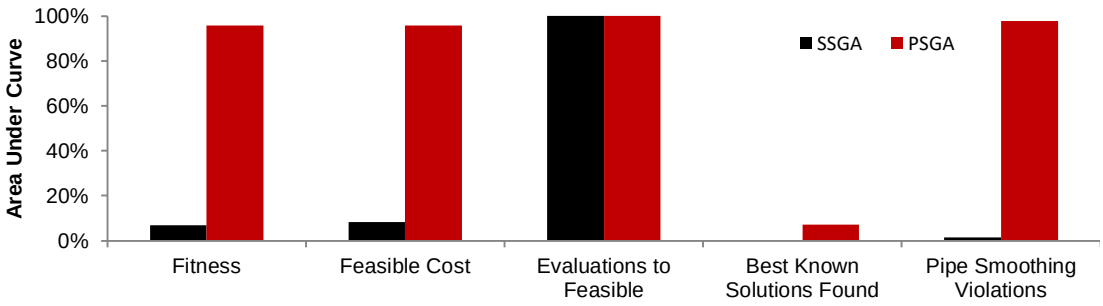


Figure 5-47. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying tournament sizes for the Modena problem - SSGA & PSGA

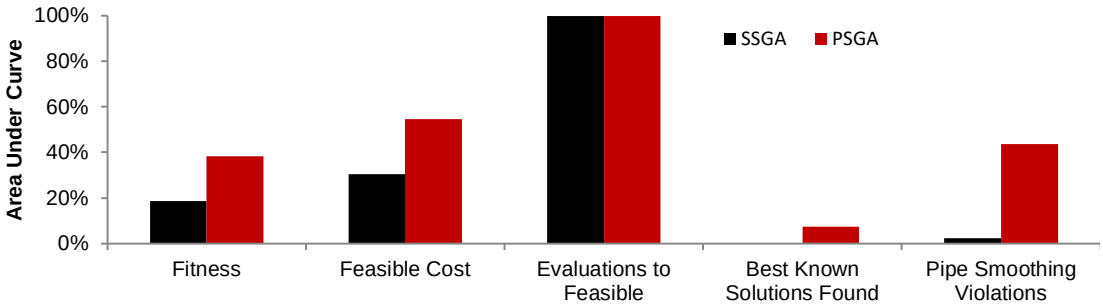


Figure 5-48. Normalized resultant area for fitness, feasible cost, evaluations to feasible and number of best known solutions found for varying mutation rates for the Modena problem - SSGA & PSGA

5.2.2.2.6 Network A

Due to the complexity of the Network A water distribution network design problem and the resultant run time this set of experiments were scaled down compared with the previously presented problems. For this experiment the tournament size ranged between 2% and 8% in 2% intervals and mutation probability ranged between 0.0016 and 0.0128 in 0.0016 intervals. For each parameter value combination SSGA and PSGA were run 10 times, each for 100,000 objective function evaluations utilizing a different randomly seeded starting population for each run. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size was fixed at 100 individuals, single-point crossover was utilized and a conditional worst individual replacement strategy was implemented.

Figure 5-49 shows the average fitness, feasible network cost, evaluations to feasible and pipe smoothing results at each of the 4 selected tournament sizes for the Network A problem. Looking at the fitness and feasible network cost plots, PSGA is seen to achieve better results than the SSGA at all tournament sizes. The results suggest that the SSGA achieves best performance at a tournament size of 0.06N along with the PSGA. Both algorithms are seen to be affected by varying tournament size; evaluations to a feasible solution decreases as the tournament size increases however it is clear that the performance of the SSGA is influenced to a much greater degree than PSGA. This is reflected in Figure 5-50, which displays the normalized resultant area for the fitness, network cost, evaluation taken to reach a feasible solution and pipe smoothing violations.

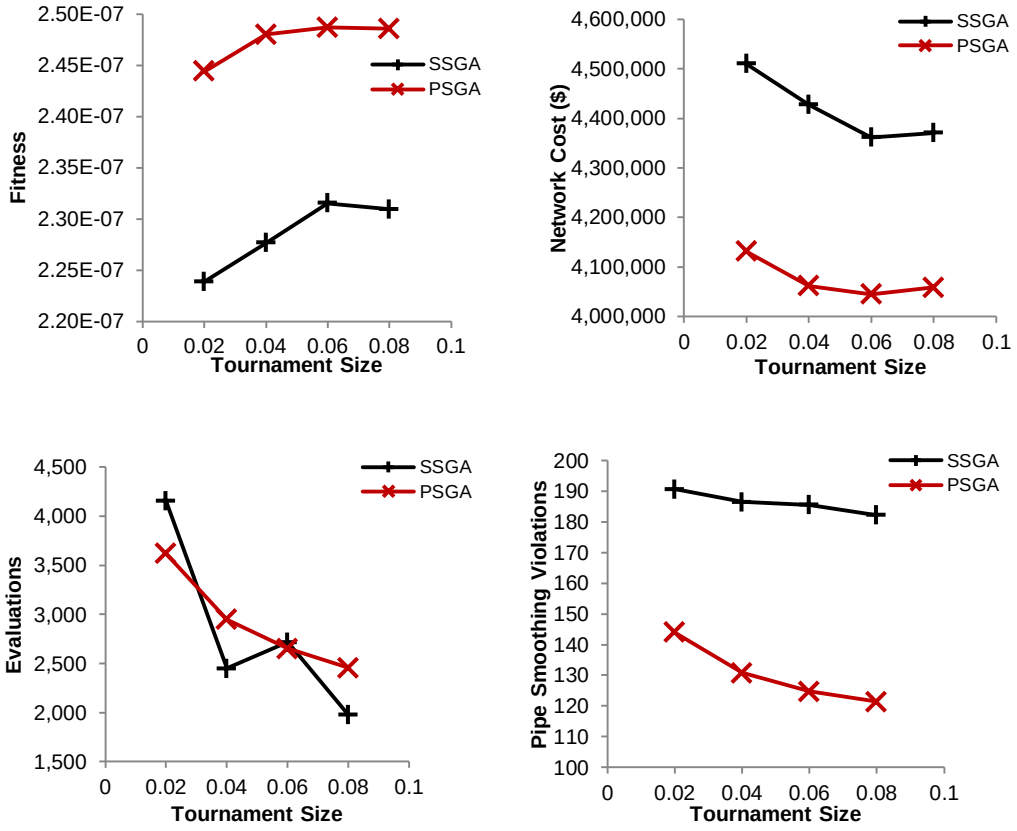


Figure 5-49. Average fitness (top left), feasible network cost (top right), evaluations taken to achieve a feasible solution (bottom left) & pipe smoothing violations at varying tournament sizes for the Network A problem – SSGA & PSGA

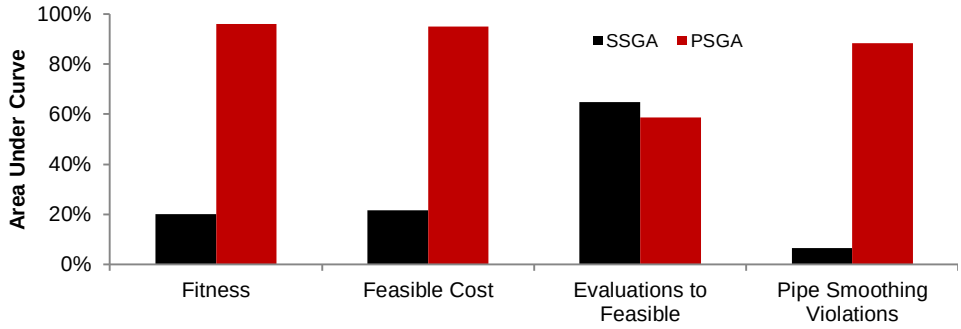


Figure 5-50. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying tournament sizes for the Network A problem - SSGA & PSGA

Figure 5-51 shows the average fitness, feasible network cost, evaluations to feasible and pipe smoothing violation results at each of the 8 selected mutation rates for the Network A problem. As can be seen, PSGA achieves an average solution quality higher than that of the SSGA at all mutation rates tested apart from the lowest mutation probability of 0.0016. Both algorithms display a general decrease in performance as mutation rate is increased however the inclusion of the pipe smoothing heuristic based mutation seems to diminish this decline in performance resulting in a decreased sensitivity to mutation rate compared to the SSGA. There is a significant difference in performance between the two algorithms in terms of pipe smoothing violations, pronounced slightly more at lower mutation rates.

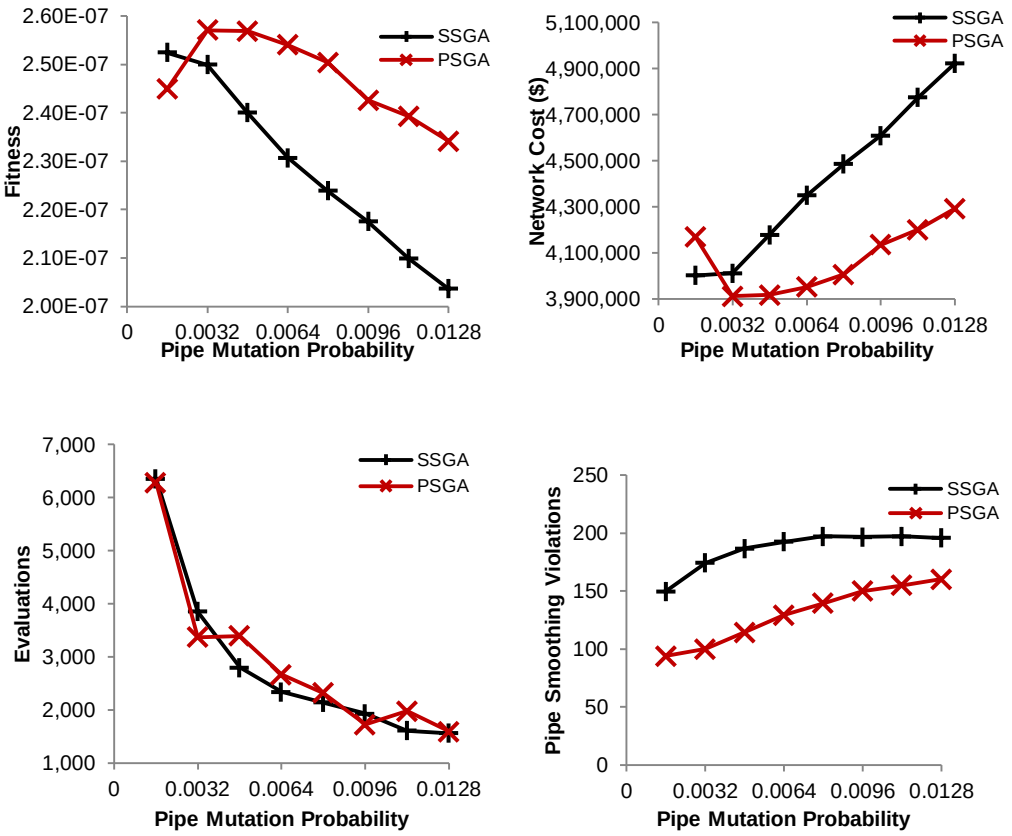


Figure 5-51. Average fitness (top left), feasible network cost (top right), evaluations taken to achieve a feasible solution (bottom left) & pipe smoothing violations (bottom right) at varying pipe mutation rates for the Network A problem – SSGA & PSGA

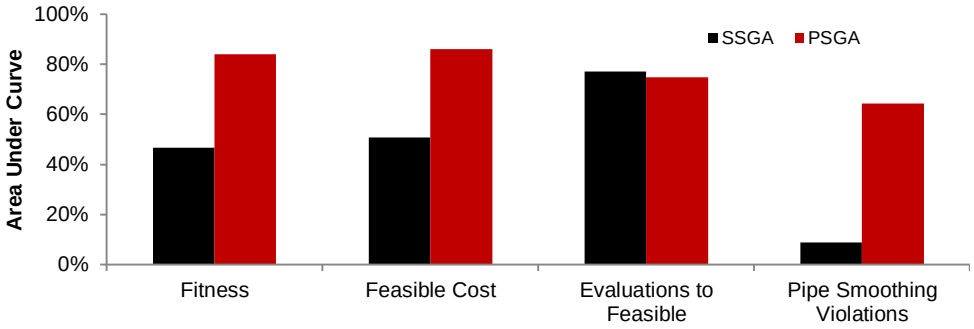


Figure 5-52. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying mutation rates for the Network A problem - SSGA & PSGA

5.2.2.2.7 Network B

As with the previous problem (Network A), due to the complexity of the Network B problem and the resultant run time this set of experiments were scaled down compared with the previously presented problems. For this experiment the tournament size ranged between 2% and 8% in 2% intervals and mutation probability ranged between 0.0008 and 0.0064 in 0.0008 intervals. For each parameter value combination SSGA and PSGA were run 10 times, each for 100,000 objective function evaluations utilizing a different randomly seeded starting population for each run. The remaining parameter values and algorithm setup remained constant through all runs for all experiments: Population size was fixed at 100 individuals, single-point crossover was utilized and a conditional worst individual replacement strategy was implemented.

Figure 5-53 shows the average fitness, feasible network cost, evaluations to feasible and pipe smoothing results at each of the 4 selected tournament sizes for the Network B problem. Looking at the fitness and feasible network cost plots, PSGA is seen to achieve better results than the SSGA at all tournament sizes. The results suggest that the SSGA achieves best performance at a tournament size of 0.06N whilst PSGA achieves optimal performance at a tournament size of 0.08N. Both algorithms are seen to be affected by varying tournament size; evaluations to a feasible solution decreases as the tournament size increases however PSGA seems to perform best at 0.06N and the evaluations to feasibility increases again with tournament size. In terms of network smoothness, the PSGA achieves much smoother solutions than the SSGA, it also seems that as tournament size increases so does network smoothness. These results are reflected in Figure 5-54, which displays the normalized resultant area for the fitness, network cost, evaluation taken to reach a feasible solution and pipe smoothing violations.

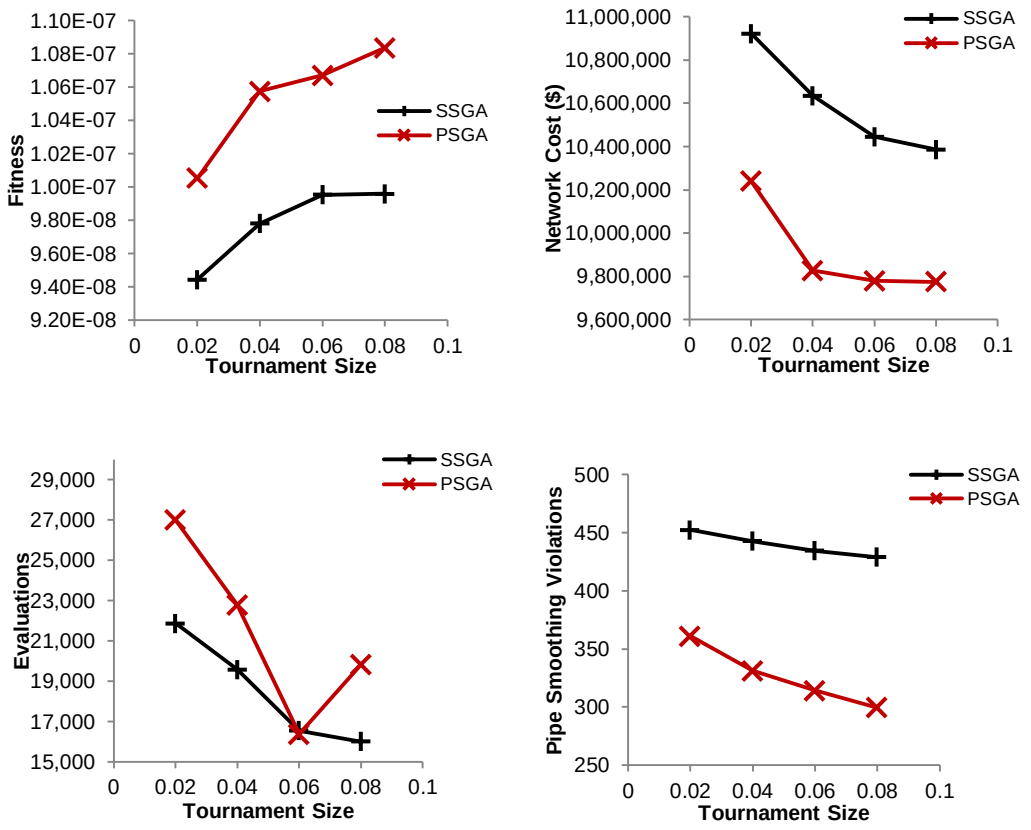


Figure 5-53. Average fitness (top left), feasible network cost (top right), evaluations taken to achieve a feasible solution (bottom left) & pipe smoothing violations at varying tournament sizes for the Network B problem – SSGA & PSGA

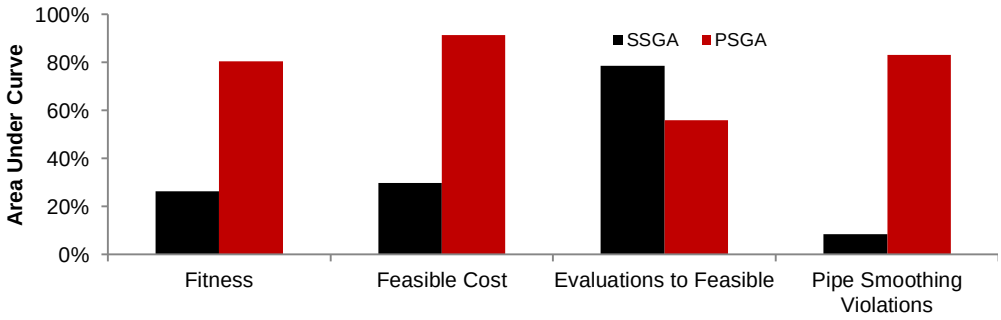


Figure 5-54. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying tournament sizes for the Network B problem - SSGA & PSGA

Figure 5-55 shows the average fitness, feasible network cost, evaluations to feasible and pipe smoothing violation results at each of the 8 selected mutation rates for the Network B problem. As can be seen, PSGA achieves an average solution quality higher than that of the SSGA at all mutation rates tested apart from the lowest mutation probability of 0.0008. Both algorithms display a general decrease in performance as mutation rate is increased however the inclusion of the pipe smoothing heuristic based mutation seems to diminish this decline in performance resulting in a decreased sensitivity to mutation rate compared to the SSGA. There is a significant difference in performance between the two algorithms in terms of pipe smoothing violations, pronounced slightly more at lower mutation rates. It is also observed that as the mutation rate is increased so does the number of pipe smoothing violations. This is interesting as one would expect the number of violations to decrease with more applications of the pipe smoothing heuristic.

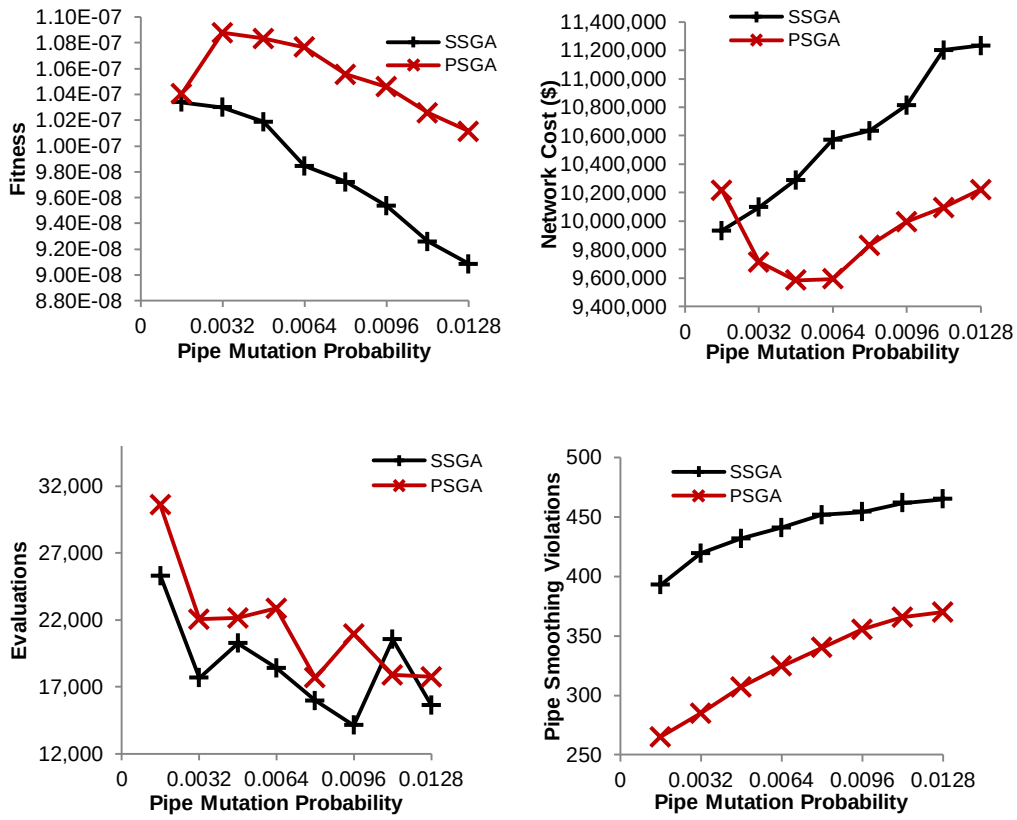


Figure 5-55. Average fitness (top left), feasible network cost (top right), evaluations taken to achieve a feasible solution (bottom left) & pipe smoothing violations (bottom right) at varying pipe mutation rates for the Network B problem – SSGA & PSGA

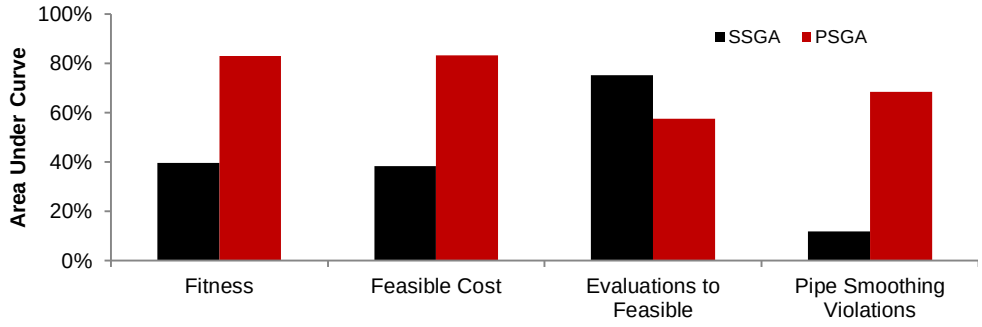


Figure 5-56. Normalized resultant area for fitness, feasible cost and evaluations to feasible for varying mutation rates for the Network B problem - SSGA & PSGA

5.2.3 Conclusion

A Pipe Smoothing Genetic Algorithm (PSGA) for the least-cost design of water distribution network optimisation problems has been described. The algorithm is based on a standard Steady State Genetic Algorithm (SSGA) with the addition of a pipe smoothing heuristic based mutation operator which utilises pipe flow direction, and pipe diameter information to guide the search of the algorithm to an engineering and mathematically optimal solution. The algorithm has been found to perform very well on a range of WDNs of varying complexity compared to the SSGA; sometimes not only finding a better solution but achieving it in less solution evaluations. The algorithm is able to generate smoother networks solutions in fewer evaluations than the SSGA and so in applications where the number of function evaluations is limited, i.e. where large real-world networks are optimised, PSGA can provide engineering feasible designs much earlier in the evolutionary process. Additional experimentation has shown that the PSGA is more robust to parameter settings than the SSGA meaning that if extensive parameter tuning is not feasible due to the complexity of the network, the rule-of-thumb parameters are more likely to function well with PSGA than the SSGA. In short, PSGA would be a good choice for an engineer wanting to optimise a large complex real-world network for the first time.

5.3 Heuristic Based Algorithm Comparison and Sensitivity Analysis

In this section the Steady State GA and both of the engineering heuristic based algorithms (ALCO-GA & PSGA) are compared to one another, not only with regards to solution fitness but also in terms of each algorithm's sensitivity to parameter variation. Utilizing a form of regression analysis and normalization techniques this section attempts to quantify the sensitivity of each algorithm to parameter variation, such that they can be accurately compared. As stated previously, an algorithm's sensitivity to parameter change can be an important consideration when selecting an optimization technique. If the algorithm is too parameter sensitive then the user is less likely to achieve high quality solutions without extensive experimentation and parameter tuning which is often undesirable in commercial settings due to time and budget constraints. In situations such as this, the wider the effective operating window, that is the wider the range of parameter values likely to produce a quality solution, the better. In an ideal world, an algorithm would have zero sensitivity to parameter change, resulting in no degradation in performance as parameter values are varied, however, in reality most algorithms have a 'sweet spot' where the algorithm performs the best. It has been observed from the extensive parameter tuning conducted earlier in this chapter that both the engineering inspired genetic algorithms seem to be less sensitive to parameter value changes resulting in a wider effective operating window.

5.3.1 Quantifying Parameter Sensitivity and Performance

A method to accurately quantify and compare the sensitivity of the two engineering inspired algorithms (ALCO-GA & PSGA) to changes in parameter value variation is described in this section. It is first necessary to define two points of reference, a best and worst case in terms of an algorithm's sensitivity and performance at a given parameter. The best case would be an "ideal" (hypothetical) genetic algorithm which would have zero sensitivity to parameter value changes and therefore would achieve the same best solution at all parameter values. It was found from the extensive experimentation that both of the engineering inspired algorithms outperformed SSGA in terms of fitness for both tournament size and mutation rate in all test problems, therefore it was

decided to use the SSGA's performance as a baseline for comparison. A similar technique to that employed in the parameter robustness section (5.1.2) was used to determine an algorithm's performance over a range of parameter values. In this case polynomial regression was employed to model the expected fitness value in terms of each parameter (tournament size & mutation rate) from the extensive experimentation results presented in sections 5.1.2 and 5.2.2. It was found that a 3rd degree or cubic polynomial was able to produce an acceptable goodness of fit for all algorithm results on all problems. The resultant polynomial could then be used to determine the area under the curve:

$$area = \int_{x_{min}}^{x_{max}} \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \, dx \qquad (25)$$

Where x is the parameter value, x_{min} and x_{max} are the minimum and maximum parameter values tested and β_0 to β_3 are the regression coefficients. The area is then normalised between the area under the curve generated for SSGA and the area generated by the theoretical "ideal" algorithm resulting in a sensitivity value (S) between 0 and 1:

$$area_{ideal} = y_{max}(x_{max} - x_{min}) \qquad (26)$$

$$S = \frac{area_{ideal} - area}{area_{ideal} - area_{ssga}} \qquad (27)$$

Where $area_{ideal}$ is the area generated by the theoretical "ideal" genetic algorithm, y_{max} is the best mean fitness value achieved by any of the algorithms on test and $area_{ssga}$ is the resultant area under the polynomial curve generated by the regression analysis.

5.3.2 Results and Analysis

5.3.2.1 Hanoi

Figure 5-57 shows the average fitness at varying tournament sizes in addition to the curves generated by the polynomial regression modeling for the Hanoi problem. It can be seen that PSGA clearly out performs both ALCO-GA and SSGA at all tournament sizes. For smaller tournament sizes both ALCO-GA and SSGA achieve similar average values however at tournament sizes larger than 0.05 ALCO-GA tends to attain better solutions nearing the quality of

those from PSGA. All algorithms exhibit a general decline in solution quality as tournament size is increased.

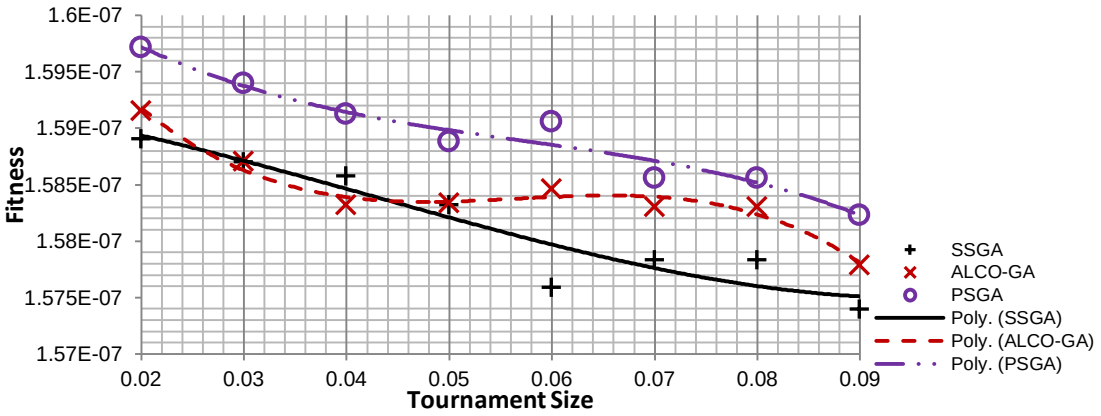


Figure 5-57 Average fitness and polynomial regression at varying tournament sizes for the Hanoi problem – SSGA, ALCO-GA & PSGA

Figure 5-58 shows the result from all three algorithms for the Hanoi problem at varying pipe mutation rates. It can be observed that although the SSGA marginally achieves the highest average fitness at a mutation rate of 0.147 that the solution quality falloff as the mutation rate continues to be increased is much more extenuated compared with the engineering inspired algorithms which exhibit less performance degradation at higher mutation rates, suggesting less overall sensitivity to mutation rate change than the SSGA.

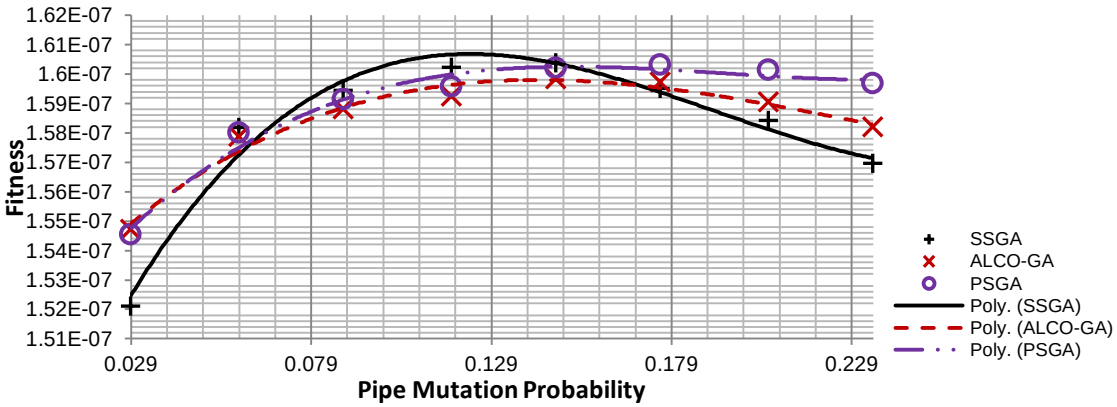


Figure 5-58 Average fitness and polynomial regression at varying pipe mutation rates for the Hanoi problem – SSGA, ALCO-GA & PSGA

The following table presents the sensitivity values of each algorithm for the Hanoi problem extensive run experiments. As stated before, the sensitivity

value is normalised between 1 and 0, where 1 is the SSGA sensitivity baseline for that parameter and 0 is based on the performance of an ideal algorithm (achieves the best known fitness values at every parameter value). From these results it is clear that PSGA is much less sensitive than both ALCO-GA and SSGA for tournament size and pipe mutation rate. Another observation is that both algorithms achieve a lower sensitivity value for tournament size than mutation rate which aligns with the observations made during the initial study into parameter variance (4.1.2.1) which suggested that mutation rate has a much greater impact on performance than tournament size.

Table 5-25 Algorithm tournament size and pipe mutation rate sensitivity for the Hanoi problem – ALCO-GA & PSGA

Algorithm Sensitivity - Hanoi		
Algorithm	***Tournament Size***	***Pipe Mutation Rate***
ALCO-GA	0.8244	0.9846
PSGA	0.4920	0.6903

5.3.2.2 Two Loop

The average fitness values at varying tournament sizes and the polynomial regression curves for the Two Loop problem are shown in Figure 5-59. Unlike the Hanoi problem both of the engineering inspired algorithms perform a lot better than SSGA at all tournament sizes with PSGA slightly out performing ALCO-GA. In general all algorithms display a minimal level of variation between tournament sizes with a slight peak present around 0.05-0.06.

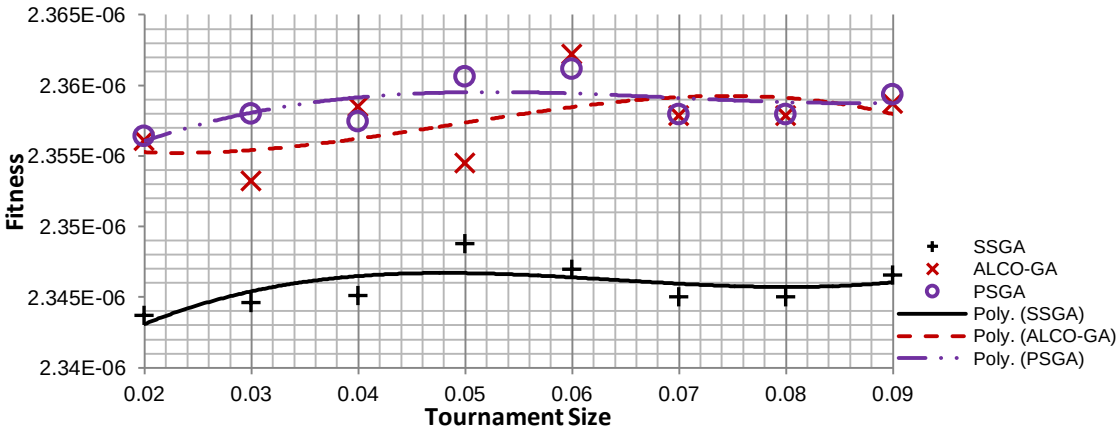


Figure 5-59 Average fitness and polynomial regression at varying tournament sizes for the Two Loop problem – SSGA, ALCO-GA & PSGA

Figure 5-60 displays the average fitness and polynomial regression curves at varying pipe mutation rates for the Two Loop problem. It can be observed that at low mutation rates all algorithms achieve very similar solutions with ALCO-GA and SSGA marginally outperforming PSGA. However at mutation rates higher than 0.5 the effectiveness of SSGA drops significantly compared to the engineering inspired algorithms which maintain relatively good performance at the higher mutation rates.

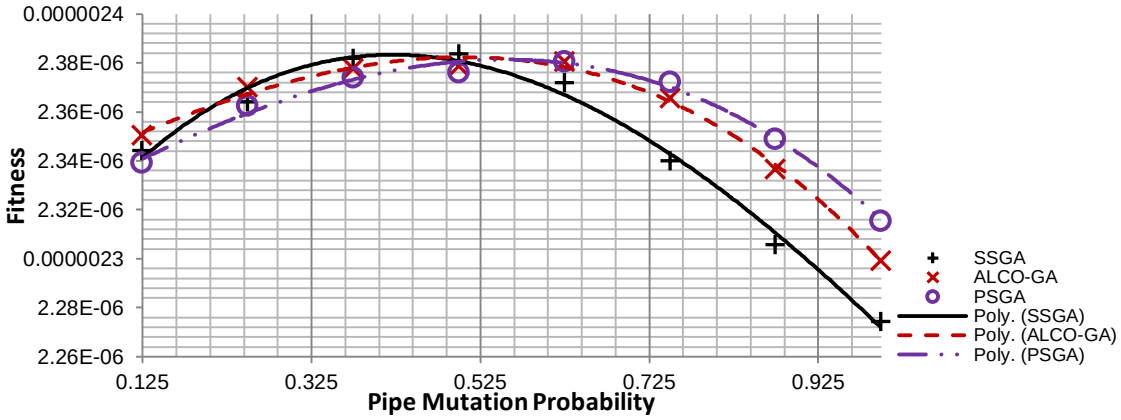


Figure 5-60 Average fitness and polynomial regression at varying pipe mutation rates for the Two Loop problem – SSGA, ALCO-GA & PSGA

Unlike the sensitivity values achieved for the Hanoi problem both ALCO-GA and PSGA attain very similar results. Once again, the algorithm's sensitivity to tournament size is much less than to the mutation rate.

Table 5-26 Algorithm tournament size and pipe mutation rate sensitivity for the Two Loop problem – ALCO-GA & PSGA

Algorithm Sensitivity - Two Loop

Algorithm	*Tournament Size*	*Pipe Mutation Rate*
ALCO-GA	0.2887	0.6666
PSGA	0.2079	0.6367

5.3.2.3 New York Tunnels

Figure 5-61 shows the performance results of the algorithms for the New York Tunnels problem. From these results it is clear that both of the engineering inspired algorithms achieve much better solutions than the SSGA with PSGA

outperforming ALCO-GA for all tournament sizes especially at lower values. Both ALCO-GA and SSGA tend to experience increased performance the higher the tournament size, however PSGA performance remains generally constant for all parameter values.

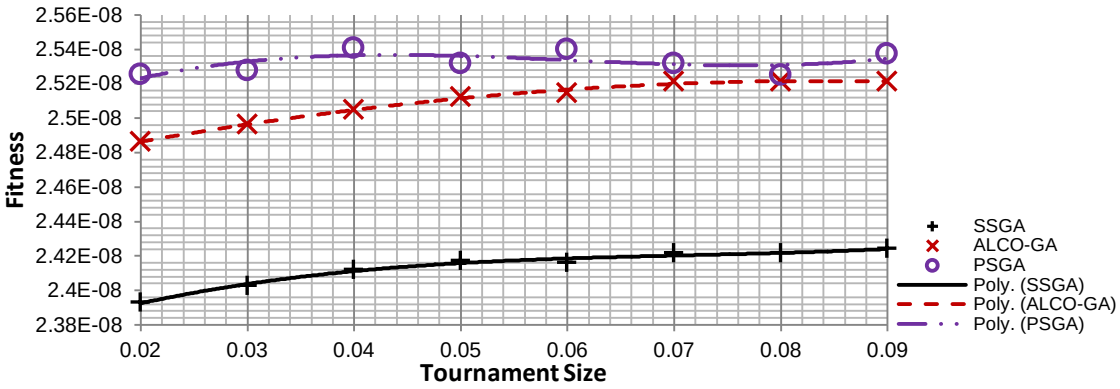


Figure 5-61 Average fitness and polynomial regression at varying tournament sizes for the New York Tunnels problem – SSGA, ALCO-GA & PSGA

Figure 5-62 shows the average fitness and polynomial regression results for varying mutation rates for the New York Tunnels problem. It is clear from these results that the SSGA is a lot more sensitive to changes in the mutation rate, displaying significant performance drop-off at higher rates of mutation compared to the engineering inspired algorithms. Both ALCO-GA and PSGA display a much flatter response over the range of mutation rates. Interestingly ALCO-GA achieves comparatively better performance than the other algorithms at low mutation rates; this suggests that the head deficit/excess eliminating heuristic is effective at lower application rates for relatively small parallel expansion problems such as this.

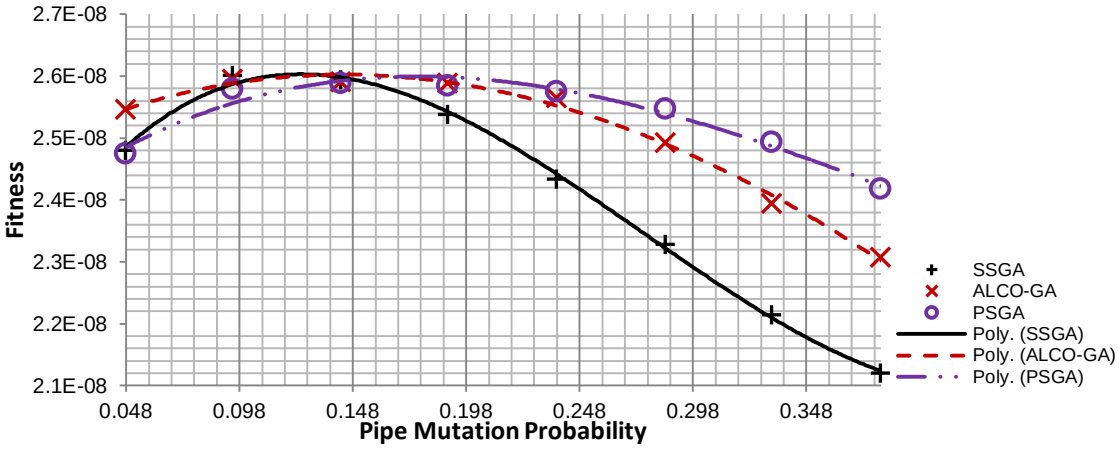


Figure 5-62 Average fitness and polynomial regression at varying pipe mutation rates for the New York Tunnels problem – SSGA, ALCO-GA & PSGA

PSGA displays a very low sensitivity to tournament size variation indicating it produces very high quality results for all tournament sizes tested. Interestingly both heuristic based algorithms are much less sensitive to mutation rate than the SSGA compared to their sensitivity for the Hanoi and Two Loop problems. This is almost certainly due to the parallel expansion nature of the New York Tunnels problem; both the pipe smoothing (PSGA) and hydraulic head excess reducing (ALCO-GA) effects have shown to work extremely well on problems of this nature.

Table 5-27 Algorithm tournament size and pipe mutation rate sensitivity for the New York Tunnels problem – ALCO-GA & PSGA

Algorithm Sensitivity - New York Tunnels

Algorithm	*Tournament Size*	*Pipe Mutation Rate*
ALCO-GA	0.2354	0.4532
PSGA	0.0593	0.3259

5.3.2.4 Foss Poly 1

The average fitness and polynomial regression analysis results at varying tournament sizes for the Foss Poly 1 problem are shown in Figure 5-63. It can be seen that both the engineering inspired algorithms produce significantly higher mean fitness solutions for all tournament sizes. A slight upward trend can be observed as tournament size is increased however both of the engineering

heuristic based algorithms display a much flatter response to the variation compared with SSGA.

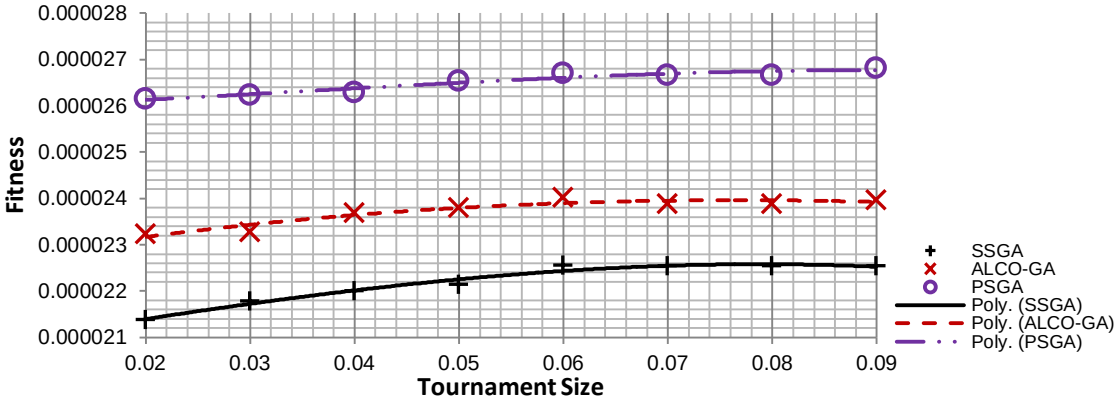


Figure 5-63 Average fitness and polynomial regression at varying tournament sizes for the Foss Poly 1 problem – SSGA, ALCO-GA & PSGA

From Figure 5-64 it can be seen that SSGA marginally achieves a better solution quality over the engineering inspired algorithms at low mutation rates, however, as the pipe mutation rate is increased both engineering algorithms consistently outperform SSGA at higher rates. Interestingly, the mutation rate at which the algorithm achieves peak performance is noticeably different, with SSGA's occurring at approximately 0.034 and both ALCO-GA and PSGA achieving peak performance at around 0.052 mutation probability.

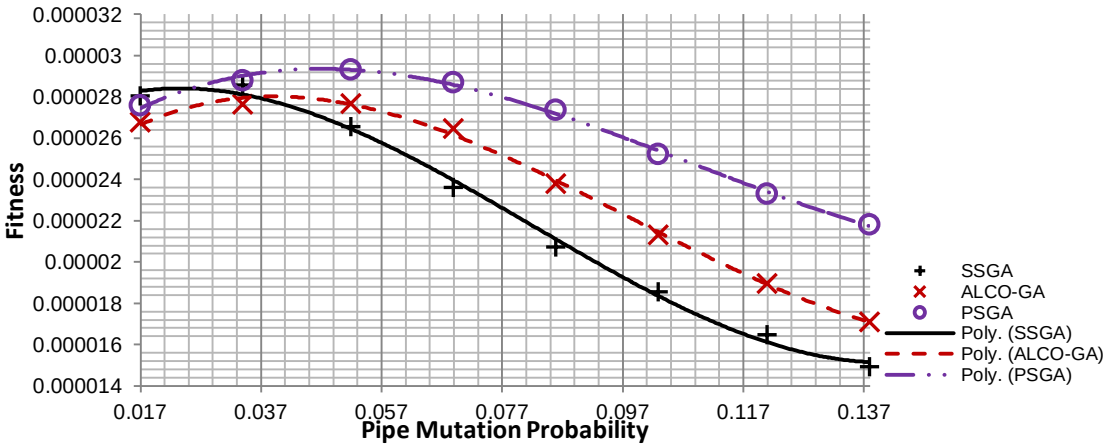


Figure 5-64 Average fitness and polynomial regression at varying pipe mutation rates for the Foss Poly 1 problem – SSGA, ALCO-GA & PSGA

The following table displays the algorithm parameter sensitivity values for the Foss Poly 1 problem. It can be seen that PSGA is significantly less sensitive to parameter value variation than ALCO-GA especially for tournament size changes where PSGA achieves a near flat response to the parameter.

Table 5-28 Algorithm tournament size and pipe mutation rate sensitivity for the Foss Poly 1 problem – ALCO-GA & PSGA

Algorithm Sensitivity - Foss Poly 1		
Algorithm	***Tournament Size***	***Pipe Mutation Rate***
ALCO-GA	0.6673	0.7498
PSGA	0.0647	0.3534

5.3.2.5 Modena

The results from the Modena problem for tournament size are shown in Figure 5-65. There is a significant difference in performance between the solution generated by PSGA and the other two algorithms on test. All algorithms exhibit a gentle increase in performance with an increase in tournament size peaking at a tournament size of approximately 0.08N.

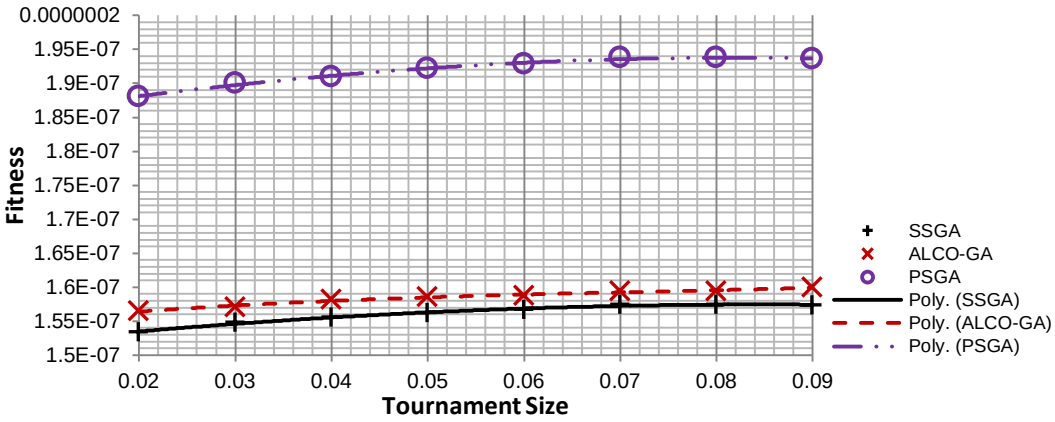


Figure 5-65 Average fitness and polynomial regression at varying tournament sizes for the Modena problem – SSGA, ALCO-GA & PSGA

Figure 5-66 displays the mean fitness results at varying mutation rates from the three algorithms for the Modena problem. As with the tournament size results, PSGA drastically outperforms both SSGA and ALCO-GA at all mutation rates tested. All algorithms are shown to perform better at a very low mutation

rates. Interestingly ALCO-GA does not significantly outperform SSGA on this problem and highlights one of the possible short comings of ALCO-GA; the hydraulic based heuristic becomes less effective in the case of multiple water sources as the heuristic may introduce a bias to one source through its upstream tracing procedure.

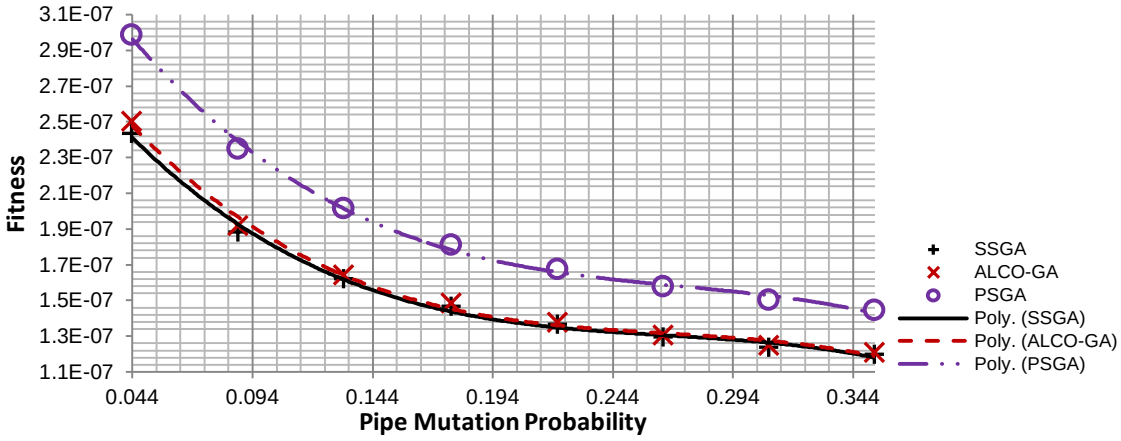


Figure 5-66 Average fitness and polynomial regression at varying pipe mutation rates for the Modena problem – SSGA, ALCO-GA & PSGA

The parameter sensitivity compared with the SSGA is shown in the following table. The sensitivity of ALCO-GA is very high compared with the PSGA especially in terms of the tournament size. This result stems from the greatly improved performance of the PSGA over all tournament sizes for the problem compared with both ALCO-GA and SSGA. As observed in previous problems, both of the engineering inspired algorithms are more sensitive to mutation rate variation than tournament size.

Table 5-29 Algorithm tournament size and pipe mutation rate sensitivity for the Modena problem – ALCO-GA & PSGA

Algorithm Sensitivity - Modena

Algorithm	*Tournament Size*	*Pipe Mutation Rate*
ALCO-GA	0.9385	0.9855
PSGA	0.0451	0.7601

5.3.2.6 Network A

The following figure shows the mean fitness and polynomial regression curves for the Network A problem at tournament sizes ranging from 0.02N to 0.08. From this figure PSGA can be observed outperforming ALCO-GA at all tournament sizes with the gap being closed slightly as tournament size is increased.

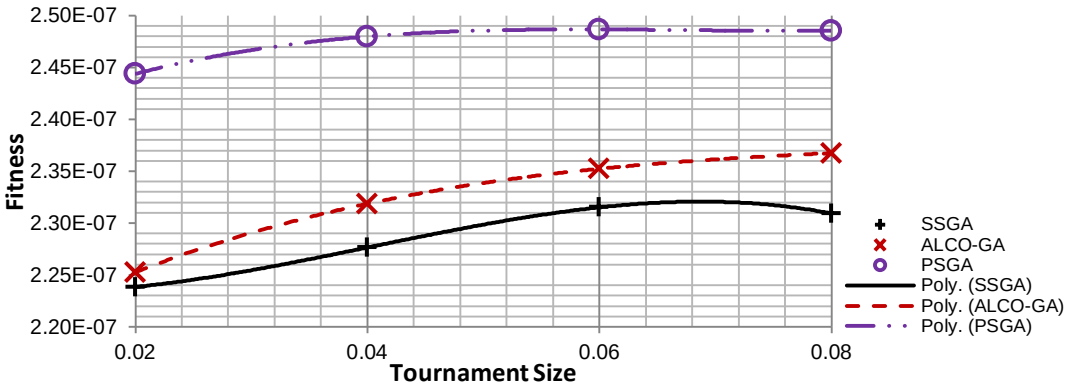


Figure 5-67 Average fitness and polynomial regression at varying tournament sizes for the Network A problem – SSGA, ALCO-GA & PSGA

Figure 5-68 shows the average fitness and polynomial regression results at a range of pipe mutation rates for the Network A problem. PSGA is outperformed by both SSGA and ALCO-GA at very low mutation rates; however both SSGA and ALCO-GA are significantly outperformed by PSGA at higher mutation rates.

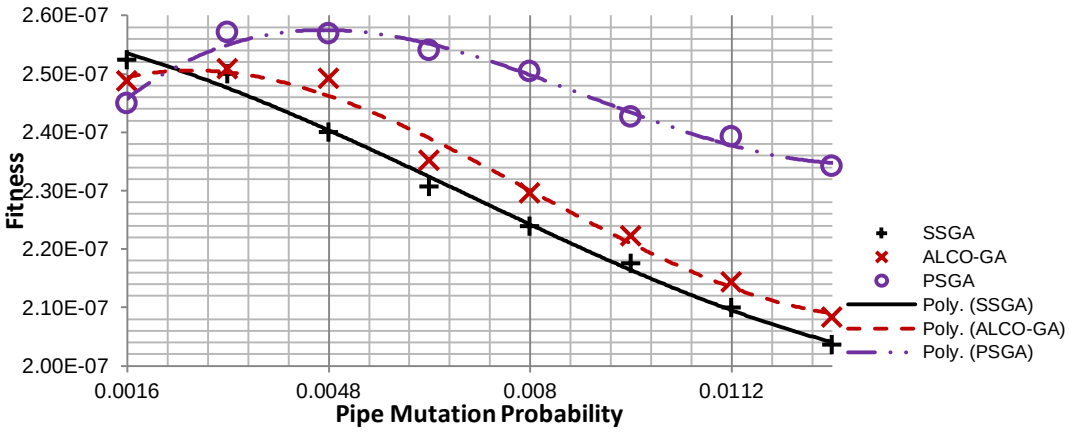


Figure 5-68 Average fitness and polynomial regression at varying pipe mutation rates for the Network A problem – SSGA, ALCO-GA & PSGA

The results in the following table show that PSGA has much less performance sensitivity than both ALCO-GA and the SSGA especially in terms of tournament size variation, which seems to have very little impact on the overall performance of the algorithm. As seen on other problems, both engineering inspired algorithms are more sensitive to changes in mutation rate than tournament size, although the mutation rate performance sensitivity of ALCO-GA is not much larger than its tournament size sensitivity.

Table 5-30 Algorithm tournament size and pipe mutation rate sensitivity for the Network A problem – ALCO-GA & PSGA

Algorithm Sensitivity - Network A

Algorithm	*Tournament Size*	*Pipe Mutation Rate*
ALCO-GA	0.8021	0.8477
PSGA	0.0411	0.2960

5.3.2.7 Network B

As with the other real world large scale network (Network A) and larger networks on test PSGA exhibits a tendency to perform best at higher tournament sizes as does ALCO-GA, however from the results in Figure 5-69 it appears that ALCO-GA reaches peak performance at a tournament size of approximately 0.06 and starts to drop off whereas PSGA seems to continue to improve, possibly beyond the scope of this experiment's bounds.

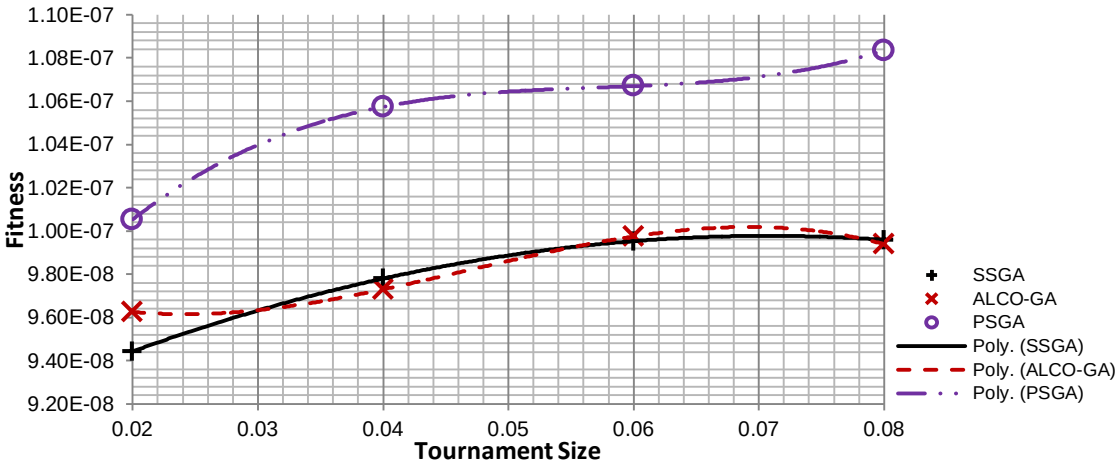


Figure 5-69 Average fitness and polynomial regression at varying tournament sizes for the Network B problem – SSGA, ALCO-GA & PSGA

The following figure shows the mean fitness values and polynomial regression curves over a range of mutation rates for the Network B problem. It can be observed that PSGA greatly outperforms ALCO-GA at all mutation rates tested. There is also less variation between the fitness values produced by PSGA compared to those from ALCO-GA and especially SSGA.

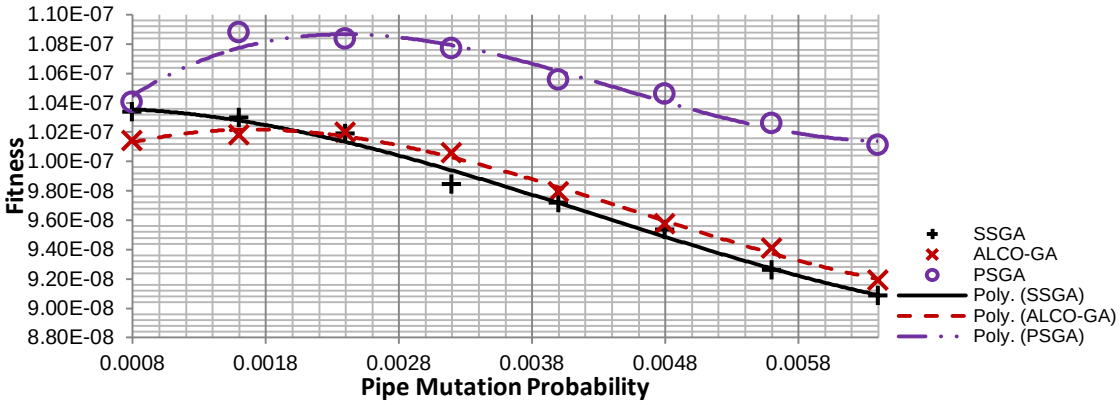


Figure 5-70 Average fitness and polynomial regression at varying pipe mutation rates for the Network B problem – SSGA, ALCO-GA & PSGA

From the following table it can be seen that ALCO-GA only marginally performs better than the SSGA in terms of performance sensitivity for both parameters, unlike PSGA which consistently produces higher quality solutions than both competing algorithms and displays less sensitivity to parameter value variation.

Table 5-31 Algorithm tournament size and pipe mutation rate sensitivity for the Network B problem – ALCO-GA & PSGA

Algorithm Sensitivity - Network B

Algorithm	*Tournament Size*	*Pipe Mutation Rate*
ALCO-GA	0.9904	0.9551
PSGA	0.2550	0.2815

5.3.3 Conclusion

In this section the two engineering inspired genetic algorithms (ALCO-GA and PSGA) are compared to one another and the steady state genetic algorithm in terms of mean fitness performance and sensitivity to parameter value

changes. In every case the Pipe Smoothing Genetic Algorithm displays lower sensitivity to both tournament size and pipe mutation rate compared to the Adaptive Locally Constrained Genetic Algorithm. Another very consistent trend observed from these results is that both engineering inspired algorithms are commonly more sensitive to mutation rate variations than tournament size, this behavior was observed and explored in 4.1.2.1.1 where is was found that mutation rate had a much higher impact on the performance of SSGA than tournament size. The gains in performance from the engineering inspired algorithms compared to SSGA tend to occur at higher mutation rates, a behavior that supports the notion that the engineering heuristic based mutation operators are tempering excessive exploration at high mutation rates, greatly improving the operational robustness of the algorithm.

5.4 Conclusion

The core focus of this work is to explore ways of integrating engineering knowledge into evolutionary algorithms and although the heuristic based mutation operators go some way to include the basic rules applied by water systems engineers into an evolutionary algorithm, there are still objectives and processes applied by practising engineers that are hard to quantify and integrate into an evolutionary algorithm as many of them are somewhat subjective; based on the engineer's situational experience of designing real world water systems. Therefore a practising water systems engineer would find it difficult to integrate a standard genetic algorithm like the SSGA into the classic water network design process due to the time it takes to achieve a near optimal solution. However, it has been observed that both ALCO-GA and PSGA can reach near optimal solutions in considerably less time than the SSGA whilst promoting engineering feasibility in the form of network smoothness and satisfying hydraulic constraints. Such performance could feasibly be applied to an engineer's design process by providing a good starting point for the engineer to then develop and refine utilising their water systems knowledge and experience potentially speeding up the overall design process.

An interesting finding of this work has been the relationship between the application of engineering heuristics and mathematical optimality. The initial notion when applying these heuristics to a standard genetic algorithm was that

they would aid the algorithm to improve the engineering feasibility of the resultant solutions whilst preserving mathematical optimality. It has been demonstrated that the addition of the engineering heuristics has improved the algorithm's ability to produce very competitive solutions often with higher fitness values that those produced by the SSGA. Another important observation is that the heuristic methods tend to find the feasible solution space considerably quicker than the SSGA for the majority of networks. This behaviour becomes important in cases where evaluating the objective function is very expensive and only a limited number of solutions can be evaluated in a feasible timeframe.

Finally the parameter robustness experiments have demonstrated that the heuristic methods are much more forgiving in terms of their sensitivity to varying algorithm parameter values than the standard algorithm. This set of experiments addresses the second research question posed in this thesis showing that the inclusion of water systems knowledge does significantly impact the sensitivity of an EA to parameter variance. Through the comparison of both engineering inspired algorithms it can be seen that ALCO-GA exhibits less sensitivity to parameter variance than the SSGA although in some cases the improvement is marginal, in general PSGA greatly outperforms both ALCO-GA and SSGA on all test problems. An important observation is that although the engineering inspired algorithms perform similarly to the SSGA at lower mutation rates, at higher rate of mutation performance is less degraded. At high rates of mutation the engineering inspired algorithms are constraining the exploration of the search space, effectively guiding the search to more feasible areas. This means that non-experts in the field of meta-heuristics will potentially be able to achieve greater performance from the engineering heuristic based algorithms without the need for specialist knowledge and large amounts of computational resources.

Chapter 6: Multi-Objective Heuristic Based Genetic Algorithms

Chapter 4 applied the heuristics presented in chapter 3 to a steady state GA and presented extensive experiments involving the effect parameter value changes have on the performance of an SSGA and the newly presented engineering heuristic based genetic algorithms; ALCO-GA and PSGA. However, water systems problems in the real-world normally have more objectives which are often conflicting. Due to this, it was decided to evaluate the performance of the engineering heuristic based approaches when they are applied to multi-objective water systems problems. In this chapter the two engineering heuristics are applied to NSGA-II, an established multi-objective GA, popular in engineering research and evaluated on a number of multi-objective water distribution network design problems. This work builds upon the experimentation presented in [145].

6.1 Method

Two formulations of the multi-objective Water Distribution Network (WDN) design problem are presented, including a novel formulation which involves the use of a network smoothing objective. NSGA-II was chosen as a base upon which the engineering heuristics are applied, integration of these heuristics was implemented in the same manner as with the single-objective steady state algorithm presented earlier in this work. As with the previous experiments, the hydraulic modelling of the problem solution was handled by the EPANET hydraulic simulation engine [146]. The following sets of experiments are conducted on the benchmark networks used in the previous single-objective experimentation and were chosen for their varying size and complexity. To assess the performance of the engineering inspired heuristics in the multi-objective domain, the newly presented algorithms are directly compared with the standard formulation of NSGA-II on all benchmark problems.

6.1.1.1 Multi-objective WDN Design Problem

Section 3.2 presents the basic formulation of the multi-objective WDN design problem. As stated earlier, this chapter presents two variations on this formulation; one dual-objective problem and one tri-objective problem. The three objectives used in the following set of experiments are reiterated below:

The total network cost or infrastructure cost which is given by:

$$f(D_1, \dots, D_n) = \sum_{i=1}^{N} c(D_i, L_i) \qquad (28)$$

Where $c(D_{i,}L_i) =$ cost of pipe i with diameter D_i and length L_i with $N =$ number of pipes in the network. This function is to be minimized during the optimization process. The second objective is to minimize the total head deficit within the network and is given by the following expression:

$$f(H_i, \dots, H_j) = \sum_{i=1}^{J} (H_i) \qquad (29)$$

Where the head deficit in junction *i* is H_i with *J* = the number of junctions present in the network. The final objective used in this formulation of NSGA-II for the optimization of least cost water distribution networks is a measure of network smoothness. A smooth network is achieved when pipes can be seen to 'smoothly' transition from large to small diameters from source to the extremities of the network. In this case the objective is to minimize the number of pipe smoothing violations in a candidate network and is given by the following expression:

$$f(S_1, \dots, S_n) = \sum_{i=1}^{N} (S_i) \qquad (30)$$

Where the smoothing violations of pipe *i* is S_i with *N* = the number of pipes in the network. For example, in the case where a pipe *i* violates the smoothing rule $S_i = 1$ otherwise if the rule is satisfied $S_i = 0$.

The first experiment presented in this chapter is the dual-objective formulation of the WDN design problem which uses the first two objectives stated above, total network cost (25) and total head deficit (26). The final experiment in this chapter involves the two objectives used in the first experiment with the addition of the pipe smoothing violations objective (27).

6.2 Experimental Setup

This section describes a novel application of engineering based heuristics, the Hydraulic Deficit and Pipe Smoothing approaches (3.4), to the two multi-objective WDN design problems presented in the previous section.

6.2.1 Experimental Data

The following experiments were conducted on the benchmarks presented and described in section 3.3, most of which are well known WDN design problems of varying size and complexity. As with the single-objective algorithms presented in this work, the solutions generated are assessed using EPANET 2 which provides all necessary hydraulic information.

6.2.2 Algorithms for Comparison

This section details the formulation and implementation of the three algorithms being compared in the following set of experiments. As with the single-objective algorithms presented in the previous chapter, the heuristic based multi-objective algorithms presented here are both developed from an established algorithm, in this case the Non-dominated Sorting Genetic Algorithm (NSGA-II) [63].

6.2.2.1 Non-dominated Sorting Genetic Algorithm

The Non-dominated Sorting Genetic Algorithm (NSGA-II) [63] is a multi-objective evolutionary algorithm which utilises a fast non-dominated sorting approach which decreases computational complexity compared with other non-dominated sorting approaches. Although presented over a decade ago NSGA-II is still considered a good benchmark algorithm as it performs well in a wide range of problem domains, producing a good spread of solutions and convergence which is close to the true Pareto-optimal front. A detailed formulation and implementation of NSGA-II is presented in section 3.2.2. As stated, this implementation of NSGA-II forms the base algorithm upon which the two engineering inspired heuristics are applied.

6.2.2.2 Multi-objective Adaptive Locally Constrained Genetic Algorithm

The Multi-objective Adaptive Locally Constrained Genetic Algorithm (MOALCO-GA) applies a constraint based rule directly to the genotype without evaluating the effect this process has on the phenotype. The heuristic employed by MOALCO-GA is developed from the constraints of a specific problem and remains constant throughout the evolutionary process. The heuristic is applied to a solution through the mutation operator; where the probability of the heuristic mutation operator being applied is directly influenced by the rate of convergence of the population. It is the aim of this operator to guide the algorithm's search to the feasible solution space in a fast and efficient manner utilising hydraulic data from previous fitness evaluations. As with the single-objective version of this algorithm shown in Section 4.3, MOALCO-GA mutation operator does not perform any additional partial or full fitness evaluations, except a single hydraulic simulation at initialisation.

MOALCO-GA is essentially NSGA-II but with some additional features; these include a heuristic based mutation operator and a hypervolume gradient monitor. The Heuristic based Mutation Operation (HMO) is designed to allow the algorithm to locate feasible network designs earlier in the optimisation. It can be configured for use with any appropriate objectives, but here the application to network hydraulic performance only is considered. MOALCO-GA employs the hydraulic deficit based heuristic detailed in Section 3.4.1

It was found that if the HMO was employed exclusively (i.e., without bitwise mutation) throughout the evolutionary process, the population would become stagnant and prematurely converge on a sub optimal solution. Therefore it was necessary as with the single objective version of this algorithm to limit the amount the algorithm employs the HMO. The algorithm employs the Fitness Gradient Monitor (FGM) presented in Section 4.3 to control the application probability of the HMO. However, in this case the FGM tracks the hypervolume of the population as this replaces the fitness value from the SSGA.

6.2.2.3 Multi-objective Pipe Smoothing Genetic Algorithm

The Multi-Objective Pipe Smoothing Genetic Algorithm (MOPS-GA) is based around the principle that in a water distribution network (WDN) the diameter of any pipe is never greater than the sum of the diameter(s) of the directly upstream pipes. Networks that adhere to this rule can be seen to 'smoothly' transition from large to small diameters from source to the extremities of the network. This rule is routinely and implicitly applied by engineers when designing such networks as it makes little sense to follow a smaller diameter pipe with a larger one in the majority of circumstances. The larger pipe will cost more to install and will not add to the hydraulic capability of the system as it will be constrained by the smaller diameter pipe upstream. One further negative aspect of this arrangement is that velocities will be lower in the larger pipe and high water age can become an issue. A standard Multi-Objective Genetic Algorithm (MOGA) of course will mutate some of these conflicting pipe selections from the final solution as they have a corresponding improvement in the cost function and no hydraulic penalty. However in the case of larger networks extensive experimentation has shown that even well-optimised solutions after hundreds of thousands of generations of a standard EA still contain significant numbers of incorrectly sized pipes.

MOPS-GA applies the rule described above directly to the genotype without evaluating the effect this process has on the phenotype (and therefore incurring additional computational cost). The heuristic employed by MOPS-GA is developed from the network topology of a specific problem and remains constant throughout the evolutionary process. The heuristic is applied to a solution through the mutation operator; where the probability of the heuristic being applied is defined by a preset algorithm parameter. It is the aim of the heuristic to guide the algorithm's search to the engineering feasible solution space to locate smoother WDN designs whilst maintaining the performance of a standard MOGA. The MOPS-GA mutation operator does not perform any additional partial or full fitness evaluations, except a single hydraulic simulation at initialisation to determine flow directions. This was an important consideration when developing MOPS-GA as additional fitness evaluations would require further hydraulic evaluations, increasing algorithm run time.

MOPS-GA is in essence a standard version of the Non-dominating Sorting Genetic Algorithm-II (NSGA-II) which incorporates an additional feature; a pipe smoothing heuristic based mutation operator. Details of the pipe smoothing heuristic and modified mutation operator can be found in Section 3.4.2.

6.2.3 Measuring Performance

To enable the comparison of MOALCO-GA, MOPS-GA and NSGA-II the hypervolume indicator [147] [148] was employed. The hypervolume indicator allows the tracking of algorithm convergence and provides a measurement of population diversity. Note that the hypervolume values are normalised from 0 to 1 using the theoretical best (utopia) and worst (nadir) points in the solution space enables the examination of convergence and population diversity. Each of the three algorithms were run 50 times (10 times for Network A & B due to problem complexity and resultant runtime) and the hypervolume results averaged to allow a fair performance comparison to be carried out. In addition, the population hypervolume values produced by each algorithm were compared for statistical significance using the Mann-Whitney U test.

6.3 Results

6.3.1 Dual-objective Experiments

This section presents the results for the dual-objective experimentation conducted on NSGA-II, MOALCO-GA and MOPS-GA. As stated previously the two objectives are the minimization of network cost and minimization of hydraulic deficit. To ensure a fair comparison between the three algorithms, the parameters of NSGA-II were tuned to each problem and the same parameter set was utilized by each algorithm.

6.3.1.1 Hanoi

The following set of results is from the dual-objective Hanoi problem. Table 6-1 presents the best achieved hypervolume and mean hypervolume from the 50 individual runs. These results show that both MOALCO-GA and MOPS-GA achieve a better best hypervolume and average hypervolume than NSGA-II after the 100,000 fitness evaluations. It is also clear that out of the two

newly proposed algorithms MOPS-GA produces superior results. Each algorithm in this case produced statistically different populations of hypervolume results when compared to the other.

Table 6-1. Best & Average Hypervolume Results for the Hanoi Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.7282	0.7201
MOALCO-GA	0.7306	0.7248
MOPS-GA	**0.7441**	**0.7395**

Figure 6-1 shows the average hypervolume from all 50 runs over the 100,000 fitness evaluations for the three algorithms for the Hanoi problem. It can be seen that both of the engineering heuristic based algorithms outperform NSGA-II in the first ~5000 evaluations, however at this point MOALCO-GA starts to converge and produces similar quality results to NSGA-II whilst MOPS-GA goes on to substantially outperform the other two algorithms until the termination of the runs. It is only after 20,000 evaluations that MOALCO-GA starts to achieve better results than NSGA-II. This behaviour is thought to be caused by the change in heuristic application strength; increasing the probability that standard mutation would be utilized instead of the deficit/excess heuristic. It would seem this shift enabled the algorithm to explore the solution space in the later stages of the search more effectively than NSGA-II.

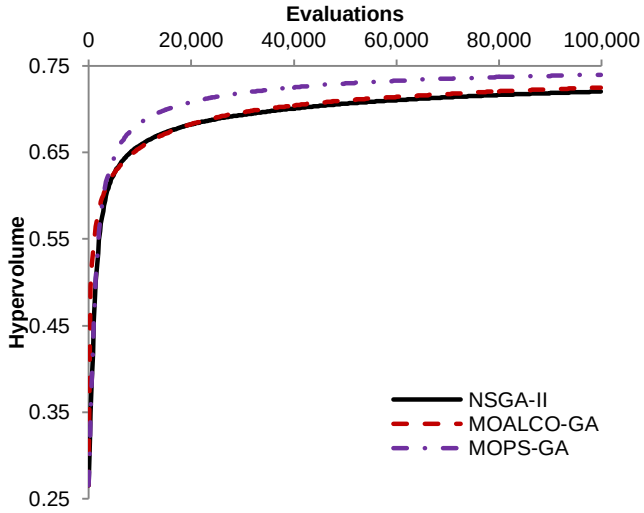


Figure 6-1. Mean Best Hypervolume for the Hanoi Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-2 presents the best (highest hypervolume) populations for the three algorithms after the allotted 100,000 fitness evaluations. It can be observed that the solutions produced by both MOALCO-GA and MOPS-GA mostly dominate the solutions found by NSGA-II, especially at lower network costs. It is not surprising that MOPS-GA achieves more dominant solutions at lower network costs as the pipe smoothing heuristic naturally restricts the selection of larger pipe diameters, hence the algorithm promotes lower cost solutions.

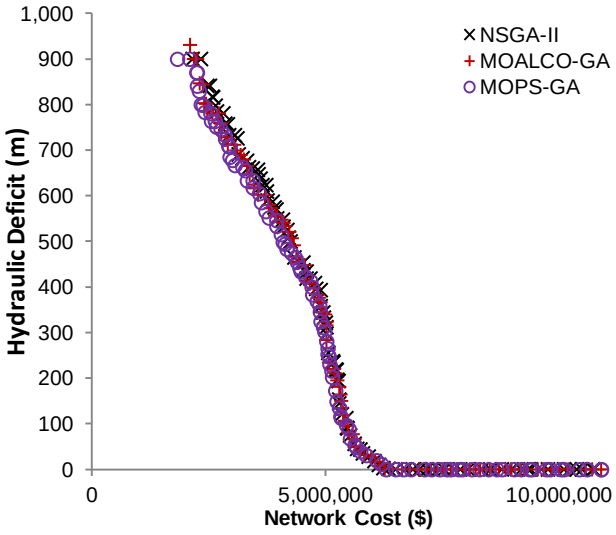


Figure 6-2. Best Final Population for the Hanoi Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.1.2 Foss Poly 1

In the case of the Foss Poly 1 problem we see MOPS-GA achieve better best and average hypervolume results than NSGA-II and MOALCO-GA. Table 6-2 shows that NSGA-II and MOALCO-GA achieve similar hypervolume results with NSGA-II obtaining slightly better results than MOALCO-GA. It was found that there was no statistically significant difference between hypervolume results produced by NSGA-II and MOALCO-GA. However there was found to be statistical difference in results when comparing these two algorithms and MOPS-GA.

Table 6-2. Best & Average Hypervolume Results for the Foss Poly 1 Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.9691	0.9156
MOALCO-GA	0.9653	0.9111
MOPS-GA	**0.9845**	**0.9612**

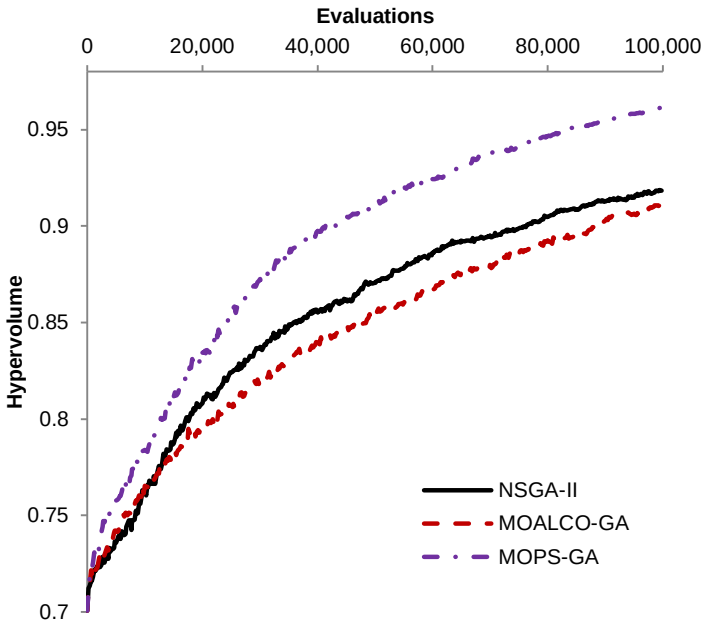


Figure 6-3. Mean Best Hypervolume for the Foss Poly 1 Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-3 shows the average hypervolume results for the Foss Poly 1 problem for the three algorithms. The performance of NSGA-II and MOALCO-GA is similar until around 20,000 evaluations where NSGA-II starts to achieve better hypervolume results. What is clear from this figure is that MOPS-GA outperforms the other two algorithms throughout the entire 100,000 evaluations, achieving a better average hypervolume at all stages of the search.

The best population of solutions for each algorithm is shown in Figure 6-4. From this figure it can be observed that the majority of solutions lie at 0 hydraulic deficit and that MOPS-GA provides the lowest cost solutions compared to the other algorithms. This is again due to the pipe smoothing heuristic restricting the selection of larger pipe diameters, hence reducing the overall cost of the network. It is also evident that MOPS-GA finds lower cost solutions than the other algorithms at all hydraulic deficit levels.

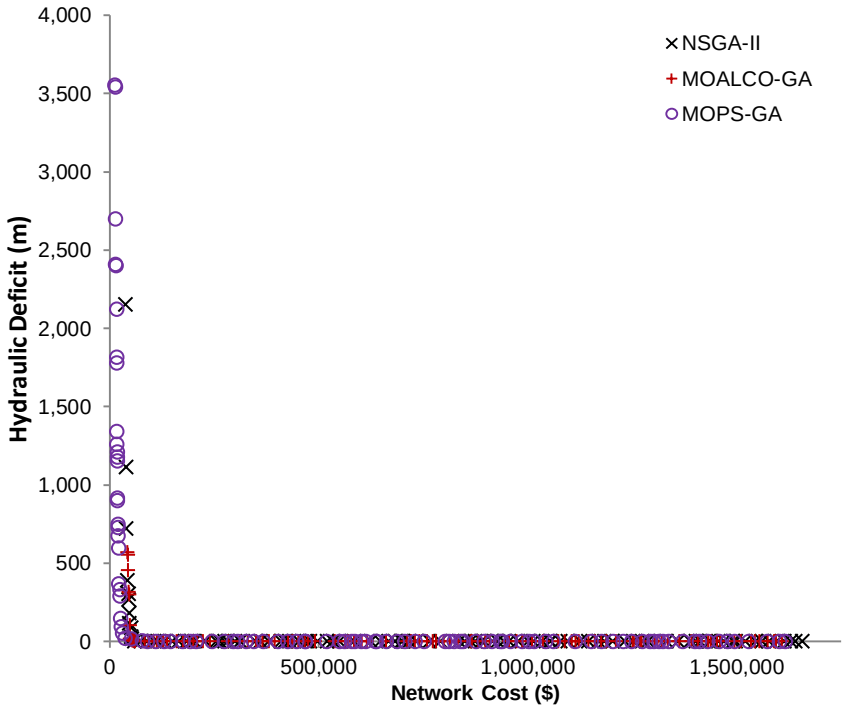


Figure 6-4. Best Final Population for the Foss Poly 1 Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.1.3 New York Tunnels

Table 6-3 gives the hypervolume results from the three algorithms for the New York Tunnels problem. In this case we see MOALCO-GA and MOPS-GA achieve similar hypervolume results when compared to each other, whilst both outperforming NSGA-II. In this instance it was found that there is statistical difference in hypervolume results between NSGA-II and the two engineering heuristic based algorithms. However, there appears to be no significant difference in result population between the two better performing algorithms.

Table 6-3. Best & Average Hypervolume Results for the New York Tunnels Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.9651	0.9614
MOALCO-GA	0.9679	0.9669
MOPS-GA	**0.9681**	**0.9673**

Figure 6-5 shows MOPS-GA clearly outperforming the other two algorithms at all stages of the search and also displays faster convergence. This fast convergence is thought to be due to the MOPS-GA heuristic mutation taking into account the diameters of the parallel pipe(s) of the pipe being mutated, which in the case of a parallel expansion problem such as this, leads to higher quality solutions in less fitness evaluations. Although MOALCO-GA starts the search underperforming compared with NSGA-II; after 20,000 evaluations the engineering heuristic based algorithm starts to outperform the standard algorithm and eventually reaches the solution quality of MOPS-GA.

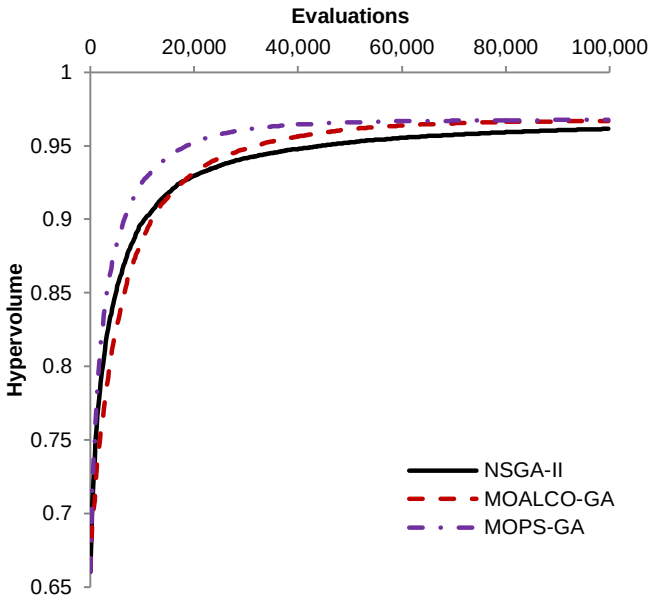


Figure 6-5. Mean Best Hypervolume for the New York Tunnels Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-6 presents the best populations for the three algorithms after the allotted 100,000 fitness evaluations for the New York Tunnels problem. From this figure it can be observed that all three algorithms produce a very similar spread of results with the engineering heuristic based algorithms dominating the solutions of NSGA-II at lower network costs, especially at low values of hydraulic deficit.

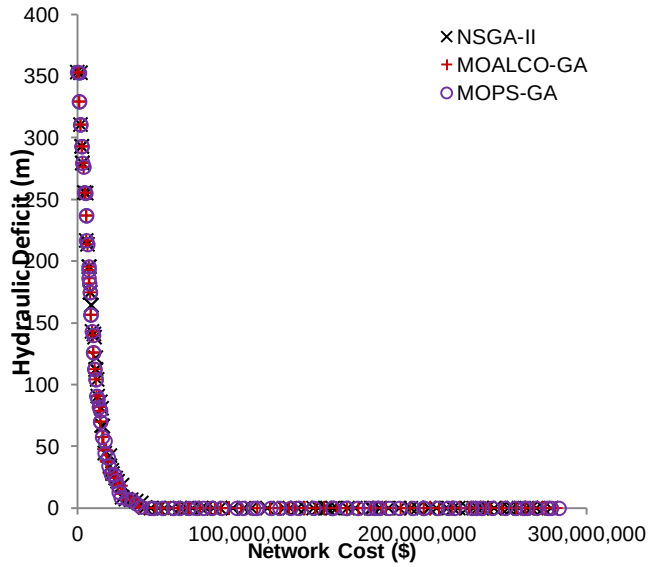


Figure 6-6. Best Final Population for the New York Tunnels Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.1.4 Modena

The best and mean hypervolume results for the Modena problem are presented in Table 6-4. This shows that following the 100,000 allotted evaluations MOPS-GA attains a much higher hypervolume value than the other two algorithms which both achieve very similar quality solutions. In the case of these results, statistical testing reveals no significant difference in the population of results between NSGA-II and MOALCO-GA, however MOPS-GA does produce a population of results with statistically different results than the other two algorithms.

Table 6-4. Best & Average Hypervolume Results for the Modena Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.7691	0.7268
MOALCO-GA	0.7664	0.7194
MOPS-GA	**0.8414**	**0.8051**

The performance difference between MOPS-GA and the other two algorithms is illustrated in Figure 6-7. MOPS-GA outperforms the other two algorithms significantly throughout the entire 100,000 evaluation search, ultimately achieving a much higher average hypervolume than NSGA-II and MOALCO-GA. MOALCO-GA does display better perform than NSGA-II up until around 80,000 evaluations.

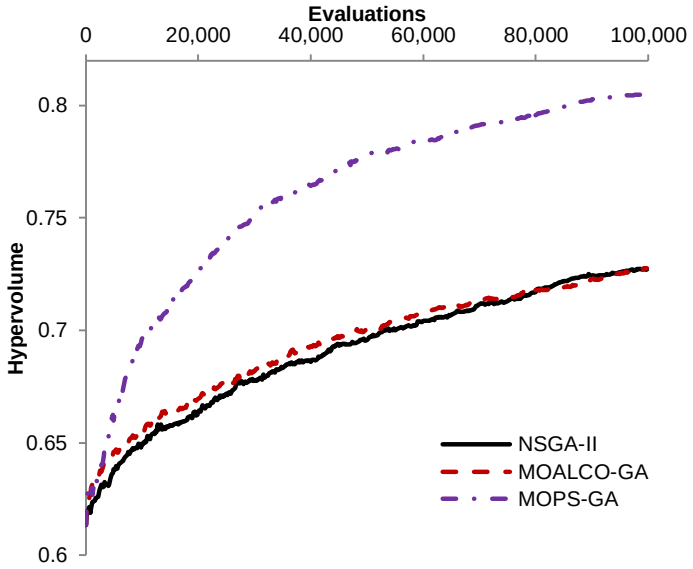


Figure 6-7. Mean Best Hypervolume for the Modena Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-8 shows the best performing populations for the three algorithms after the allotted 100,000 fitness evaluations for the Modena problem. It is clear from these results that MOPS-GA achieves much lower network cost solutions at zero hydraulic deficits compared with the other competing algorithms.

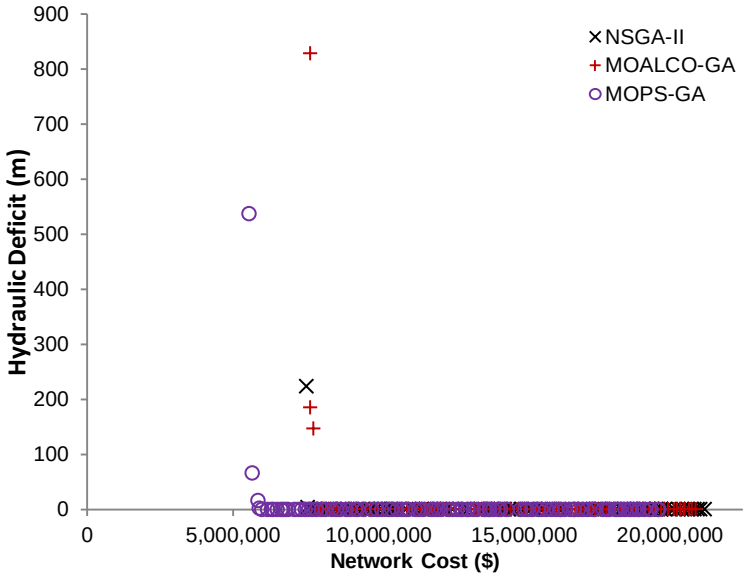


Figure 6-8. Best Final Population for the Modena Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

It is thought that the pipe smoothing heuristic employed by MOPS-GA is very effective when applied to a multi-source (reservoir) configuration such as that of the Modena problem. Interestingly, the majority of solutions (>95%) have zero hydraulic deficit, with only a small number of solutions with a hydraulic deficit. As observed in the single-objective experiment in chapter 4, the hydraulic requirements of the Modena problem are very easy to meet and as show previously have a high probability of being satisfied with a randomly generated solution.

6.3.1.5 Network A

Table 6-5 presents the best and mean results achieved for the three algorithms following 100,000 fitness evaluations for the Network A problem. As can be seen from the results, MOPS-GA achieves the best hypervolume for this problem, followed by NSGA-II and finally MOALCO-GA. Each algorithm in this case produced statistically different populations of hypervolume results when compared to each other.

Table 6-5. Best & Average Hypervolume Results for the Network A Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.9347	0.9200
MOALCO-GA	0.9274	0.9122
MOPS-GA	**0.9417**	**0.9305**

The average hypervolume obtained by the three algorithms during the 100,000 evaluation search is displayed in Figure 6-9. The plot shows that all three algorithm perform equally during the first 10,000 evaluations, however following this point MOPS-GA starts to outperform NSGA-II, achieving a better average hypervolume throughout the remainder of the search. ALCO-GA does not perform as well as NSGA-II following the first 10,000 iterations.

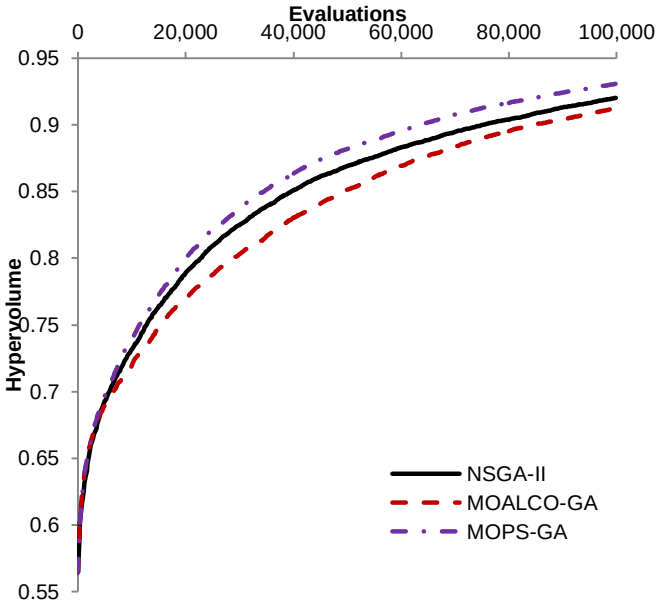


Figure 6-9. Mean Best Hypervolume for the Network A Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

The final best performing populations from each algorithm are presented in Figure 6-10. From this plot it is apparent that the solutions generated by MOPS-GA dominate those produced by the other two algorithms, particularly for the low network cost solutions.

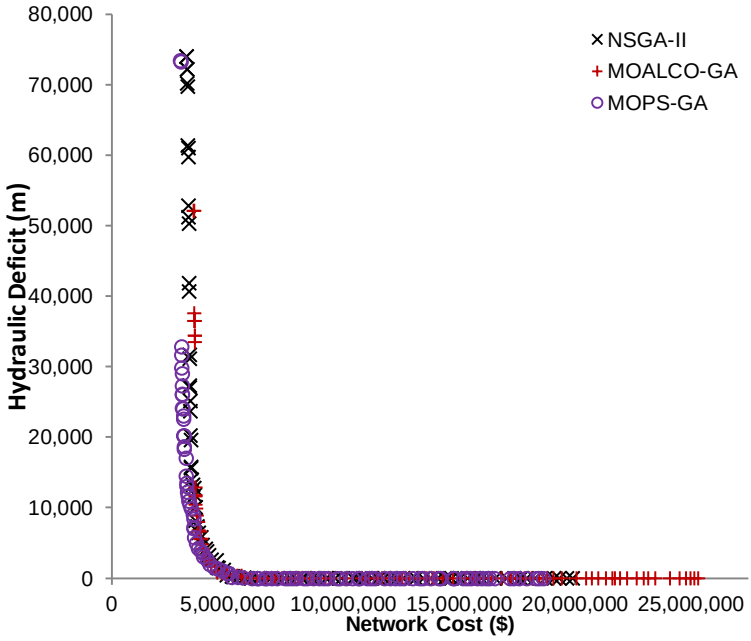


Figure 6-10. Best Final Population for the Network A Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

Although the MOPS-GA solutions dominate those found by NSGA-II, the established algorithm does appear to find a more even spread of solutions especially at higher hydraulic deficit. As seen in previous problems presented in this chapter, MOALCO-GA is prone to locate high cost solutions; this is apparent in this case and is most likely due to the deficit reducing heuristic based mutation present in the algorithm which strongly promotes hydraulic excess. This behaviour coupled with NSGA-II's crowding distance sorting is most likely the primary cause of the higher network cost solutions generated by MOALCO-GA.

6.3.1.6 Network B

Table 6-6 shows the best and average hypervolume values achieved from the 10 runs of each algorithm for the Network B problem. The pattern of results is very similar to the previously presented Network A, with MOPS-GA achieving the best solution quality, followed by NSGA-II and finally MOALCO-GA. Each algorithm in this case produced statistically different populations of hypervolume results when compared to each other.

Table 6-6. Best & Average Hypervolume Results for the Network B Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.9121	0.9050
MOALCO-GA	0.9032	0.8981
MOPS-GA	**0.9180**	**0.9113**

The mean best hypervolume of the three algorithms for the Network B problem is presented in Figure 6-11. Unlike with the Network A problem, MOALCO-GA displays better performance in the early stages of the search compared to the other competing algorithms, however at around 15,000 evaluations the progression of the adaptive algorithm slows and the other two algorithms overtake. It is also at this point where MOPS-GA splits from NSGA-II and starts to outperform the standard algorithm going on to achieve a better overall average hypervolume value.

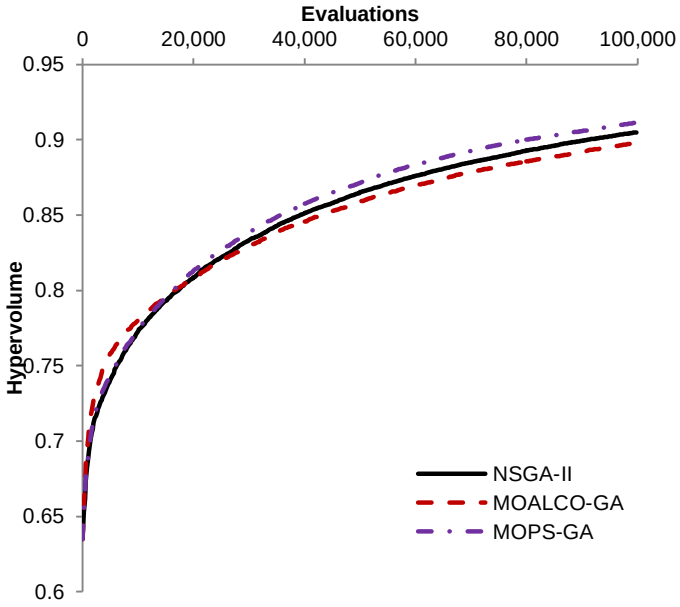


Figure 6-11. Mean Best Hypervolume for the Network B Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

The best final populations generated by the three algorithms for the Network B problem are shown in Figure 6-12. It can be observed that both MOPS-GA and NSGA-II achieve a relatively comparable spread of results apart from at lower network costs where the spread of solutions produced by MOPS-GA is somewhat superior to NSGA-II.

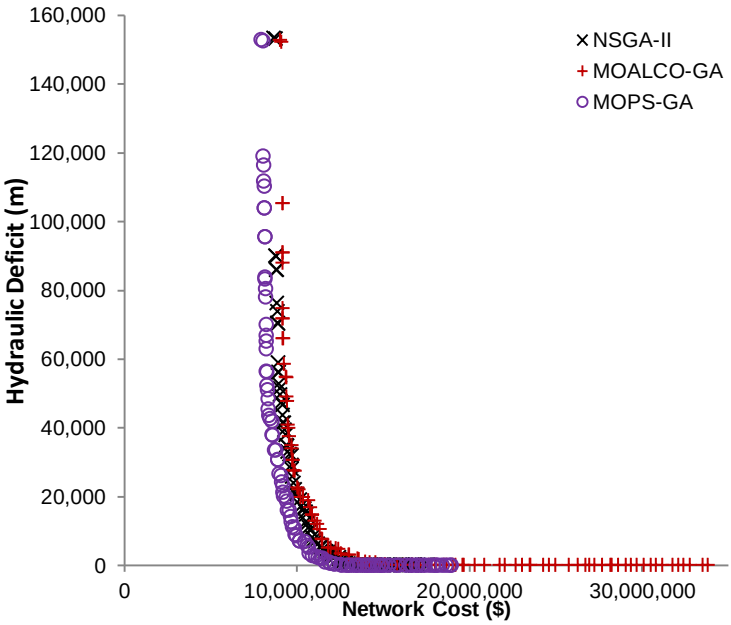


Figure 6-12. Best Final Population for the Network B Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

It is also visible that the solutions produced by MOPS-GA dominate those of NSGA-II and MOALCO-GA, especially by the lower network cost solutions. As with the Network A results presented previously, MOALCO-GA produces a large number of solutions with much higher network costs than the other two algorithms, and in the case of this, the most complex problem from the benchmarks tested, this behaviour is more pronounced.

6.3.2 Tri-objective Experiments

The following set of experiments involves the addition of the pipe smoothing violations objective as stated in 6.1.1.1. In the field of water distribution network design the engineer is often required to consider a number of design parameters some of which are not normally considered when optimizing a network design. One of these considerations is ensuring the diameter of any pipe is never greater than the sum of the diameter(s) of the directly upstream pipes. Networks that adhere to this rule can be seen to 'smoothly' transition from large to small diameters from source to the extremities of the network. This rule is routinely and implicitly applied by engineers when designing such networks as it makes little sense to follow a smaller diameter pipe with a larger one in the majority of circumstances. The larger pipe will cost more to install and will not add to the hydraulic capability of the system as it will be constrained by the smaller diameter pipe upstream. It is often the case when utilising an optimisation method, such as an evolutionary algorithm, that the engineer has to modify the resultant solution to make it adhere to such rules. The inclusion of pipe smoothing violations as an additional objective is intended to aid the algorithm produce high quality solutions which are not only competitive but also more feasible from the perspective of a water systems engineer. As with the dual-objective experiments, NSGA-II was tuned to each problem and the same parameter values were used by each algorithm to ensure a fair comparison.

6.3.2.1 Hanoi

Table 6-7 presents the hypervolume results of the three algorithms for the tri-objective Hanoi problem. It can be seen that MOPS-GA obtains the highest best and average hypervolume values, followed by MOALCO-GA which

achieves better results than NSGA-II. Each algorithm in this case produced statistically different populations of hypervolume results when compared to each other.

Table 6-7. Best & Average Hypervolume Results for the Hanoi Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.6861	0.6674
MOALCO-GA	0.6970	0.6734
MOPS-GA	**0.7108**	**0.6886**

Figure 6-13 displays the average hypervolume value of the 50 individual runs for each of the three algorithms. It can be observed that both of the engineering heuristic based algorithms display increased performance over NSGA-II in the initial stages of the search. Following the preliminary expansion into the search space both MOALCO-GA and NSGA-II begin convergence at a faster rate to that of MOPS-GA. These results in MOPS-GA achieving a superior average hypervolume value compared to that of the other two algorithms.

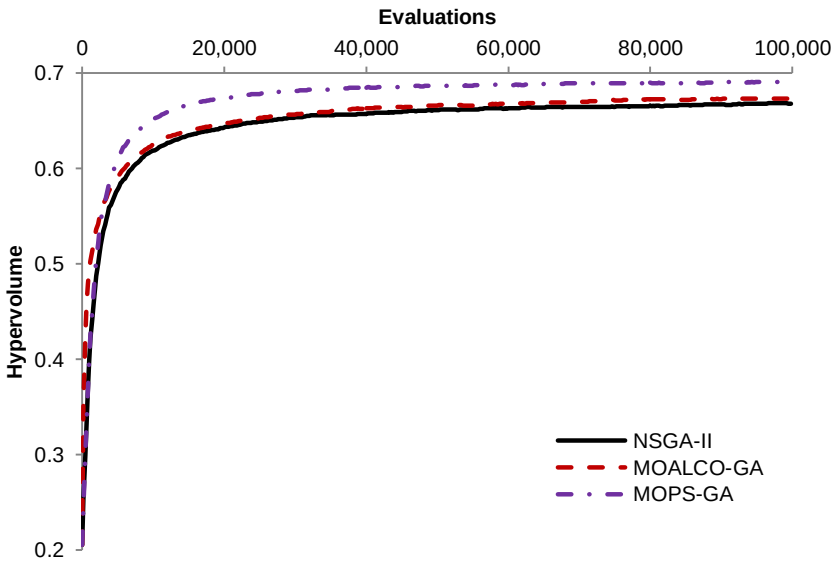


Figure 6-13. Mean Best Hypervolume for the Hanoi Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-14 presents the best final population from each of the algorithms for the Hanoi problem. Due to the tri-objective nature of the problem, the solutions are presented utilising four plots to increase clarity; three 2D figures displaying each side of the 3 dimensional search space and one 3D plot of the same data. It can be observed that the solutions produced by MOPS-GA tend to dominate those produced by the other algorithms at low network costs and high hydraulic deficit values. However, MOALCO-GA does appear to achieve dominant solutions at low hydraulic deficit values. Looking at the second plot it is clear that the solutions produced by MOPS-GA dominate those from the other algorithms in terms of pipe smoothing violations and network cost. Interestingly MOPS-GA produces the solutions with the join highest number of pipe smoothing violations although these have lower cost than solutions generated by NSGA-II which have the same number of violations. The third plot presents the solutions in terms of hydraulic deficit and pipe smoothing violations. As previously observed the solutions found by MOPS-GA are mainly located at low pipe smoothing violations, however this is at the cost of higher hydraulic deficit values. It can also be observed that the majority of MOPS-GA solutions with zero hydraulic deficits have a relatively high number of smoothing violations. Interestingly it is MOALCO-GA and NSGA-II that achieve solutions with the lowest hydraulic deficit at zero pipe smoothing violations, mostly dominating the competing solutions found by MOPS-GA.

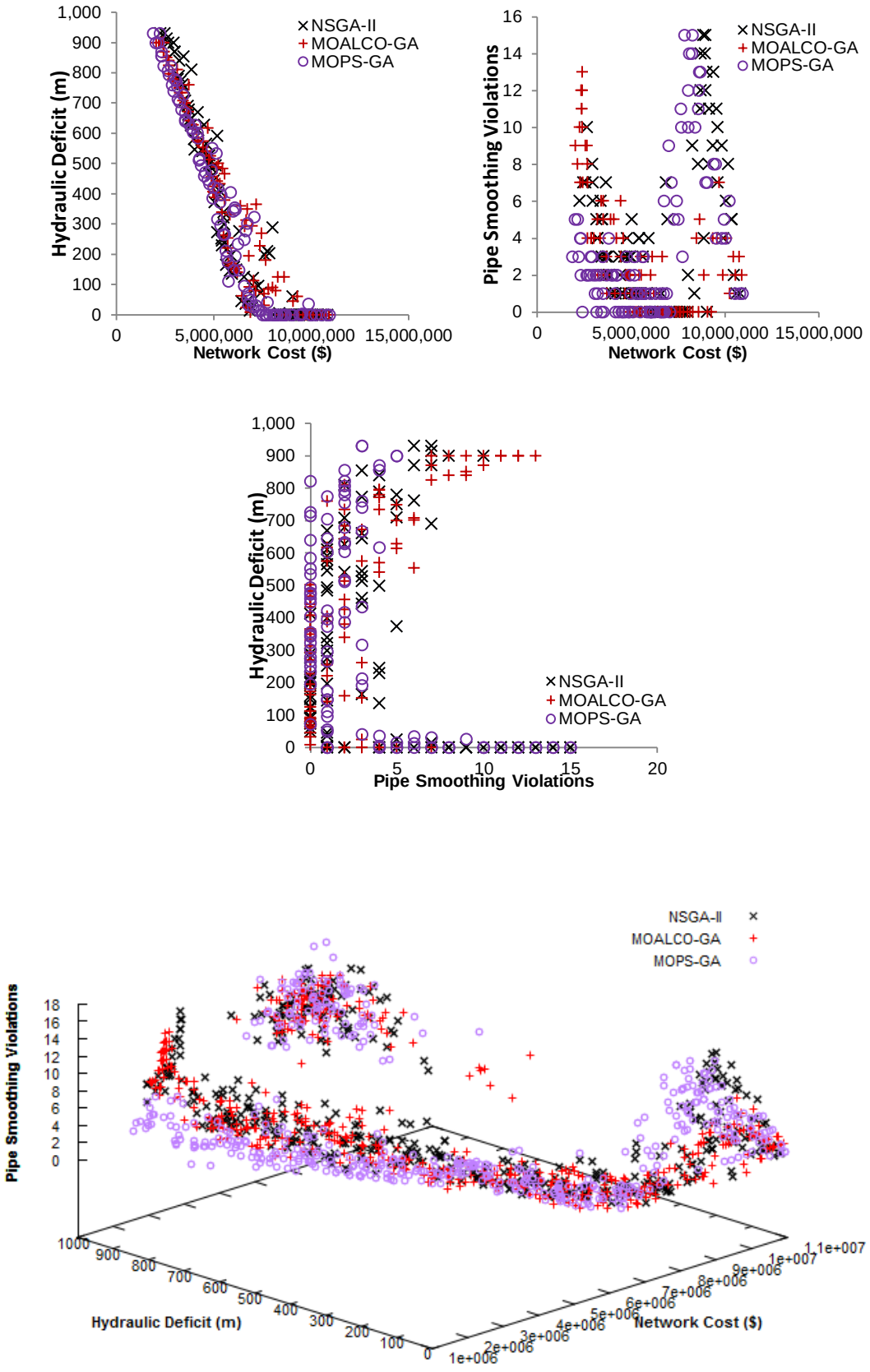


Figure 6-14. Best Final Population for the Hanoi Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.2.2 Foss Poly 1

Table 6-8 presents the best and mean hypervolume results from the three algorithms for the Foss Poly 1 problem. As with the tri-object Hanoi problem, MOPS-GA produces the highest best and average hypervolume values from the 50 runs, followed by MOALCO-GA and finally NSGA-II. Each algorithm in this case produced statistically different populations of hypervolume results when compared to each other.

Table 6-8. Best & Average Hypervolume Results for the Foss Poly 1 Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.9539	0.9166
MOALCO-GA	0.9555	0.9275
MOPS-GA	**0.9811**	**0.9594**

The mean best hypervolume values from the three algorithms for the Foss Poly 1 problem are presented in Figure 6-15. It is first apparent that MOPS-GA clearly outperforms the other two algorithms throughout the entire duration of the search. With regard to MOALCO-GA and NSGA-II, both algorithms display similar performance in the first 30,000 solutions evaluations, however following this point MOALCO-GA starts to produce populations with higher hypervolume values than the standard algorithm.

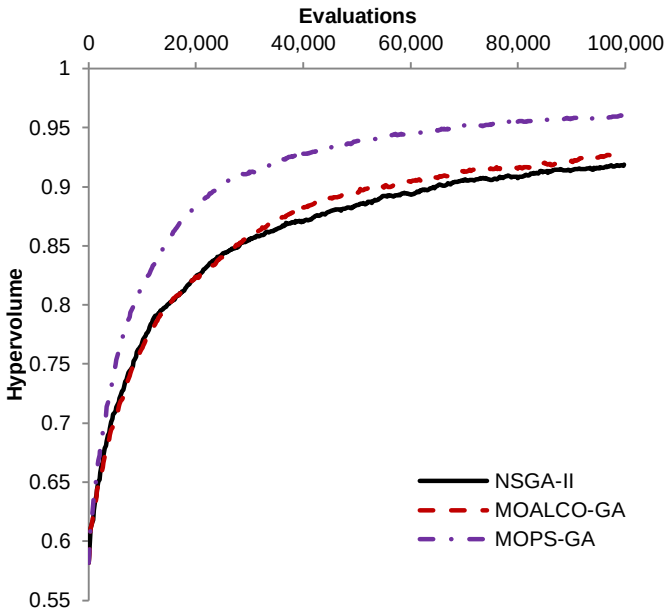


Figure 6-15. Mean Best Hypervolume for the Foss Poly 1 Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-16 shows the best population produced by each of the three algorithms for the Foss Poly 1 problem. It can be observed from the first plot that although MOPS-GA achieves a good spread of solutions which dominate solutions produced by the other two algorithms at low network costs, it fails to produce solutions with low cost and zero hydraulic deficit. The second plot shows MOPS-GA producing a dominant group of solutions with very low network cost and pipe smoothing deficit compared to the other two algorithms. The relationship between hydraulic deficit and smoothing violations is presented in the third plot and it shows that MOALCO-GA and NSGA-II are able to find good quality solutions at low cost and low hydraulic deficit, unlike MOPS-GA which struggles to find solutions with both a low cost and low deficit. As was the case in the Hanoi problem, the majority of MOPS-GA solutions with zero hydraulic deficits have a relatively high number of smoothing violations.

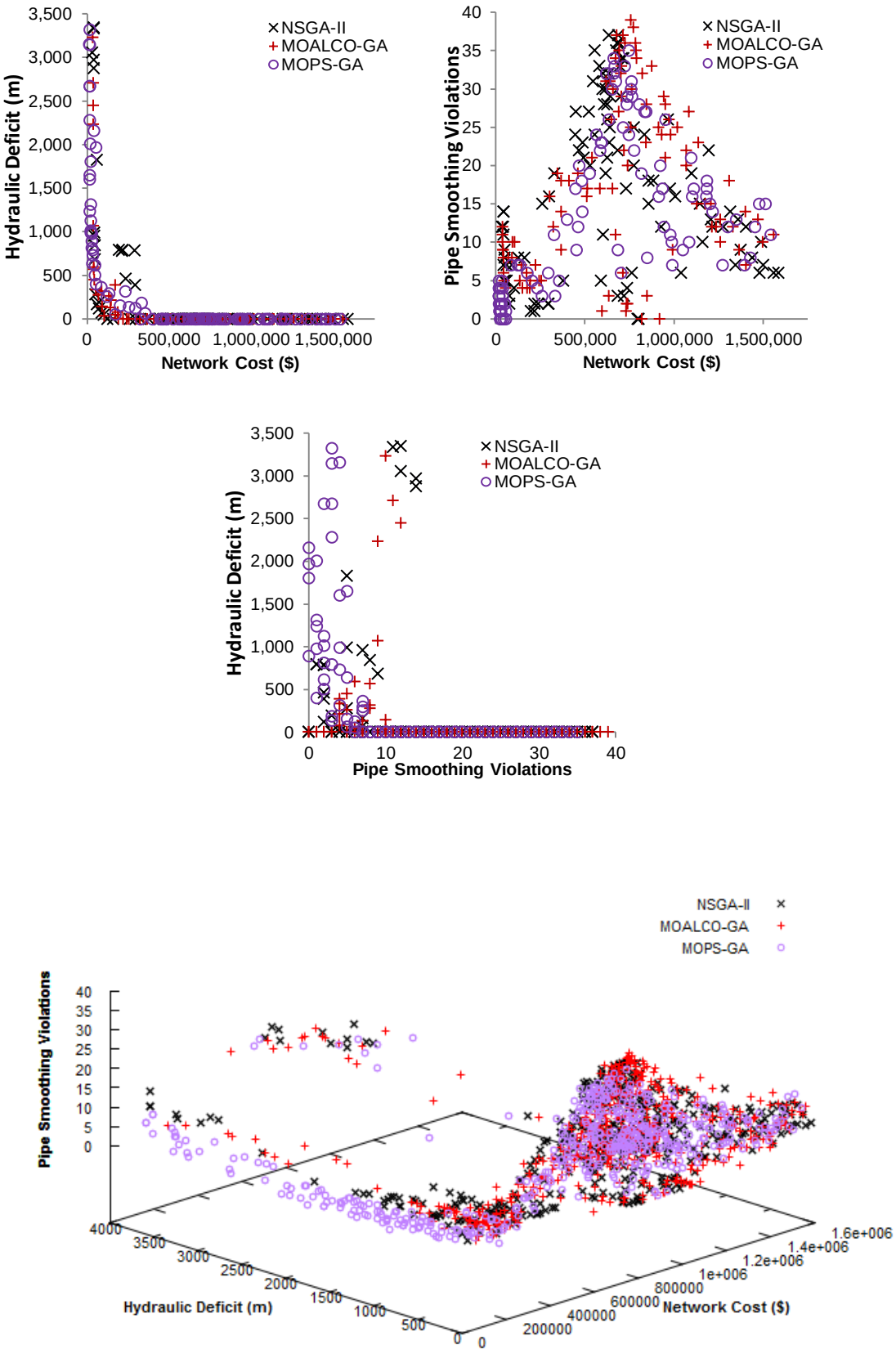


Figure 6-16. Best Final Population for the Foss Poly 1 Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.2.3 New York Tunnels

The best and average hypervolume results from the three algorithms for the New York Tunnels problem are presented in Table 6-9. In this case MOALCO-GA produces both the best performing set of results for this problem followed by NSGA-II and finally MOPS-GA. The population of hypervolume results produced by NSGA-II and MOPS-GA were found not to be statistically different. However MOALCO-GA was found to produce statistically significant results when compared to the other two algorithms.

Table 6-9. Best & Average Hypervolume Results for the New York Tunnels Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.8933	0.8097
MOALCO-GA	**0.9112**	**0.8289**
MOPS-GA	0.8989	0.8053

Figure 6-17 displays the average best hypervolume values from the three algorithms over the 100,000 evaluation search for the New York Tunnels problem. It can be seen from these results that both of the engineering heuristic base algorithms perform better than NSGA-II for the majority of the allotted search.

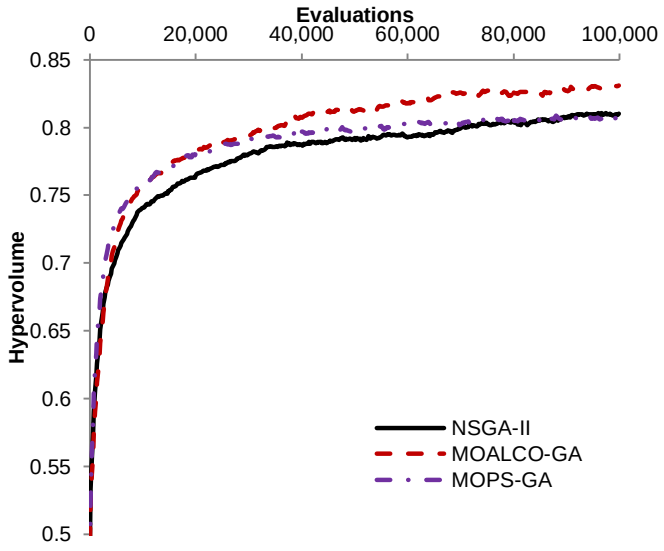


Figure 6-17. Mean Best Hypervolume for the New York Tunnels Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

At around 30,000 solution evaluations MOALCO-GA starts to outperform MOPS-GA which displays what looks like early convergence and is eventually overtaken by the standard algorithm at approximately 80,000 evaluations.

Figure 6-18 displays the best final population from each of the algorithms for the New York Tunnels problem. We see that the MOPS-GA individuals mostly dominate the solutions produced by the other algorithms especially at higher hydraulic deficit values, although at values near zero MOALCO-GA produces the dominant solutions in terms of network cost and hydraulic deficit. Interestingly it is NSGA-II and MOALCO-GA that achieve the most number of solutions with zero pipe smoothing violations instead of the MOPS-GA which promotes smooth pipe diameter transitions. MOPS-GA actually produces the solutions with the highest number of smoothing violations. This is a very interesting result as MOPS-GA was the best performing algorithm for the dual-objective formulation of the New York Tunnels problem and one would expect with the addition of an objective which reflects the pipe-smoothing heuristic that the performance would be enhanced, however this is not the case.

This could be due to the parallel expansion nature of the New York Tunnels problem. The original network in terms of the pipe smoothing objective has no smoothing violations, therefore a solution that does not add a single parallel pipe to the network would have very good pipe smoothing and network cost objective values, although this solution would perform extremely poorly when it comes to hydraulic head. The combination of the pipe smoothing objective and MOPS-GA heuristic results in a relatively strong bias towards smooth pipe transitions, and unlike other problem types where this bias has proven beneficial, in a parallel expansion problem such as this MOPS-GA is prevented from effectively exploring the solution space in the later stages of the search.

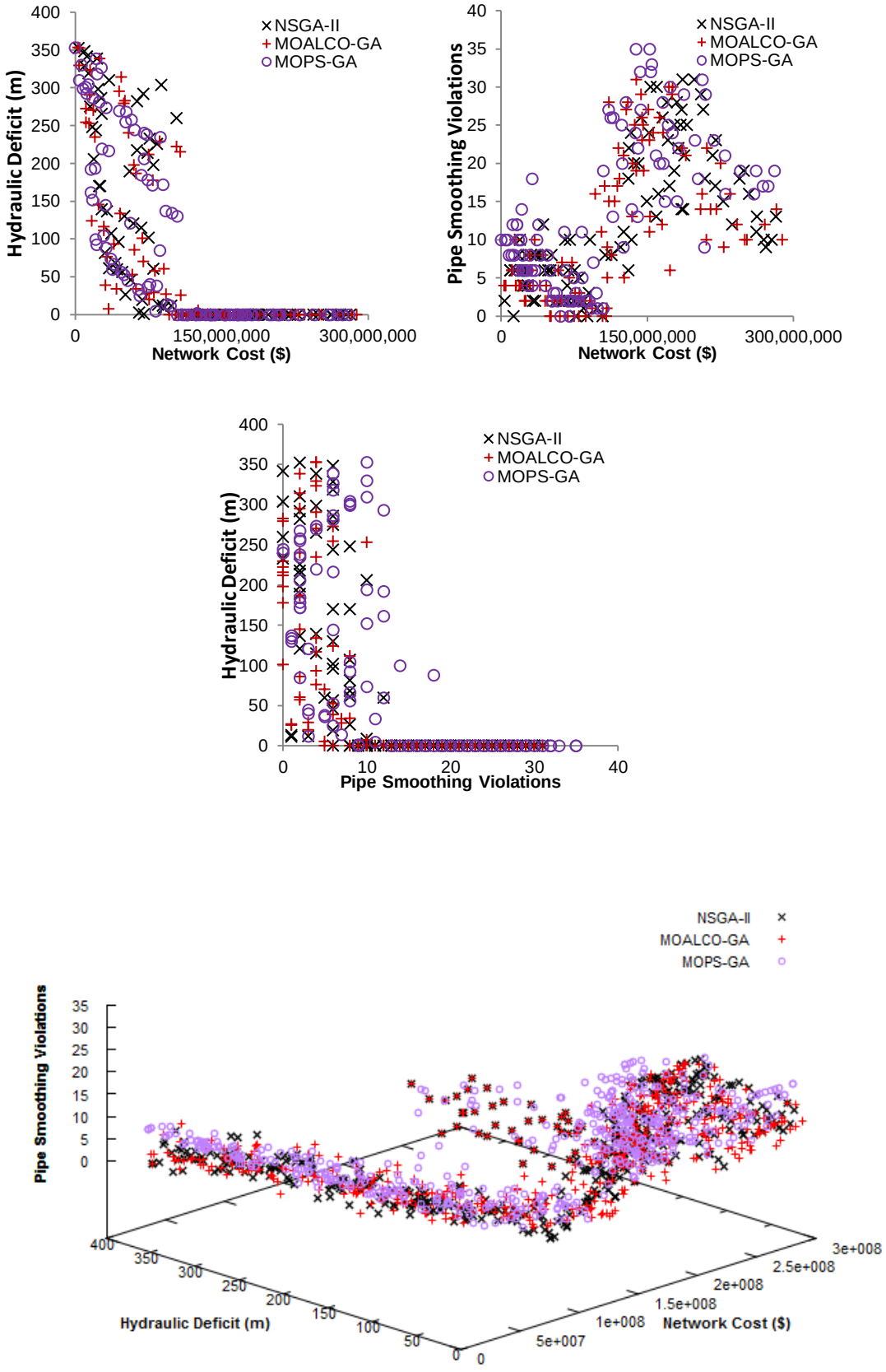


Figure 6-18. Best Final Population for the New York Tunnels Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.2.4 Modena

Table 6-10 presents the best and average hypervolume results for NSGA-II, MOALCO-GA and MOPS-GA for the tri-objective Modena problem. It is apparent from these results that MOPS-GA is able to generate populations with significantly higher hypervolume values than the other two algorithms. No statistical significance in results was found between NSGA-II and MOALCO-GA, although MOPS-GA produced statistically significant results when compared to the other two algorithms.

Table 6-10. Best & Average Hypervolume Results for the Modena Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.6000	0.5795
MOALCO-GA	0.6117	0.5812
MOPS-GA	**0.6720**	**0.6463**

The mean best hypervolume results for the three algorithms for the Modena problem are presented in Figure 6-19. It is observed that MOPS-GA drastically outperforms the other two algorithms throughout the entire search of the algorithms. Whilst MOALCO-GA does achieve better hypervolume results in the early stages of the search compared to NSGA-II, the difference in performance between the two algorithms diminishes in the later stages of the search. It should be noted that it takes under 20,000 evaluations for MOPS-GA to achieve the highest average hypervolume achieved by both MOALCO-GA and NSGA-II.

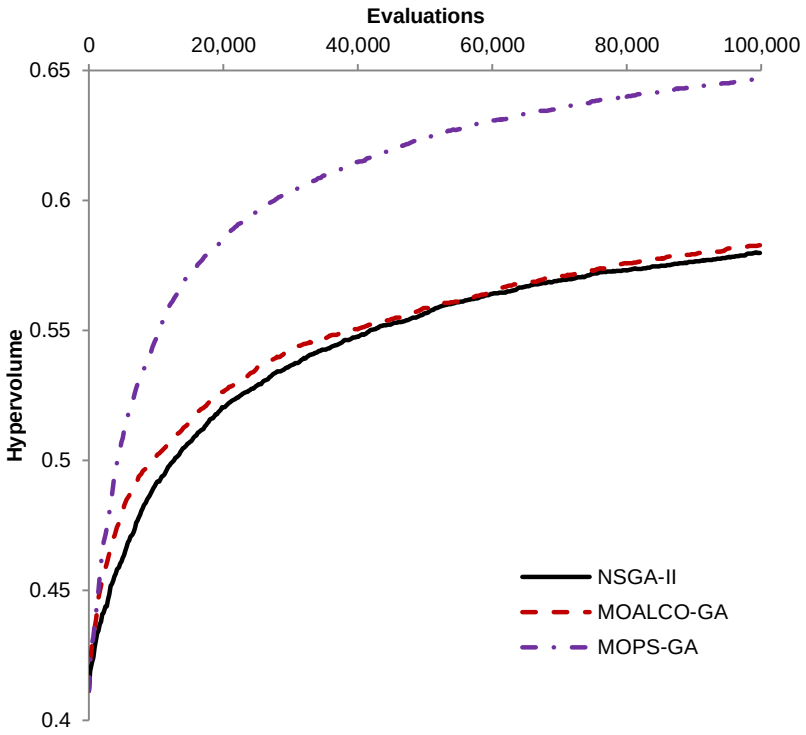


Figure 6-19. Mean Best Hypervolume for the Modena Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-20 displays the best final population of solutions generated by the three algorithms after 100,000 evaluations for the Modena problem. It is apparent from the first plot that MOPS-GA is able to find the lowest cost solutions, followed by the other two algorithms, although this is done at the cost of increased hydraulic deficit. The second plot shows the ability of MOPS-GA to find a good number of smoother, low cost solutions compared to the other two algorithms. It can also be observed that although MOALCO-GA and NSGA-II do achieve solutions with similar network smoothness they are mainly located at much higher network costs. Being a larger network, it is more difficult to find solutions with very smooth pipe diameters transitions and hence why the lowest number of pipe smoothing violations generated by a solution is 60. Looking at the third figure, it is apparent that although MOPS-GA achieves the most solutions with the least number of pipe smoothing violations, the majority of these solutions have relatively high hydraulic deficit.

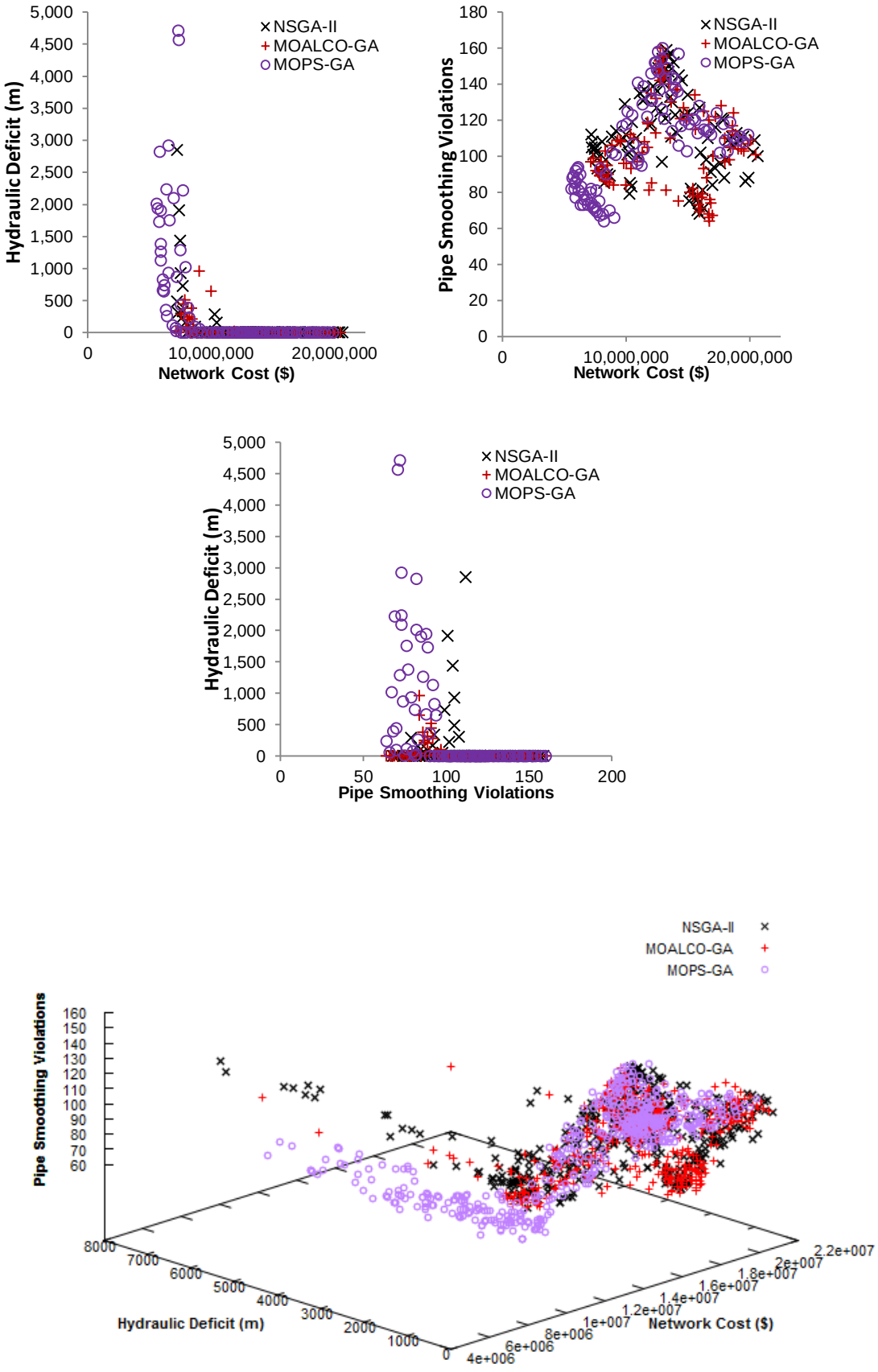


Figure 6-20. Best Final Population for the Modena Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.2.5 Network A

The best and average hypervolume results for the Network A problem produced by the three algorithms are shown in Table 6-11. From these results it can be seen that both NSGA-II achieve very similar results, with MOALCO-GA finding the best single solution out of the two. However it is MOPS-GA which attains the highest performance, finding the best single hypervolume and average hypervolume results. No statistical significance in results was found between NSGA-II and MOALCO-GA, although MOPS-GA produced statistically significant results when compared to the other two algorithms.

Table 6-11. Best & Average Hypervolume Results for the Network A Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.6734	0.6529
MOALCO-GA	0.6813	0.6505
MOPS-GA	**0.7002**	**0.6758**

Figure 6-21 presents the average best hypervolume values from the three algorithms for the Network A problem over the allotted 100,000 fitness evaluations. It is clear from this figure that the three algorithms display similar performance in the first quarter of the search with MOALCO-GA achieving a slight increase in performance over the others during the first 15,000 evaluations. Following 30,000 evaluations is MOPS-GA which splits away from the other two algorithms and starts to achieve better average hypervolume values, whilst both NSGA-II and MOALCO-GA go on to exhibit comparable performance for the remainder of the search, with the former reaching a slightly higher average result.

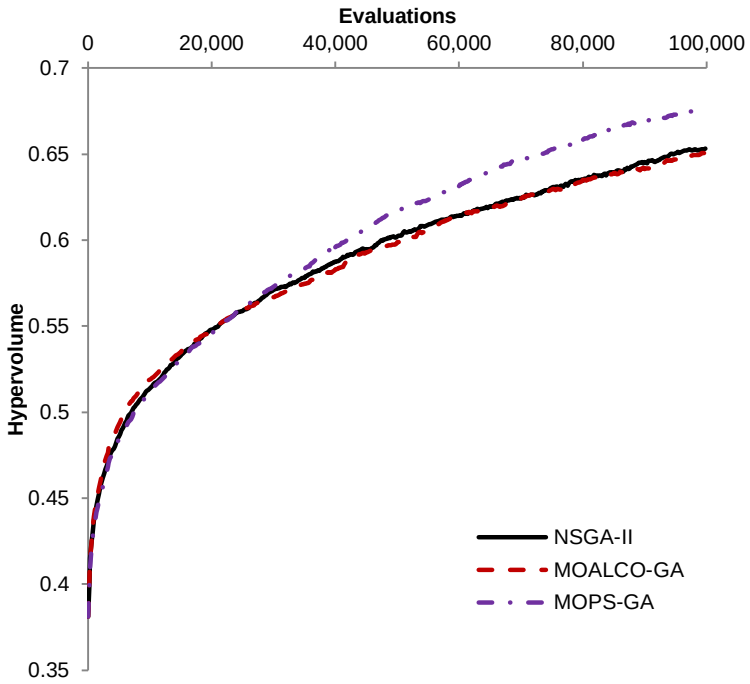


Figure 6-21. Mean Best Hypervolume for the Network A Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-22 presents the best final population from each of the algorithms for the Network A problem. As suggested by the previous figure, it is clear that from these plots that MOPS-GA is able to produce a set of solutions which mostly dominate those generated by the other algorithms. However, it can be seen in the first plot that although MOPS-GA generally produces dominant solutions at lower network cost values, it is NSGA-II and MOALCO-GA that produces the lowest cost solutions at zero hydraulic deficit. Observing the second plot it would seem that as network cost is decreased, the number of pipe smoothing violations increases. It is also apparent that the majority of solution produced by MOPS-GA dominates those found by the other two algorithms in terms of pipe smoothing violations and network cost. It should also be mentioned that as with the Modena problem, in the case of very large networks such as this, it appears to be very difficult for an algorithm to locate smooth network solutions whilst satisfying the other competing objectives in the allotted evaluations. It can be seen from the third plot that as expected MOPS-GA locates the most solutions with the least pipe smoothing violations and produces along with NSGA-II the dominate solution in terms of hydraulic deficit and pipe smoothing violations.

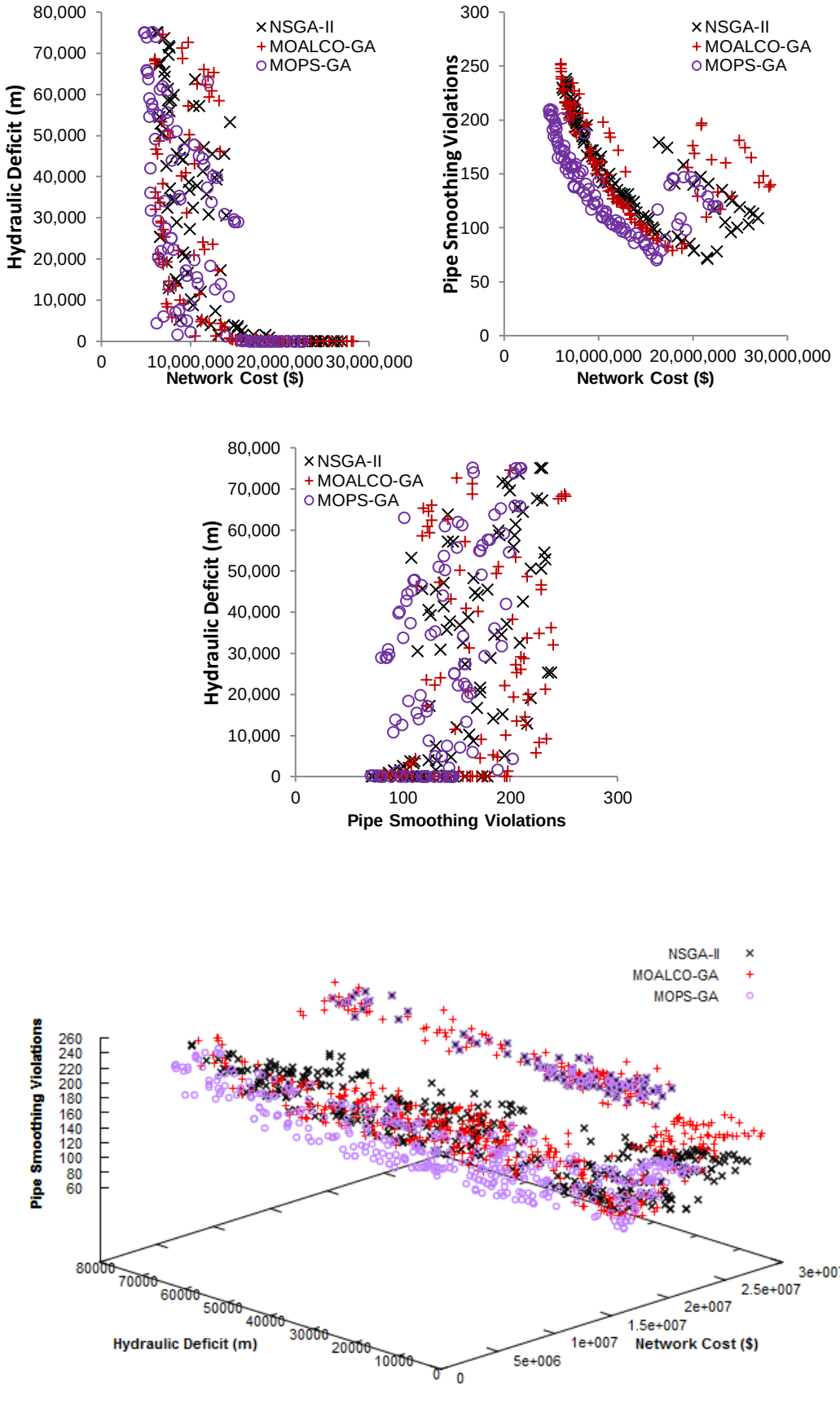


Figure 6-22. Best Final Population for the Network A Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.3.2.6 Network B

Table 6-12 presents the best and average hypervolume results from the three algorithms for the Network B problem. It is apparent that both NSGA-II and MOALCO-GA achieve a similar average population of solutions, reaching comparable hypervolume values. As was the case with the Network A experiment, MOPS-GA displays the highest average performance, obtaining the best hypervolume values out of all the algorithms. No statistical significance in the final population of results was found between NSGA-II and MOALCO-GA; however MOPS-GA produced statistically significant results when compared to the other two algorithms.

Table 6-12. Best & Average Hypervolume Results for the Network B Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Algorithm	Best Hypervolume	Average Hypervolume
NSGA-II	0.5847	0.5687
MOALCO-GA	0.5852	0.5673
MOPS-GA	**0.5895**	**0.5771**

Figure 6-23 shows the average hypervolume of the three algorithms for the Network B problem. Interestingly it is MOALCO-GA that exhibits the best performance in the early stages of the search, only being surpassed by MOPS-GA at 20,000 and NSGA-II at the end of the search. NSGA-II and MOPS-GA display comparable performance during the first 10,000 evaluations, however following this stage MOPS-GA produce higher quality solutions than the standard algorithm for the remainder of the search.

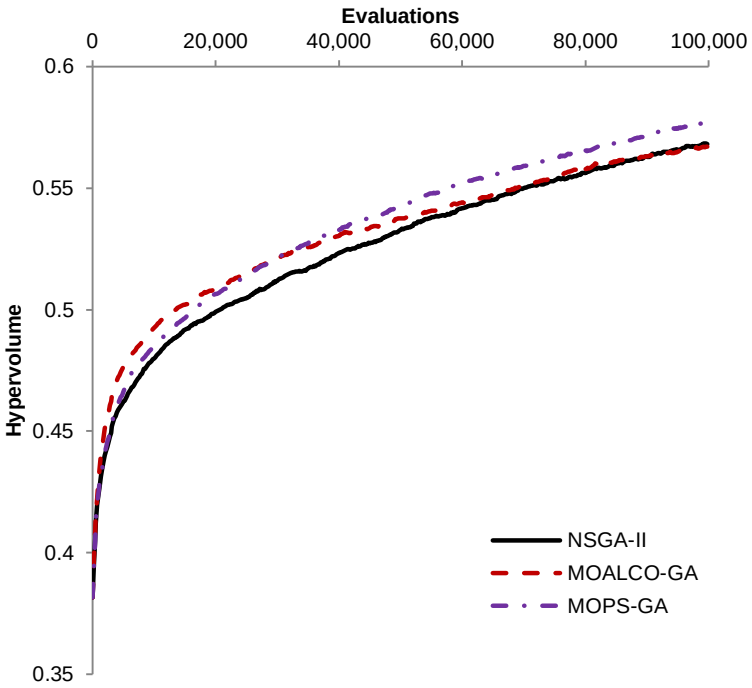


Figure 6-23. Mean Hypervolume for the Network B Problem – NSGA-II, MOALCO-GA & MOPS-GA Comparison

Figure 6-24 shows the best population produced by each of the three algorithms for the Network B problem. It can be observed that the majority of solutions found by MOPS-GA dominate those produced by the other two algorithms in terms of network cost and hydraulic deficit, especially at lower network costs; although NSGA-II does produce some dominant solutions at in the range of 60,000 – 80,000m deficit. It is also apparent that MOALCO-GA tends to find the highest cost solutions, generally located at zero hydraulic deficits. As observed with the Network A tri-objective experiment, as network cost is decreased, the number of pipe smoothing violations tends to increase. It is also apparent that the majority of solution produced by MOPS-GA dominates those found by the other two algorithms in terms of pipe smoothing violations and network cost. Looking at the third plot in the figure shows that MOPS-GA is good at finding the solutions with lower pipe smoothing violations at relatively low deficit values, often dominating those produced by the competing algorithms. Interestingly it is NSGA-II that finds a number of solutions with lower pipe smoothing violations but at the cost of a high hydraulic deficit value.

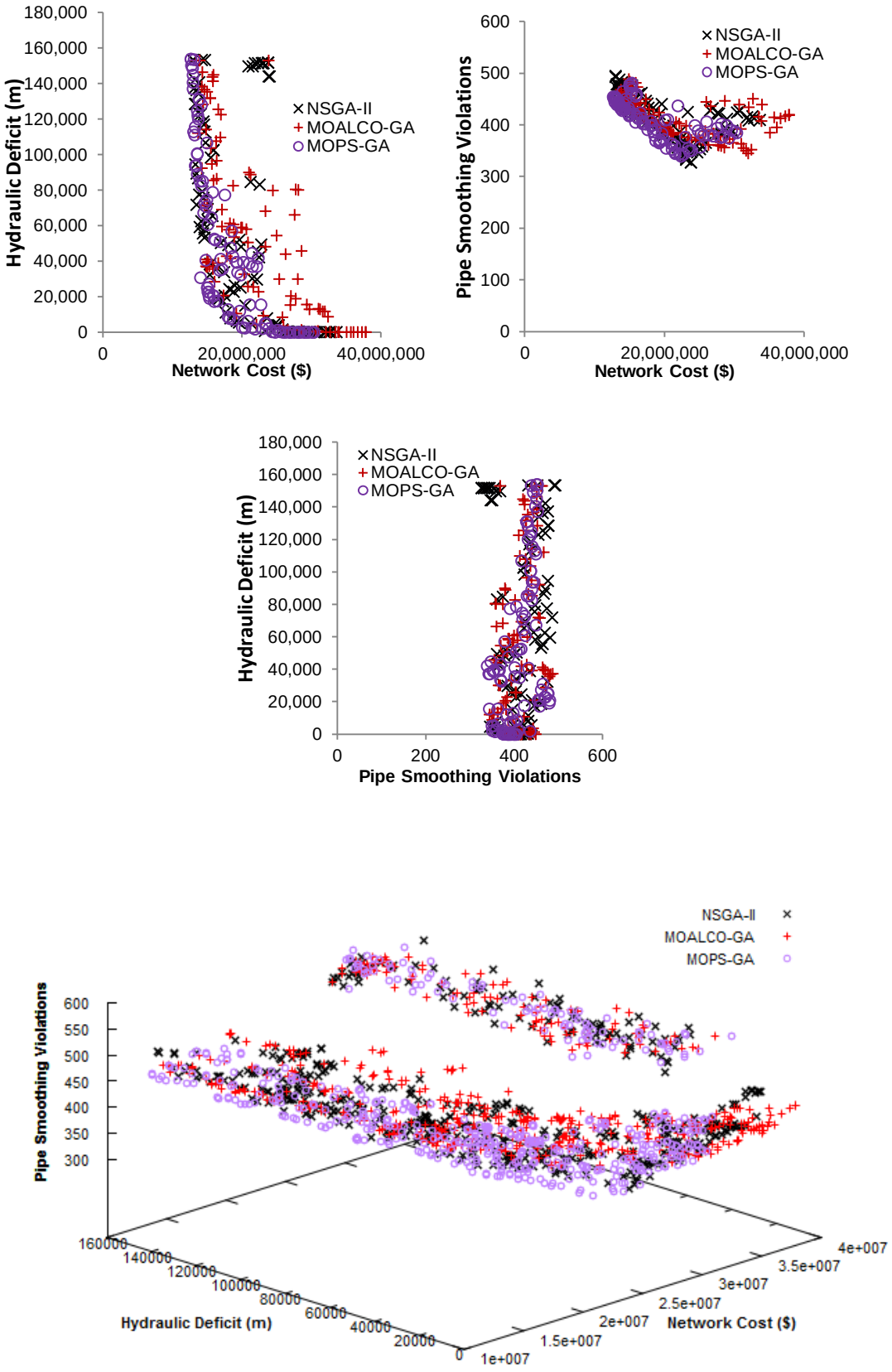


Figure 6-24. Best Final Population for the Network B Problem - NSGA-II, MOALCO-GA & MOPS-GA Comparison

6.4 Conclusion

A Multi-objective Adaptive Locally Constrained Genetic Algorithm (MOALCO-GA) and Multi-objective Pipe Smoothing Genetic Algorithm (MOPS-GA) have been developed and assessed on a number of well-known benchmarks from the literature. Utilising two different heuristics, Both MOALCO-GA and MOPS-GA encode engineering knowledge into the Non-dominating Sorting Genetic Algorithm - II (NSGA-II) with the view to improving the performance of the algorithm utilising the mutation operator.

MOALCO-GA has been shown to perform relatively well from the experiments presented in this chapter when compared to NSGA-II. With regard to the dual-objective experiment set, MOALCO-GA performed well often achieving solutions of equal or higher quality than NSGA-II. The exception to this is in the case of the two large-scale problems, Network A and Network B, although for the larger Network B it outperformed both NSGA-II and MOPS-GA in the first stages of the search. In terms of the tri-objective experimentations, MOALCO-GA was again shown to often outperform NSGA-II in a number of cases and never produced statistically worse results than the standard algorithm.

The influence of the pipe smoothing mutation operator of MOPS-GA has shown to outperform the standard configuration of NSGA-II on all benchmark problems tested in this chapter with the exception of the tri-objective version of the New York Tunnels problem. For the majority of problems tested in this chapter, MOPS-GA displayed faster convergence than NSGA-II and achieved a better set of final solutions. The results also suggest that MOPS-GA performs very well when tackling water distribution network design problem that involve multiple water sources.

The introduction of a pipe-smoothing component into the multi-objective formulation improves performance in both dual and tri-objective formulations. Whilst the modified algorithm might be expected to perform well in the tri-objective case where one of the objectives reflects the heuristic, it is highly interesting that it should perform so much better on the dual-objective problem. This is a key finding as provides some of the first evidence that incorporating

engineering expertise into an algorithm enables it to improve mathematical optimality in multiple objectives.

In conclusion, both of the engineering heuristic based multi-objective algorithms presented in this chapter were found to outperform a tuned version of NSGA-II in the vast majority of cases, with MOPS-GA generally achieving the best solutions out of all of the algorithms put on test.

Chapter 7: Visualisation of Evolutionary Optimisation for Water Distribution Systems

This chapter investigates the use of modern 3D visualisation techniques to enable the interactive analysis of water distribution systems with the aim of providing the engineer with a clear picture of the optimisation problem and thus aid the overall design process. The content of this chapter is based on the work presented in [149][14] and details the development of WDNet3D, an interactive software package for the visualisation and optimisation of water distribution systems. As stated in chapter 3, water distribution systems are complex entities that are difficult to model and optimise as they consist of many interacting components each with a set of considerations to address, hence it is important for the engineer to understand and assess the behaviour of the system to enable its effective design and optimisation. A substantial amount of research has been conducted into the visualisation of objective spaces, however the decision spaces are less often visualised, especially with the view to engaging the end user. This chapter presents a new three-dimensional representation of pipe based water systems and demonstrates a range of innovative methods to convey information to the user resulting in the ability to simultaneously display more useful information than traditional two-dimensional plan view network representations. The interactive visualisation system presented (WDNet3D) not only allows the engineer to visualise the various parameters of a network but also to observe the behaviour and progress of an iterative optimisation method. This chapter contains examples of the combination of the interactive visualisation system and an evolutionary algorithm enabling the user to track and visualise the actions of the algorithm down to an individual pipe diameter change. The visualisation will aggregate changes to the network over an evolutionary algorithm run and 'lift the lid' on the operations of an EA as it is optimising a network. In addition, the method allows the engineer to view other important optimisation-related information such as the extent to which constraints have been violated in the current design. It is proposed that this interactive visualisation system will provide engineers an unprecedented view of the way in which optimisation algorithms interact with a network model and may pave the way for greater interaction between engineer, network and optimiser in the future.

A considerable amount of research has been conducted on the visualisation of objective spaces, both single objectives, multi and many objectives and in the visualisation of decision spaces as characterised by the search procedure. This investigation has naturally been inclined to focus on these spaces in which an evolutionary algorithm operates in an effort to improve performance and understanding of the progress of an evolutionary algorithm (EA). These visualisations often consider the problem statically e.g. in the visualisation of the final solution or Pareto front of an algorithm, or through temporal snapshots where the execution of the algorithm can be traced and the dynamics of the optimisation can be observed. In addition conventional visualisation can either be seen to be focussing on the developer of algorithms to aid in the algorithm's development and refinement, or on the end user as a method to deliver the outcomes of an optimisation to its intended audience and to aid decision support in the real-world. The work presented here intends to provide a visualisation method of the optimisation process itself, providing a user an insight into the behaviour of an EA.

7.1 Water Distribution Network Visualisation

WDNet3D utilises a three dimensional domain to represent pipe based water distribution networks to effectively convey relevant hydraulic information to the user. In addition to presenting hydraulic data, the system can also be used to visualise the progress of an iterative optimisation technique such as a Genetic Algorithm (GA) [2] a popular technique when solving the problem of least-cost water distribution design [12]. This gives the engineer an insight into the actions of the algorithm and enables a greater understanding not only of the optimisation process, but of the aspects of the problem that are more difficult for the algorithm to solve.

When developing WDNet3D it was important to provide the user with an intuitive yet powerful method of navigating the 3D environment, this is especially important when visualising large and complex networks. The software employs a middle mouse button centric control method for viewport manipulation where the user can pan, orbit and dolly the viewport camera to explore the 3D domain. This type of camera navigation is utilised in a number of 3D computer graphics programs and has been shown to be an effective and intuitive control method.

It is important that system information is conveyed to the user in an effective manner, to facilitate this, the key principles of display design presented by Wickens et al. [150] were consulted when designing the geometry that would represent the network elements and variable communication methods. It is accepted that the user's perception and interpretation of signals is based on past experiences therefore the core network elements shown in Figure 7-1 were initially inspired by the classic network component representation utilised in EPANET's 2D graphical display. It was important to keep the elements simple but distinct so that specific components of a complex network could be identified quickly and with ease. As with other WDN mapping software such as EPANET 2, WDNet3D utilises a node & link based representation method where all network element fall into the category of either a node or link.

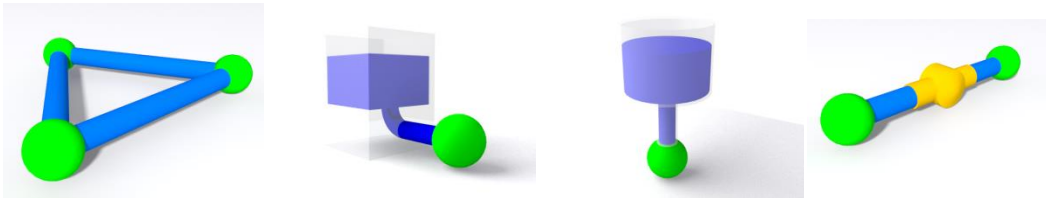


Figure 7-1. 3D visualisation of water distribution network components (left to right – pipes, reservoir, tank and pump)

Hydraulic values and other network parameters can be displayed to the user using a variety of methods, such as element colour, size, texture and movement. For example, when displaying hydraulic pressure at a junction the software utilises the HCI principal of redundancy gain by conveying the pressure not only with colour, but also with element size shown in Figure 7-2, thus reinforcing the user's perception of the data being communicated.

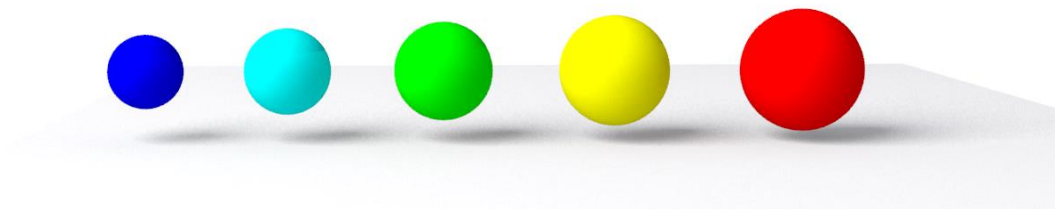


Figure 7-2. Redundancy gain value representation for junctions

The user has the ability to interact with a water system by clicking on any network element within the network. Doing so will display that object's hydraulic and optimisation information and enable the user to manipulate other parameters, for example junction demand, pipe roughness or pump speed. WDNet3D utilises the junction coordinate and elevation data provided in a standard EPANET 2 input file to build and display a network. Pipes are drawn between connecting nodes/junctions and the diameters displayed accordingly. The data is normalised and adjusted to ensure all network elements are visible and correctly displayed. Figure 7-3 shows EPANET example network 2 rendered in WDNet3D.

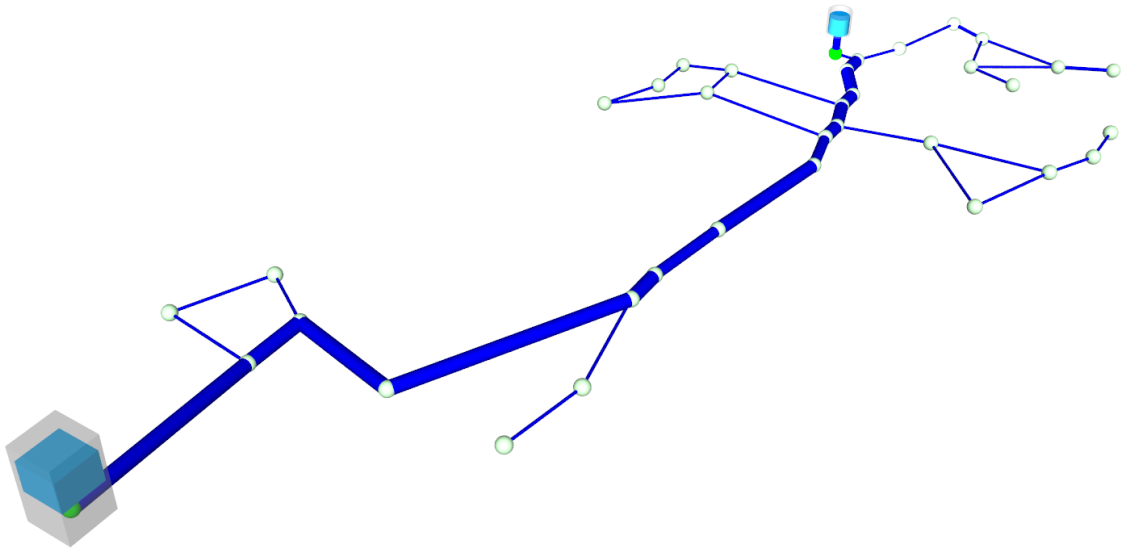


Figure 7-3. WDNet3D visualisation of Network 2

Figure 7-4 shows a relatively rudimentary plan view visualisation of the Hanoi network produced by EPANET 2 which utilises colour coding to show a selected aspect of the network, e.g. diameter, elevation, velocity, pressure etc. Although more sophisticated visualisations exist within commercial packages in the industry, this plan view of the network is typical of the visualisations of water distribution networks.

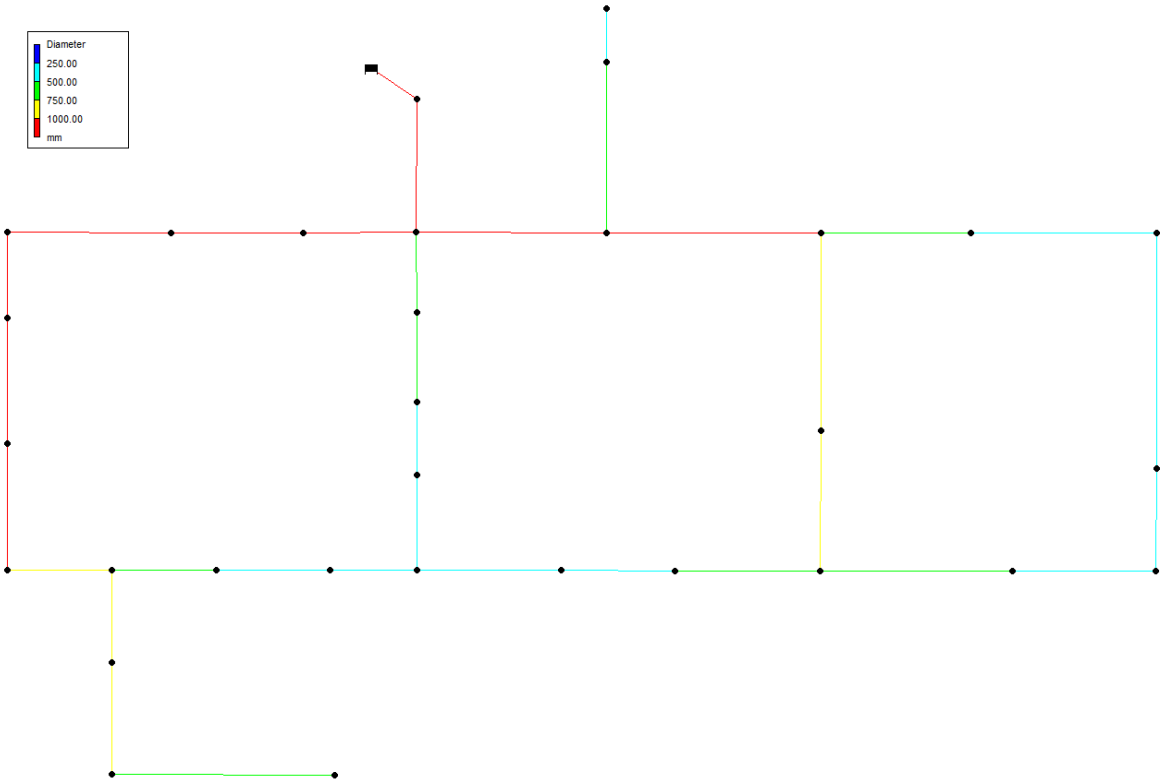


Figure 7-4. Epanet 2 visualisation of the Hanoi network

Unlike in 2D plan view visualisations, the use of a 3 dimensional domain ensures spatial elements such as distances, pipe diameters, lengths and crucially the elevation of elements, are visualized implicitly without recourse to colour coding or other artificial mechanisms. This effect is illustrated in Figure 7-5, a 3D visualization of the Hanoi benchmark network produced by WDNet3D. Although the pipe network grid appears flat, the elevated position of the reservoir can be seen implicitly and is a common arrangement in this type of network known as a ‘gravity fed’ system. Also notable is that the diameters of the pipes can be seen in the visualization. This implicit visualization allows the user to see potential bottlenecks and over-sizing of mains within a network. The colour of the pipes has not yet been used to convey any information and so additional functionality is gained from visualizing velocity, pressure, water quality and other variables within the network. Of course, this is possible whilst also visualizing the implicit aspects of the pipe assets within the system.

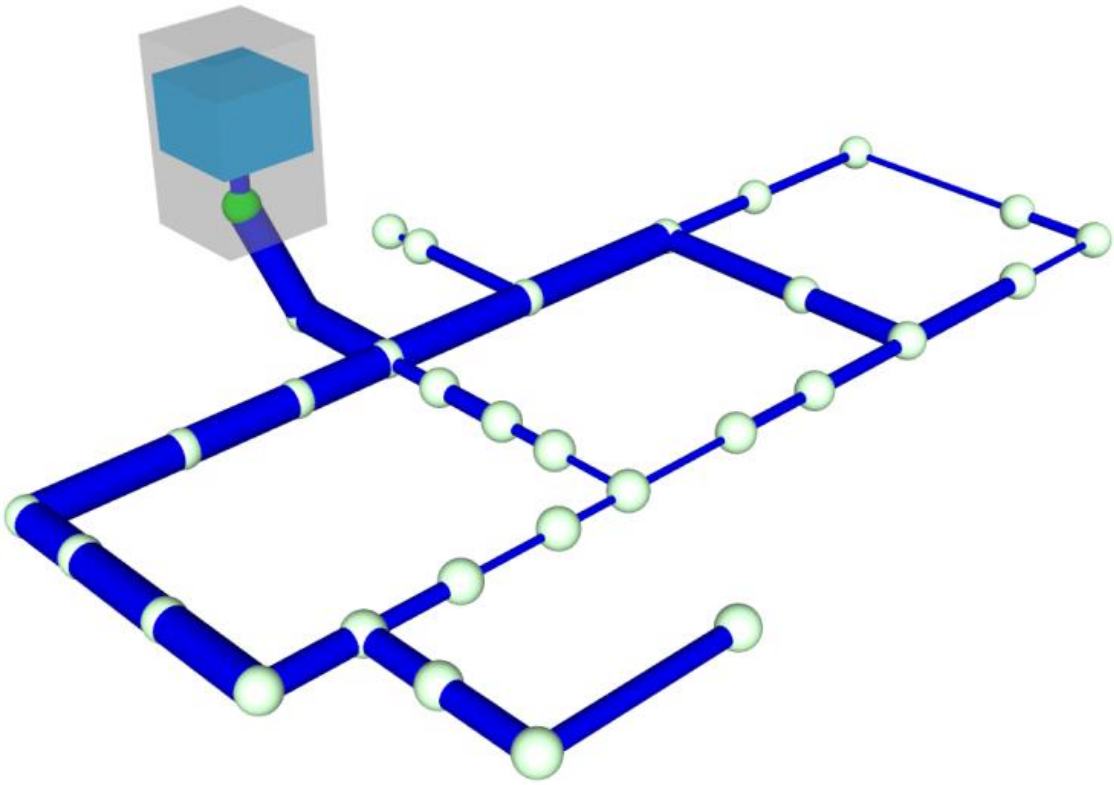


Figure 7-5. WDNet3D visualisation of the Hanoi network

Figure 7-6 shows EXNET [86], a large water distribution network based on a real-world system rendered in plan view in EPANET 2. The diameter of the pipes are represented using a user defined colour scale, although the scale is limited to 5 discreet colours which in turn represent a range of diameters. In the case of visualising large networks with a wide range of pipe diameters such as EXNET the limited resolution of this colour scale can make it difficult for the user to fully grasp the network topology. Utilising a normalised continuous scaled representation of pipe diameter, WDNet3D allows the user to observe the relative differences in pipe diameter in a network to a better degree than one can with EPANET 2.

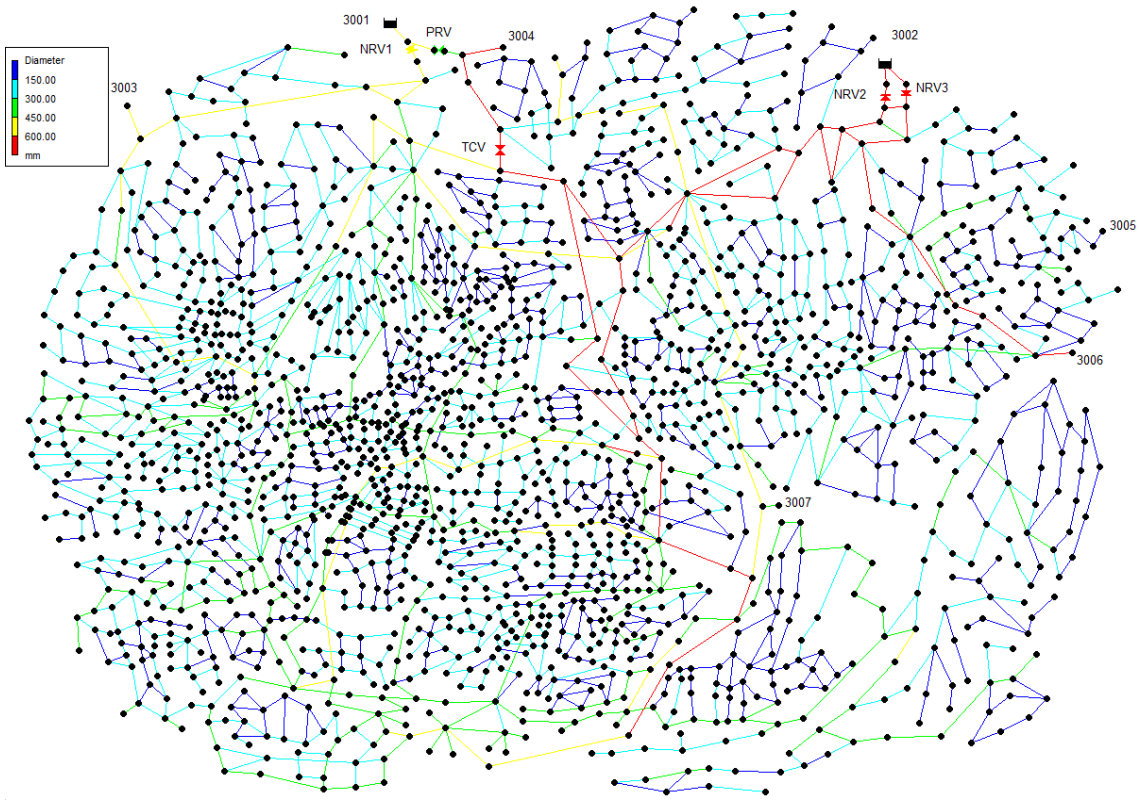


Figure 7-6. Epanet 2 visualisation of the Exnet network

WDNet3D is able to display large networks consisting of thousands of nodes and links and still give the user a good representation of the network topology. Figure 7-7 shows EXNET rendered in WDNet3D. Note it is easy to identify pipe grouping and transmission mains and to get an immediate impression of the topology of the network.

Figure 7-7. WDNet3D visualisation of the Exnet network

WDNet3D utilises the EPANET 2 Programmer's Toolkit for hydraulic and water quality modelling, allowing solutions to be assessed so that relevant data can be passed to the visualisation component of the software. The interactive 3D visualisation module of the application is implemented using Panda3D, a rendering engine which is coded in C++.

7.2 The Visualisation of Water Distribution Network Optimisation

WDNet3D not only allows the engineer to visualise the various parameters of a network but also allows the user to observe the behaviour and progress of an optimisation method, such as a Genetic Algorithm. Combining such an optimisation method with this visualisation system enables the user to track and visualise the actions of the algorithm down to an individual pipe diameter change. To achieve this, WDNet3D aggregates changes made to the network over a GA run and displays them on the network model using a colour scale. It is understood that such a method has not been applied to the design of water distribution networks.

7.2.1 Experimental Setup

By visualizing network construction implicitly, the colour of elements is available to visualize other aspects of the system and this work focuses on this to visualize evolutionary progress. In particular, the colour of pipes is used to represent the number of times that the EA has modified that variable in the best solution for the most recent N generations. This allows the end user to understand the interaction of the EA with the network on a spatial level and to better understand which pipes are requiring most effort from the EA to resolve.

The standard steady state, single objective evolutionary algorithm presented in 3.1 is used to produce solutions for visualisation. It uses a binary representation with each pipe represented by the requisite number of bits for the number of available pipe diameters, e.g. a pipe with 16 possible diameters requires 4 bits for each pipe. This algorithm is popular in the application area and is sufficient to illustrate the proposed visualisation method. In practice, any

EA formulation (including multi-objective formulations), or iterative optimization algorithm for that matter could be used.

The WDNet3D system is applied to a number of benchmark networks from the literature and are as follows:

Hanoi – A single source network consisting of three loops, 34 decision pipes and 6 available pipe diameters with a resultant search space of 2.86 x 10^{26}.

Two Loop - Single reservoir and 8 decision pipes arranged in to twin loop. There are 14 available pipe diameters, resulting in an effective search space size of 1.48 x 10^{9}.

Anytown - 35 existing pipes and 6 possible new pipes with 10 possible pipe diameters. 19 nodes with varying demands. It should be noted that the original version of this problem includes a number of operational aspects that have been fixed in this formulation where only pipe sizing is considered. Possible combinations: ~1.0x10^{41}.

EPANET 2 Example Network 3 - Larger real-world inspired network with 117 pipes and 92 nodes. 16 pipe diameters are used in this formulation. Possible combinations: ~ 7.6x10^{140}.

7.2.2 Results

In each of the following, a network has been visualised with the WDNet3D system at points throughout the optimisation process. In each case, the period of the optimisation can be seen in the left panel highlighted in blue and network, which EA changes to pipe diameters shown in the right panel. The legend (which changes for each ‘snapshot’) shows the relationship between colour and the number of changes made to the best solution in the selected period of optimisation.

7.2.2.1 Hanoi

Figure 7-8 shows the Hanoi network displayed in WDNet3D after a GA optimisation run. The number of diameter changes between the first generation and final best solution during the optimisation process is displayed using a colour key on each pipe, where blue indicates low diameter variance and red high diameter variance.

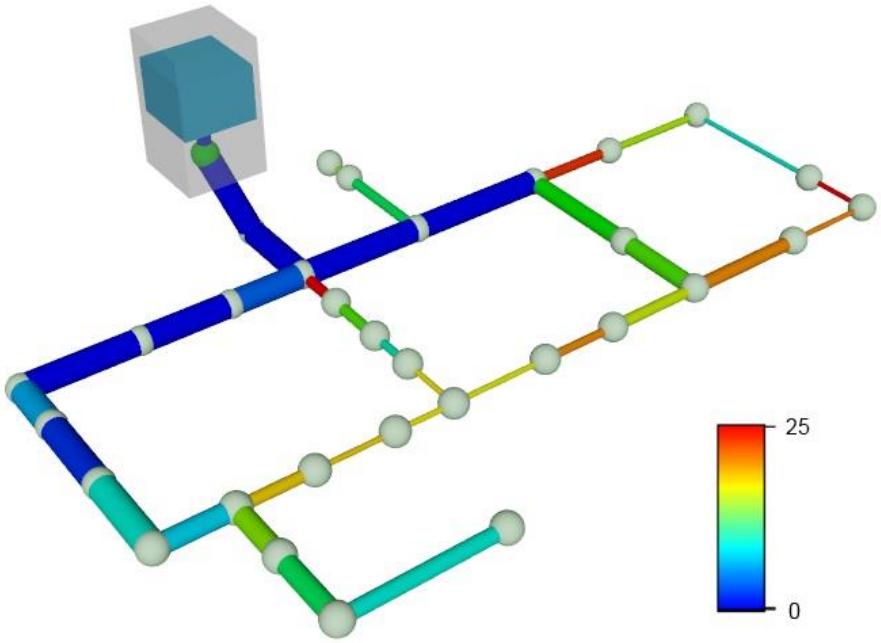


Figure 7-8. Pipe diameter changes for the Hanoi problem

It can be observed that the larger pipes (dark blue) closest to the reservoir have not been frequently altered in the fittest solution during the optimisation process whereas the small diameter pipes (red) towards the extremities of the network have had their diameters varied more frequently. This behaviour can be expected as diameter changes to the pipes near the reservoir would have a more significant effect on the hydraulic performance of the network than changes to pipes further down-stream.

WDNet3D allows the user to pause the GA optimisation at any point in a run and view the algorithm's progress and behaviour. Figure 7-9 to Figure 7-12 show the performance of the GA over 30,000 solution evaluations with the pipe diameter variance displayed at four highlighted periods (light blue). It can be clearly seen from this figure that the algorithm settles on a set of sizes for the 'trunk mains' of this problem early on in the optimisation and that these diameters remain relatively fixed for the remainder of the optimisation. As time progresses, more of the network becomes fixed and the optimisation focuses on making changes to the extremities of the network. It can be observed that from the second period (7,500 evaluations) of the search onward, more than half (17+) of the pipes in the network have reached their final diameter and will not be changed for the remainder of the search. As this is showing the progression of the best individual, it is perhaps not surprising that the solution becomes progressively more fixed as the algorithm converges towards a solution. However, the key is that the spatial distribution of diameter changes can be seen across the network and can help identify areas that are proving difficult for the algorithm to optimise. An additional benefit of this approach is that it suggests that portions of the design that are fixed early on could be removed from the optimisation process and therefore reduce the number of mutations that lead to poorer results in the latter stages of an optimisation.

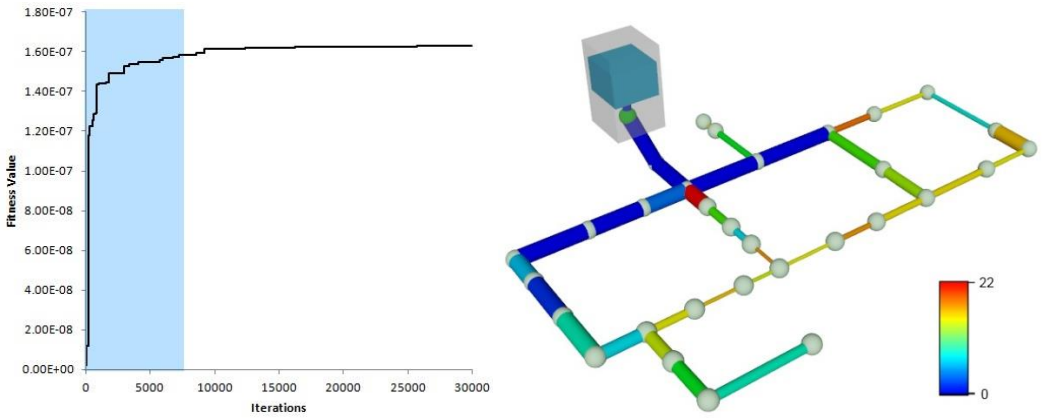


Figure 7-9. Best solution network cost (0 - 7,500) – pipe diameter variations – Hanoi problem

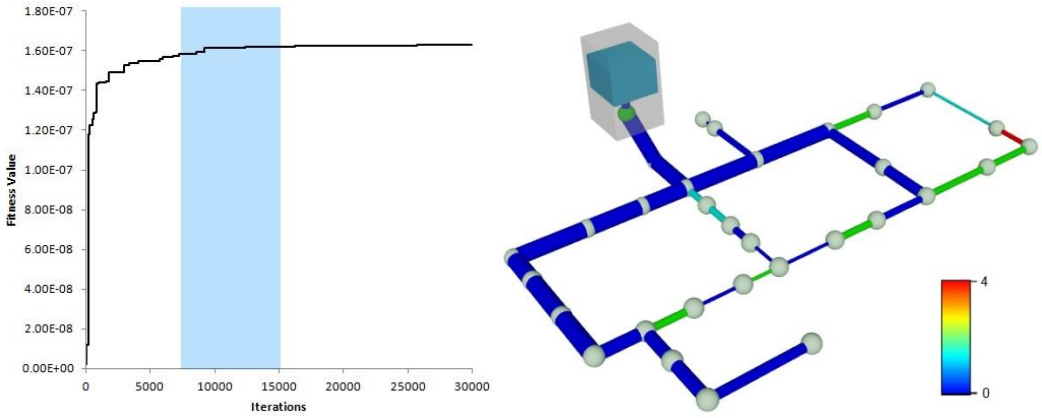


Figure 7-10. Best solution network cost (7,500 – 15,000) – pipe diameter variations – Hanoi problem

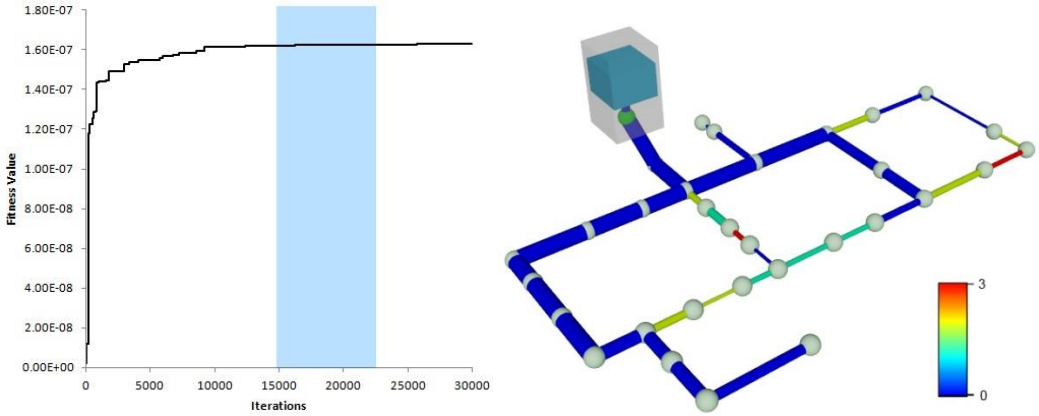


Figure 7-11. Best solution network cost (15,000 – 22,500) – pipe diameter variations – Hanoi problem

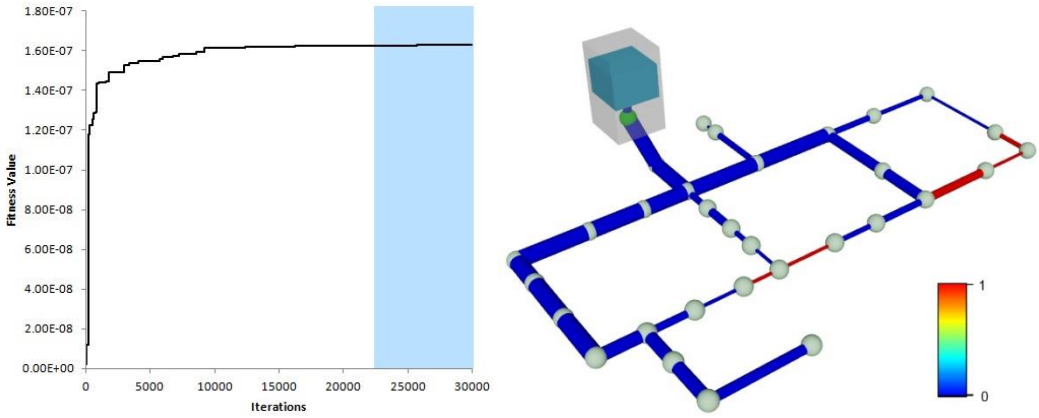


Figure 7-12. Best solution network cost (22,500 – 30,000) – pipe diameter variations – Hanoi problem

7.2.2.2 Two Loop

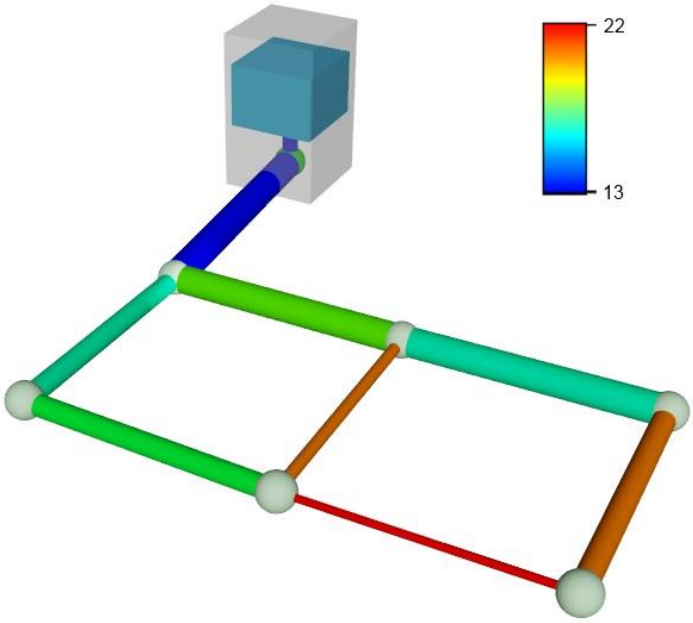


Figure 7-13. Pipe diameter changes for the Two Loop problem

Figure 7-13 shows the Two Loop network displayed in WDNet3D after a GA optimisation run. The number of diameter changes between the first generation and final best solution during the optimisation process is displayed using a colour key on each pipe, where blue indicates low diameter variance and red high diameter variance. It can be observed that the closer a pipe is to the reservoir the less it varies in the best solution throughout the evolutionary process.

Looking at Figure 7-14 to Figure 7-17 is clear that the majority of diameter changes occur at the beginning of the GA's search. What is interesting is that the majority of pipes in this search do not become fixed by the algorithm until after ~34,000 iterations. Looking at Figure 7-16 it is clear that the best solution in the population is replaced with a solution where nearly every variable is different from its predecessor.

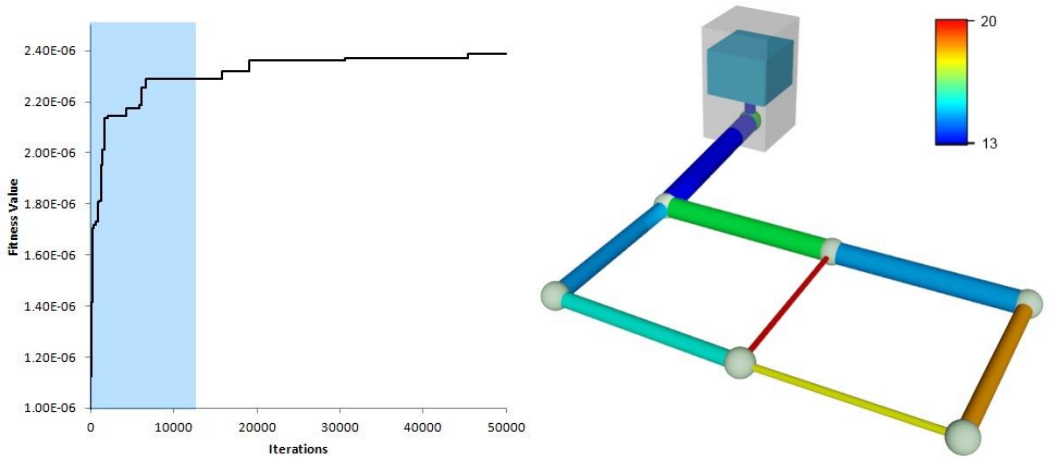


Figure 7-14. Best solution network cost (0 – 12,500) – pipe diameter variations – Two Loop problem

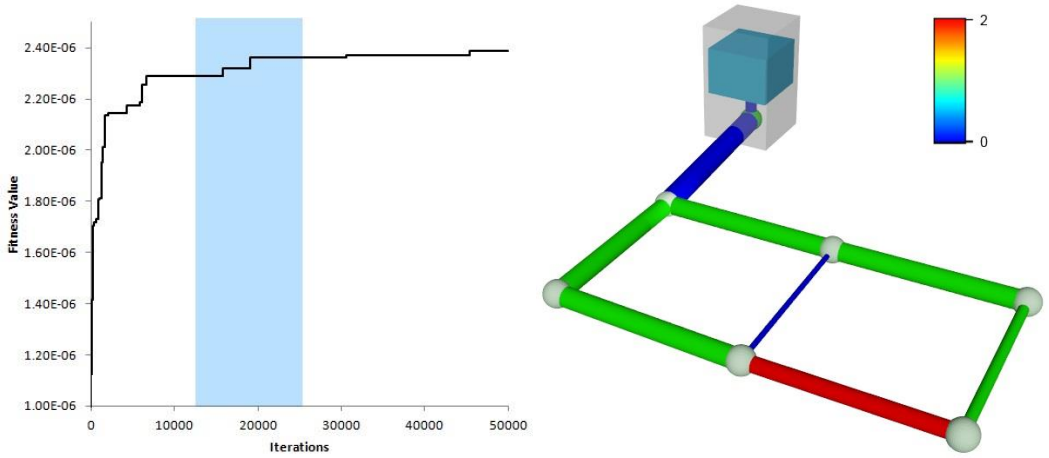


Figure 7-15. Best solution network cost (12,500 – 25,000) – pipe diameter variations – Two Loop problem

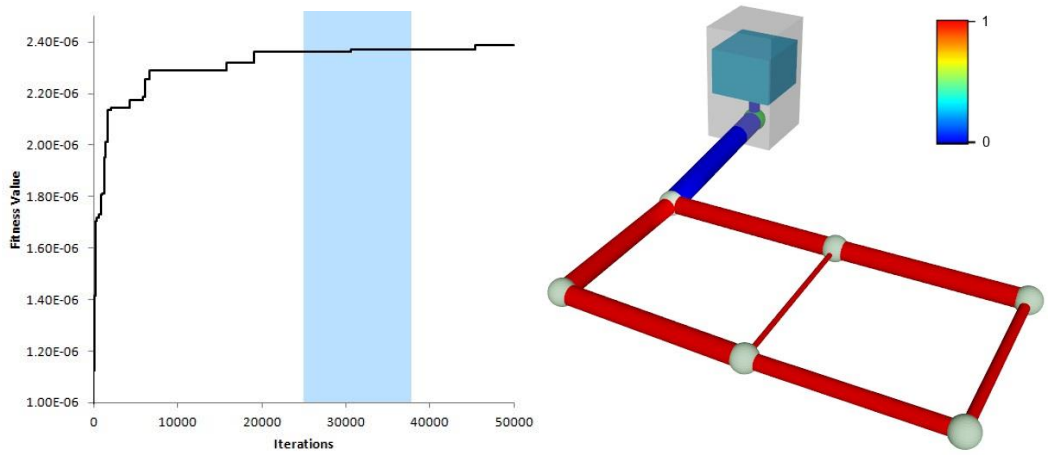


Figure 7-16. Best solution network cost (25,000 – 37,500) – pipe diameter variations – Two Loop problem

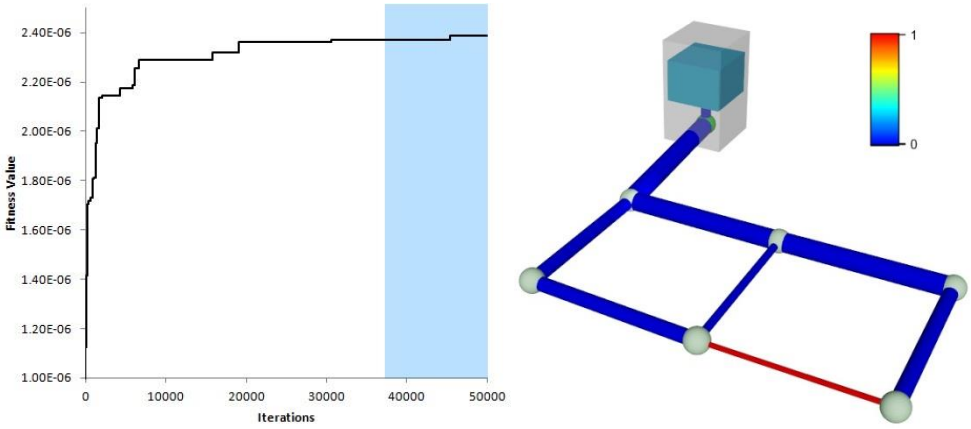


Figure 7-17. Best solution network cost (37,500 – 50,000) – pipe diameter variations – Two Loop problem

7.2.2.3 Anytown

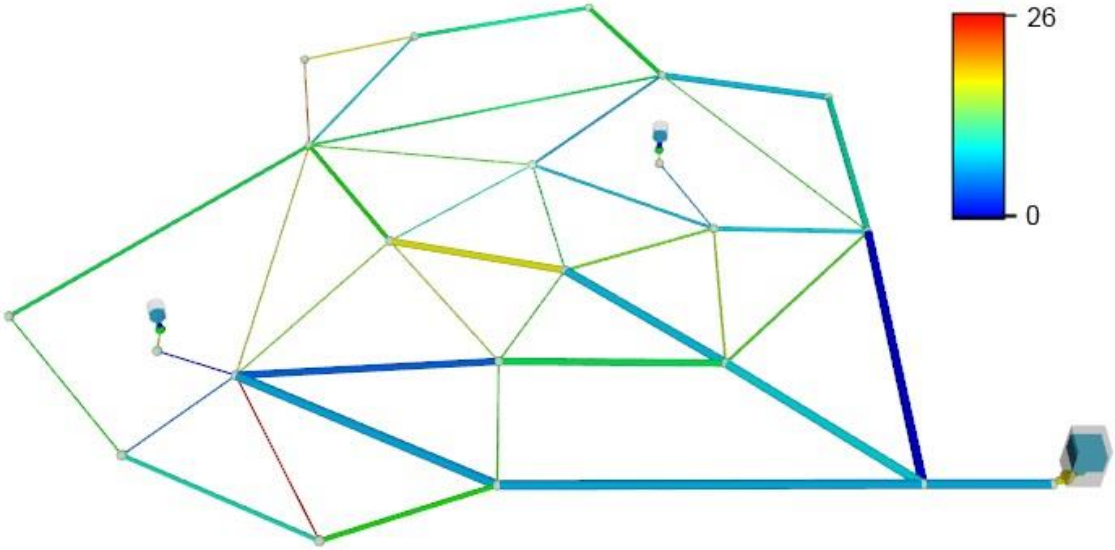


Figure 7-18. Pipe diameter changes for the Anytown problem

From these figures (Figure 7-18 to Figure 7-22) it is clear that Anytown presents a more difficult problem to the GA, with very few pipes achieving 'fixed' status over the first 2000 iterations. From thereon, the story is somewhat similar to the Hanoi example with infrastructure emanating from the main reservoir being progressively fixed along with some close to the tanks. It is clear that the majority of effort is being expanded along the more 'looped' sections of the network where small improvements can be made.

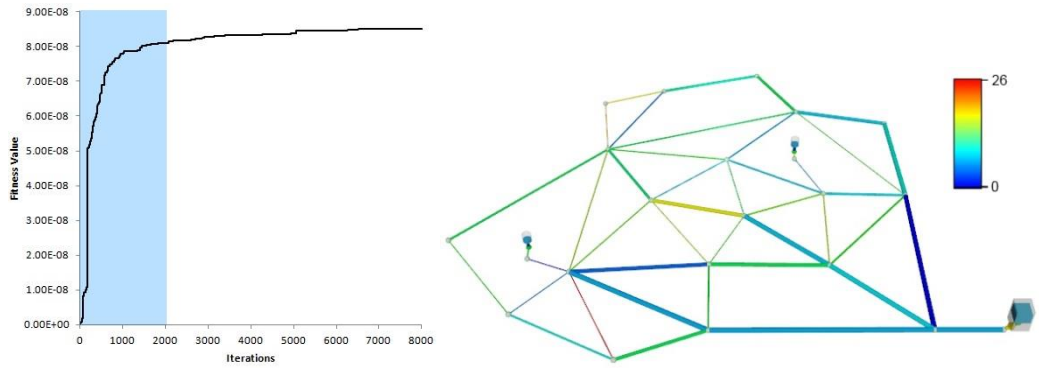


Figure 7-19. Best solution network cost (0 – 2,000) – pipe diameter variations – Anytown problem

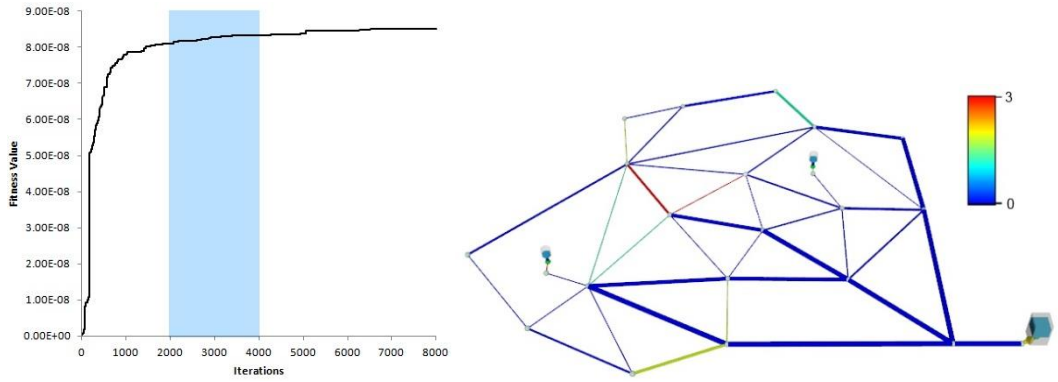


Figure 7-20. Best solution network cost (2,000 – 4,000) – pipe diameter variations – Anytown problem

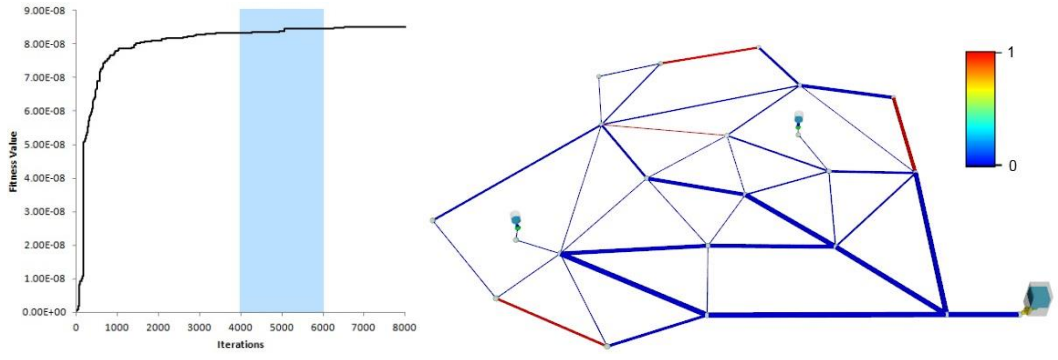


Figure 7-21. Best solution network cost (4,000 – 6,000) – pipe diameter variations – Anytown problem

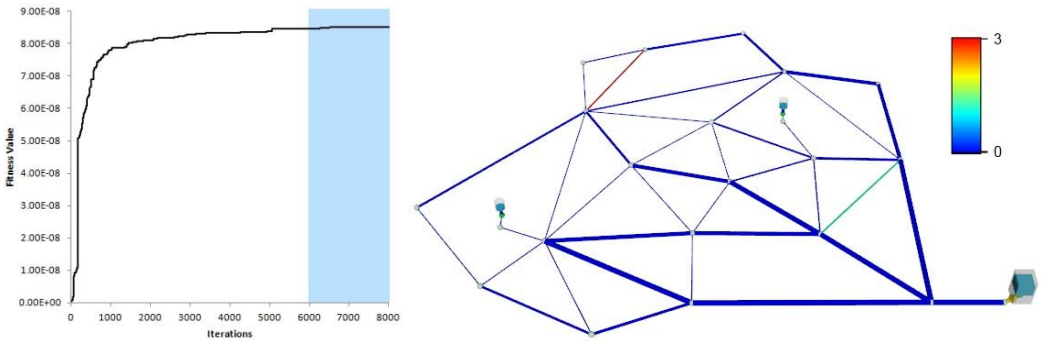


Figure 7-22. Best solution network cost (6,000 – 8,000) – pipe diameter variations – Anytown problem

7.2.2.4 Network 3

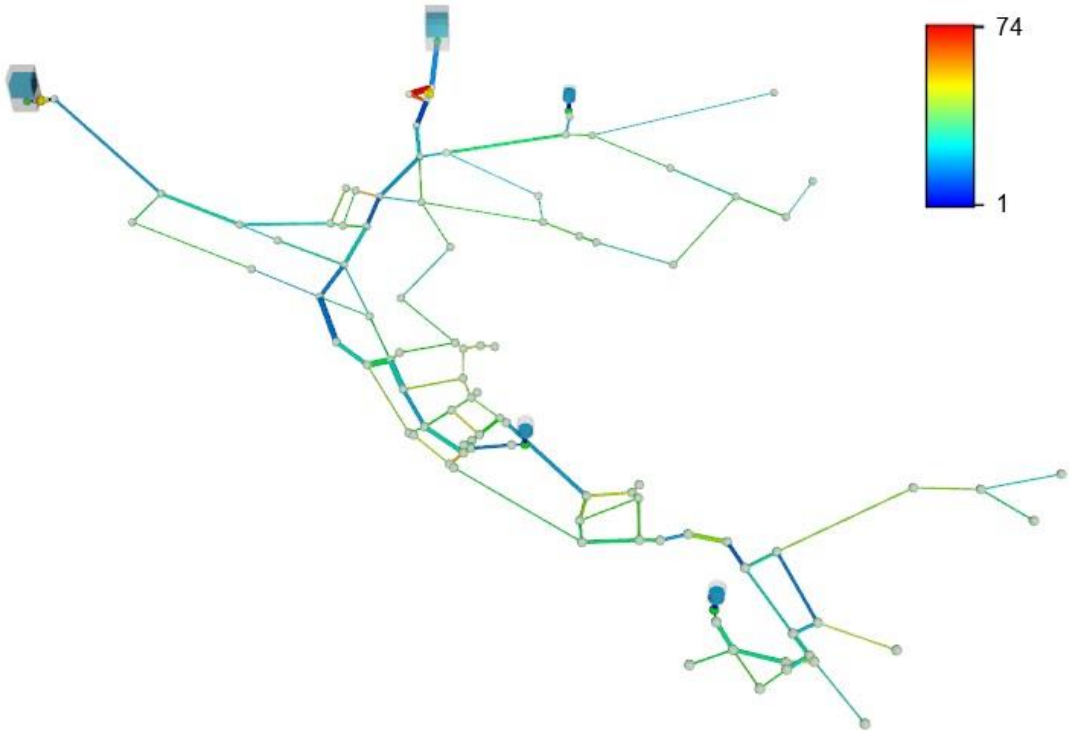


Figure 7-23. Pipe diameter changes for the Network 3 problem

Network 3 is much closer to a real-world style network and has more variables than the other three examples. In this example, the GA first optimizes the important infrastructure between the two reservoirs and tank towards the North of the network. It then proceeds to identify and 'fix' the central 'trunk' mains that link the reservoirs to the North with the rest of the network in the South. By the final snapshot, as with the two previous examples, the algorithm is concentrating on other sections and the majority of refinement appears to be taking place in the smaller mains parallel to the trunk.

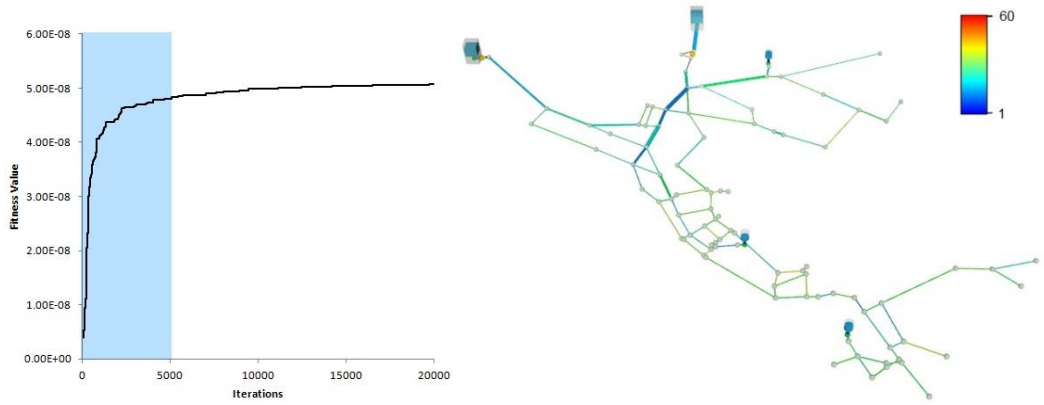


Figure 7-24. Best solution network cost (0 – 5,000) – pipe diameter variations – Network 3 problem

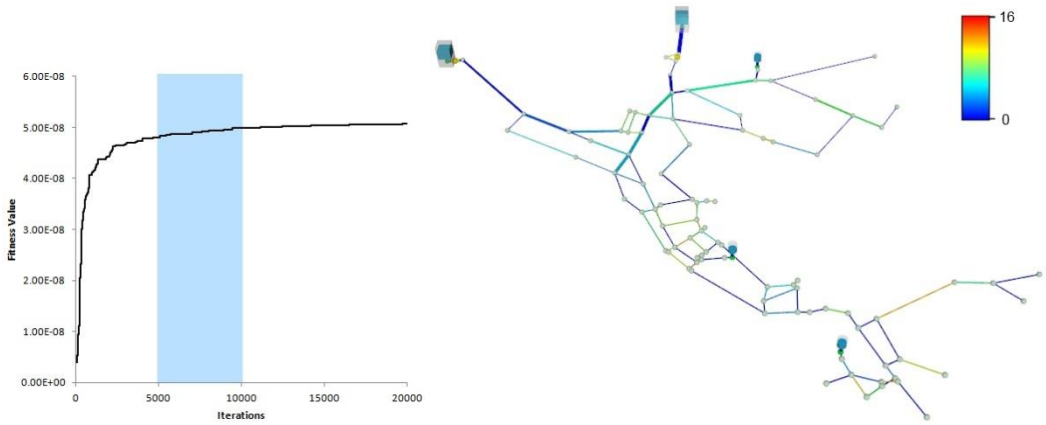


Figure 7-25. Best solution network cost (5,000 – 10,000) – pipe diameter variations – Network 3 problem

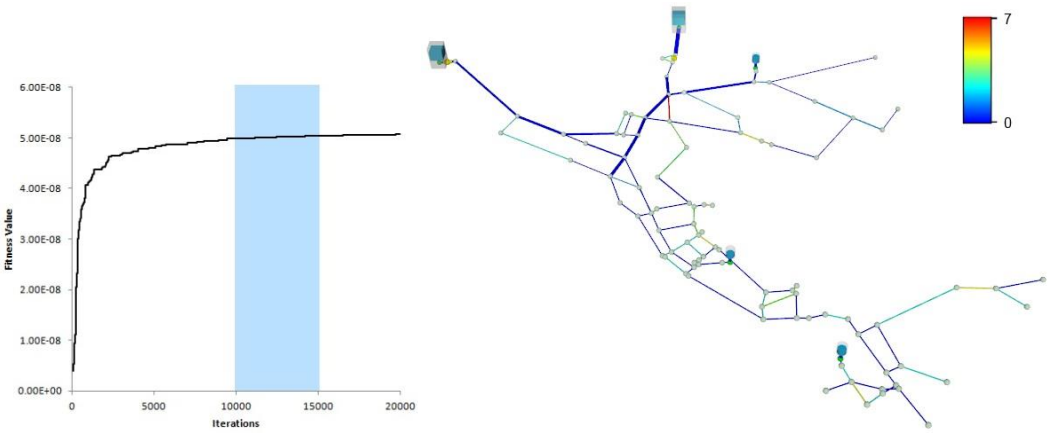


Figure 7-26. Best solution network cost (10,000 – 15,000) – pipe diameter variations – Network 3 problem

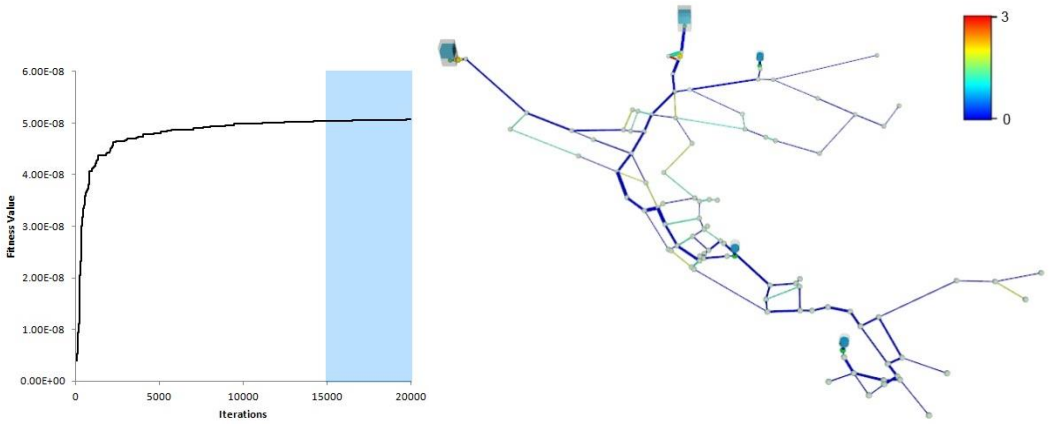


Figure 7-27. Best solution network cost (15,000 – 20,000) – pipe diameter variations – Network 3 problem

7.2.3 Conclusion

The four example networks all show that the evolutionary algorithm tends to size and 'fix' the trunk main infrastructure (effectively the macro-level problem) early on in the optimization. However, it would appear that the extent to which this occurs is dependent on the size of the problem but also its interconnectedness. Despite its relatively modest size, Anytown shows a resistance to this fixing behaviour, that is likely to be due to its lack of a central set of trunk mains and highly looped structure. The larger 'real-world' Network 3 experiences more fixing despite its larger number of assets because it has a more traditional 'trunk main' style layout.

As these changes are made to the best solution in each generation, there may be a case for fixing these parameters during the EA optimization to reduce running times and to allow the algorithm to focus on those areas of the network that require modification. The relatively small number of changes seen in the latter stages of the optimization belie the fact that, through the uniform random mutation operator, many thousands of alternative 'trunk main' configurations will have been tried and discarded.

7.3 Conclusion & Future Work

In this chapter, the development of WDNet3D, an interactive three dimensional program for the visualisation of optimisation for water distribution systems has been presented. By drawing upon the key principals of human-

computer interaction an effective and intuitive tool for the visualisation of water distribution systems has been developed. The system not only allows the engineer to visualise the various parameters of a network but also allows the user to observe the behaviour and progress of an iterative optimisation method. The 3D visualisation and the use of colour allows the system to make good use of the screen 'real estate' and to communicate large amounts of information to the user in a single snapshot. For example, the 3D system means that both diameter and relative elevation are implicitly presented before colour is used to show the various hydraulic parameters. The presentation is also more naturally interpreted by the user than viewing diameters and elevations of links and nodes as colours or floating point values.

In addition, the system has been shown to aggregate changes to the network over a genetic algorithm run and 'lift the lid' on the operations of a genetic algorithm as it is optimising a network. The link between genotype and phenotype spaces in evolutionary algorithm optimisation is often not well explained, particularly for real-world problems. WDNet3D demonstrates that for WDN optimisation, it is possible to gain an understanding of the spatial distribution of algorithm behaviour on this problem.

Users of WDNet3D have reported that the intuitive three dimensional representation of water distribution networks provides more insight into network operation than tools such as EPANET 2. This is primarily due to the tool's ability to display more data on screen than standard two dimensional display methods and is particularly effective on large-scale networks such as Exnet (figure 7-7). Users also report that the ability to view the operations of an optimisation algorithm on the network design itself provides useful information to the designer that could then be employed to refine algorithm operation and efficiency. Some have also suggested that the tool could be adapted to form an educational package to engage water system engineering students in the study of advanced optimisation methodology.

By integrating both state of the art optimisation and visualisation methods, WDNet3D hopes to pave the way for greater interaction between algorithm and network in the future.

7.3.1 WDNet3D Future Development

With further development WDNet3D will not only display the operations of an optimisation method but also enable the algorithm's search to be automatically constrained. From the observations made when visualising pipe diameter variation during an optimisation run it was noted that a large proportion of pipe diameters in a network are finalised in the relatively early stages of the search and as a result a large number of solution evaluations are potentially wasted in the later parts of the search. Therefore if pipe diameters of specific pipes can be fixed hence limiting the search space by reducing the number of decisions, then in theory the optimisation will arrive at an acceptable solution using fewer evaluations.

This could be achieved through direct user interaction with network components, for example, the user would select a pipe and choose to fix a diameter; this could be a single diameter or a smaller range of diameters from the available set. Such an action would apply a hard constraint to the problem through manipulation of the solution representation to ensure no violation of the user imposed rules occurs during the remainder of the optimisation. Another approach would be to incorporate a fitness penalty linked with user imposed constraints that penalise a solution's fitness value dependant on the magnitude of the violation.

Another possible development could be the inclusion of a pattern matching method to build heuristics based on user interaction with network components. These heuristics could then be applied automatically throughout the entire network. For example, if a user fixes the diameter of a pipe connected to a water source the system then searches the network for similar configurations and applies the same diameter restriction. It is envisaged that such a routine could boost user productivity especially on large networks where there are repetitive design configurations. The notion of observing user interactions could be taken further by employing machine learning techniques to build heuristics that could then be applied to future optimisation problems.

7.3.2 Possible Application to Other Problem Domains

The approach applied here to water distribution network optimisation could be implemented in other optimisation problem domains. A number of such application domains are discussed here.

7.3.2.1 Network-Based Problems

It is simple to imagine how this approach could be tailored to other network-based problems such as the travelling salesman problem or gene regulatory network optimisation. However, as these problems require the discovery of routes/sub-networks which represent a subset of the fully connected network, the approach would need to be modified somewhat to this problem type. In this adaptation, edges in the final solution could be coloured according to the number of times they have featured in the best solution discovered by the algorithm over time in a similar way to the method presented here, although it would be possible to colour the whole network search space for these problems to provide more information.

7.3.2.2 Operations Research Problems

Operations research problems typically include problems such as resource allocation, scheduling and routing. In each case, the colouring of the resource being allocated, the event being scheduled or route fragment being selected could be adopted. An example of the former is the bin packing problem, an operations research problem that is not network-based. The application of the approach to this problem is illustrated in Figure 7-28.

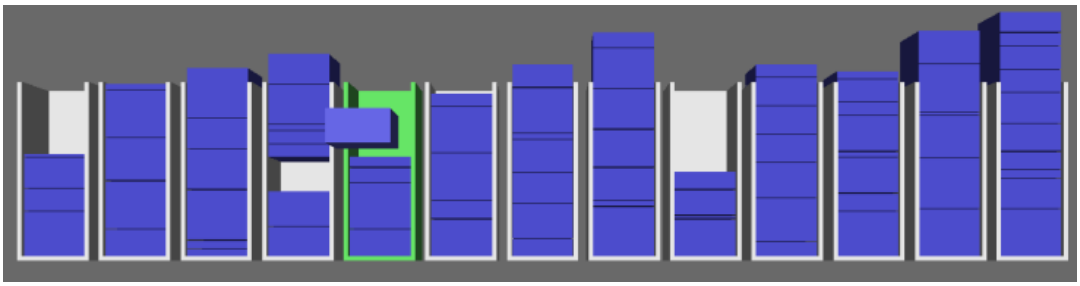


Figure 7-28. Interactive bin packing concept

Figure 7-28 illustrates a (poor) solution to the bin packing problem where items of differing size are being packed into the various bins available. To adapt the proposed approach, each item would be coloured according to the number of times it has moved between bins in the best solution discovered by the algorithm. It is anticipated that this would yield similar results to the water distribution network optimisation problem in that certain items would remain fixed within the solution and the algorithm would then seek to optimise around these fixed blocks.

7.3.2.3 Theoretical and Mathematical Problems

Many EA benchmark problems contain decision variables that are not mapped to a physical construct in the same way that those in the above problems are. In this case, a potential adaptation could visualise the optimisation as a heatmap where rows represent decision variables and columns represent the 'snapshots' of the optimisation through time. The colour of each cell in the heatmap would represent the number of times that each decision variable has changed in the best solution discovered by the algorithm to that point in time.

One further variation to the method, regardless of optimisation domain, and a potential focus for further work, would investigate not simply the number of changes to the network for each variable but also the magnitude and direction of those changes. This revised colouring would then provide more information for all the problem domains thus far described and would make the approach more open to real-valued optimisation problems where the magnitude and direction of change in the best solution variables is more important than simply whether the variable has changed.

Chapter 8: Conclusion

In the field of hydroinformatics the use of expert knowledge to aid in the optimisation of complex water system problems has started to be explored and although prevalent throughout a number of other research fields, the incorporation of knowledge into optimisation techniques such as evolutionary algorithms has only recently began to occur. A review of the literature surrounding the incorporation of knowledge into EAs and the lack thereof of specialist water system mutation operators led to the formulation of the primary research question posed in this thesis:

To what extent does the incorporation of expert water systems knowledge into an evolutionary algorithm mutation operator impact performance of the optimisation of water distribution network problems?

To address the primary research question posed by this work it was first necessary to identify water system based heuristics that engineers utilise during the design of water distribution networks. This work was presented in Chapter 3 along with how these rules could be applied to an evolutionary algorithm through the mutation operator.

The first engineering based heuristic presented in this work was based upon the idea of hydraulic bottleneck elimination, a process which first requires the identification of a pipe or pipes that are restricting flow and preventing adequate pressure in downstream sections of the network. The other engineering heuristic is based upon the idea that the diameter of any pipe is never greater than the sum of the diameter(s) of the directly upstream pipes and the network can be seen to 'smoothly' transition from large to small diameters from source to extremities. A method for integrating these heuristics into a Genetic Algorithm (GA) was then formulated. It was important from the outset that the heuristic should not incur any additional full of partial fitness evaluations which would increase runtime, especially when tackling large scale real-world problems. This was achieved by modifying the mutation operator of the GA so that the engineering heuristics could influence the mutation of the solutions in an efficient manner whilst retaining a good amount of variance in the solutions produced.

Chapter 4 presents a set of extensive experiments designed to identify the optimal parameter values for the Steady State Genetic Algorithm (SSGA) (Chapter 3) and the development of two new algorithms based on the engineering heuristics identified in chapter 3. Various parameters of the SSGA were tuned to provide not only a good platform upon which the new mutation operators could be embedded but also a means to facilitate the fair comparison between the new algorithms and the SSGA. It was found that the variation of static penalty costs has little to no significant effect on the performance of the SSGA except for very low penalty costs on some problems. A correlation was found between problem complexity and tournament size; the larger the problem the greater the optimal tournament size. This behaviour was somewhat expected as a greater selection pressure aids the convergence of the SSGA in large solution spaces. The third set of experiments investigated the effect mutation rate has on the performance of the algorithm and found that as the problem complexity grows the optimal mutation rate generally falls, although it is very difficult to predict an optimal mutation rate based purely on problem specifications alone. It was also observed that there is a statistically significant interaction between the tournament size and mutation rate parameters for all test networks which implies mutation rate and algorithm performance is dependent on tournament size. The final set of parameter tuning experiments focused on the replacement strategy of the SSGA. It was found that the algorithm performed best when the worst individual in the population was selected for replacement. It was also observed that the use of conditional replacement worked best for the smaller problems from the problem set and an unconditional replacement approach performed best for the larger problems. This work was not only useful in ensuring that the SSGA was performing at peak effectiveness but also offers an insight into the effect a number of parameters have on the performance of an SSGA when applied to a large number of benchmark problems from the water systems literature.

The two new algorithms presented in Chapter 4, the Adaptive Locally Constrained Genetic Algorithm (ALCO-GA) and the Pipe Smoothing Genetic Algorithm (PSGA) were then tested on a large range of WDN problems in Chapter 5. Both algorithms were found to display increased performance in the early stages of the search compared with the SSGA and commonly went on to

produce competitive final solutions, often better than the highly tuned SSGA. The results of these experiments lend weight to the argument that encoding an algorithm's mutation operator with water systems knowledge can improve the algorithm's ability to effectively search the solution space; often improving the time it takes for a GA to achieve good performing feasible solutions. Chapter 5 also included a novel study into parameter robustness; the sensitivity of the algorithms to their parameter configuration. This set of experiments was designed to address the second research question posed in this thesis:

How does the inclusion of water systems knowledge impact the sensitivity of an evolutionary algorithm to parameter variance?

Through extensive experimentation it was found that the performance of both ALCO-GA and PSGA were less sensitive to parameter change than the SSGA meaning that non-experts in the field of meta-heuristics will potentially be able to get much better performance out of the engineering heuristic based algorithms without the need for specialist evolutionary algorithm knowledge.

Chapter 6 presented the development and testing of the Multi-objective Adaptive Locally Constrained Genetic Algorithm (MOALCO-GA) and Multi-objective Pipe Smoothing Genetic Algorithm (MOPS-GA); multi-objective variants of the ALCO-GA and PSGA based upon NSGA-II. It was necessary to test whether the engineering based heuristic mutation operators could be applied to a multi-objective evolutionary algorithm and explore how the modifications affected algorithm performance in multi-objective search spaces. To enable this the new algorithms were applied to a variety of multi-objective water distribution network design problems from the literature along with NSGA-II which had been tuned prior to the experiments to provide a competitive platform for comparison purposes. The chapter was split into two main sections; dual-objective and tri-objective problems. The dual-objective set of problems focused on the network cost and hydraulic head performance of solutions, whilst the tri-objective introduced a novel third objective of network smoothness which is based on the engineering inspired pipe smoothing heuristic. In the majority of problems MOALCO-GA displayed the ability to produce a solution set comparative or better than that produced by NSGA-II and often displayed increased performance in the early stages of the search. However, it was

MOPS-GA that significantly outperformed both MOALCO-GA and NSGA-II in the vast majority of multi-objective problems tested. The pipe smoothing heuristic based algorithm performed particularly well on multi-source and large-scale problems based on real-world networks. In summary, the engineering heuristic based multi-objective genetic algorithms presented in this work were found to outperform NSGA-II in the vast majority of cases, with MOPS-GA generally achieving the best solutions out of all of the algorithms put on test.

Chapter 7 presented a novel system to enable the interactive 3D visualisation of optimisation for water distribution networks and set out to address the final research question posed in this thesis:

Can state of the art visualisation techniques provide engineers with a greater insight into the operation and behaviour of an EA when applied to the problem of water distribution network design?

To develop the three dimensional water distribution network visualisation tool (WDNet3D) it was first necessary to develop 3D representations of various water distribution network components so that they would convey as much relevant information to the user as possible whilst remaining intuitive to a water systems engineer. This was achieved using principals from the field of human computer interaction, display design practices and inspiration gained from traditional 2D water system visualisation. Following the 3D geometry concept and creation an intuitive control scheme was formulated and implemented to enable the user to manipulate the 3D space and traverse any network visualisation quickly and effectively. Utilising the EPANET 2 hydraulic simulation engine WDNet3D is able to run complex hydraulic simulations and visualise the resultant data directly on the 3D network model in a manner such that an engineer is able to quickly assess the water system and easily identify any issues with the network. The final feature of WDNet3D enables the user to observe the behaviour and progress of an iterative optimisation method. It does this by aggregating changes made to the network over an optimisation algorithm run and displays them on the network model using a colour scale. This system was demonstrated on a number of water distribution network design problems from the literature using a Genetic Algorithm (GA). It was shown that WDNet3D is able to effectively ‘lift the lid’ on the operations of a

genetic algorithm as it is optimising a network, demonstrating it is possible to gain an understanding of the spatial distribution of algorithm behaviour on water distribution network design problems.

8.1 Conclusions and Future Research

The research presented in this thesis demonstrates the effectiveness of integrating water systems engineering expertise into evolutionary based optimisation methods. Not only is solution quality improved over standard methods utilising these new heuristic techniques, but the potential for greater interaction between engineer, problem and optimiser has been established.

The incorporation of water systems knowledge into the mutation operator of an EA has been demonstrated to improve both algorithm performance and sensitivity to parameter variance when applied to a range of water distribution network design problems. The work presented here establishes a foundation of knowledge to enable the further exploration of knowledge exploitation in the field of hydroinformatics. The engineering inspired algorithms presented in this work applied a single water systems heuristic in each, focusing on one area of WDN design, however there is scope to combine heuristics. For example, the bottleneck eliminating heuristic in ALCO-GA which tends to have a greater impact in the early stages of the search could be used initially, gradually giving way to the pipe smoothing heuristic in the later stages. Another avenue of exploration would be to employ the engineering heuristics to generate the initial population of solutions in an EA. Although knowledge based initialisation has been used in this manner within the field it would be interesting to observe the impact that starting in a more favourable location in the solution space would have on the performance of the special mutation operators. Another idea is to integrate engineering knowledge into the selection operator of an EA, this could be achieved, for example, by applying metrics such as network smoothness as a measure of solution quality to promote engineering feasibility.

The engineering inspired algorithms presented in this work were tested on a large set of gravity-fed WDN design problems from the literature, presenting a very challenging combinatorial optimisation problem. However, a large proportion of real-world WDNs are not gravity-fed and require pumping to

provide the network with sufficient water. Pumped networks present additional complexity, introducing further components for consideration including pumps, valves and storage tanks. These types of problem commonly introduce further objectives and constraints to the problem including, for example, pump energy consumption and tank level restrictions. In addition, these problems generally require the designer to take into account the varying demand over an extended time period and consider emergency scenarios such as power outage. With increased network complexity there would be scope to either adapt the heuristics presented in this work or develop new heuristic methods to aid with the inclusion of an extended set of network components (pumps, tanks etc.). In the case of extended period simulation problems both the ALCO-GA and PSGA heuristics could be applied with minimal alteration, however an issue such as pipe flow direction shift during a simulation poses a potential problem that would need to be addressed as both heuristics depend on pipe flow direction information. To address a case of flow reversal during the extended simulation it would be necessary to aggregate the flow rate vectors of a pipe to ascertain the dominant flow direction in order for the pipe smoothing and bottleneck heuristics to function correctly. With regard to development of further engineering heuristics for more complex problem, metrics such as tank level, tank state (filling or discharging) and pumping patterns could be used to guide the methods, for example it is seen as beneficial if tanks are filled during times of low demand and when the electricity used for pumping is cheaper. A set of heuristics could be built from this notion to promote pumping at off-peak times whilst discouraging the filling of tanks during times of peak demand.

This research has demonstrated that the incorporation of water systems knowledge to an EA not only leads to improvements in computational efficiency and mathematical optimality, but also the generation of solutions industry engineers would find more intuitive (i.e. smooth pipe diameter transitions). The novel WDNet3D visualisation tool presented in this work addresses the need to bridge the gap between engineer and optimisation algorithm with the view of not only giving the engineer a greater insight into the behaviour of an EA but also to facilitate direct interaction with the optimisation process to produce solutions that better meet the requirements of the engineer. Although limited, the interactive visualisation system has demonstrated potential; however continued

research into this area is needed to further close the gap between engineer and optimiser. Future work could include the refinement of the WDN visualisation engine to enable the visualisation of more interactions between algorithm and problem, including constraint violations, objectives and decisions. Another potential expansion to the tool would be the development of a system to allow user interactions to guide the optimisation process. This could include the direct interaction with the optimisation algorithm by allowing the user to impose constraints to a problem such as reducing the number of available diameters for a specific pipe or by imposing additional penalty costs to the objective function when solutions violate user defined constraints. In addition, pattern matching and machine learning techniques could in theory be used to identify and develop new heuristics based on the engineer's interaction with solutions produced by an EA.

Collectively, these developments open up the potential for greater interaction between the human expert and evolutionary optimiser. The heuristics and visualisation methods presented here point to new human-machine interfaces for optimisation that can improve the subjective and objective optimality of results. The use of heuristics guides the search towards promising areas and reduces the number of human perceived errors, whilst the visualisation provides a more intuitive connection between the engineer and the optimisation algorithm, lifting the lid on the black box of the evolutionary algorithm. With further development, the techniques presented in this work could be utilised by practising engineers to aid in the efficient and effective design of commercial water distribution systems. It is also envisaged that these methods for integrating knowledge into an EA could be applied to a much larger and diverse set of problems well beyond the field of engineering.

Chapter 9: Appendix i

9.1 Steady State Genetic Algorithm Parameter Tuning

9.1.1 Penalty, Tournament Size & Mutation Rate

9.1.1.1 Two Loop

Table 9-1. Mean penalty cost results for the Two Loop benchmark problem

Mean Penalty Cost							
Penalty	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
$ 10,000.00	2.38592E-06	$ 419,125.00	2.3505E-06	$ 425,684.06	$ 6,174.02	-	34%
$ 20,000.00	2.38583E-06	$ 419,140.63	2.34461E-06	$ 426,785.31	$ 7,849.49	-	30%
$ 30,000.00	2.38513E-06	$ 419,265.63	2.33822E-06	$ 427,979.69	$ 8,367.28	-	28%
$ 40,000.00	2.38382E-06	$ 419,500.00	2.33415E-06	$ 428,803.75	$ 9,206.02	-	26%
$ 50,000.00	2.38469E-06	$ 419,343.75	2.3345E-06	$ 428,693.75	$ 8,930.08	-	25%
$ 60,000.00	2.38416E-06	$ 419,437.50	2.3316E-06	$ 429,260.63	$ 9,498.21	-	24%
$ 70,000.00	2.38531E-06	$ 419,234.38	2.33125E-06	$ 429,320.31	$ 9,601.08	-	24%
$ 80,000.00	2.38583E-06	$ 419,140.63	2.32885E-06	$ 429,804.38	$ 9,912.81	-	24%

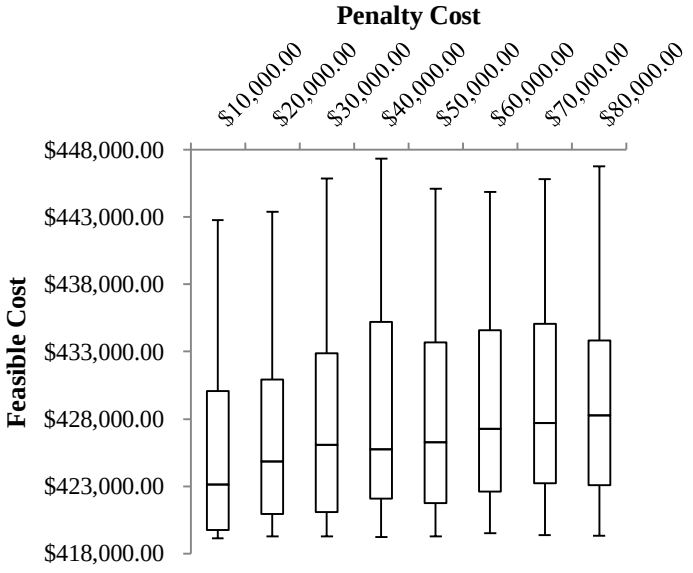


Figure 9-1. Box and whisker plot showing mean penalty cost results for the Two Loop benchmark problem

Table 9-2. Mean tournament size results for the Two Loop benchmark problem

Mean Tournament Size							
Tournament Size	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
2%	2.38408E-06	$ 419,453.13	2.33936E-06	$ 427,767.81	$ 6,540.02	-	32%
3%	2.38338E-06	$ 419,578.13	2.3391E-06	$ 427,815.31	$ 7,385.26	-	30%
4%	2.38539E-06	$ 419,218.75	2.33898E-06	$ 427,831.56	$ 7,912.93	-	28%
5%	2.38592E-06	$ 419,125.00	2.33871E-06	$ 427,889.06	$ 8,176.36	-	27%
6%	2.38478E-06	$ 419,328.13	2.33804E-06	$ 428,027.81	$ 8,672.93	-	27%
7%	2.38583E-06	$ 419,140.63	2.33334E-06	$ 428,980.00	$ 10,396.90	-	23%
8%	2.38583E-06	$ 419,140.63	2.33334E-06	$ 428,980.00	$ 10,396.90	-	23%
9%	2.38548E-06	$ 419,203.13	2.3328E-06	$ 429,040.31	$ 10,057.72	-	23%

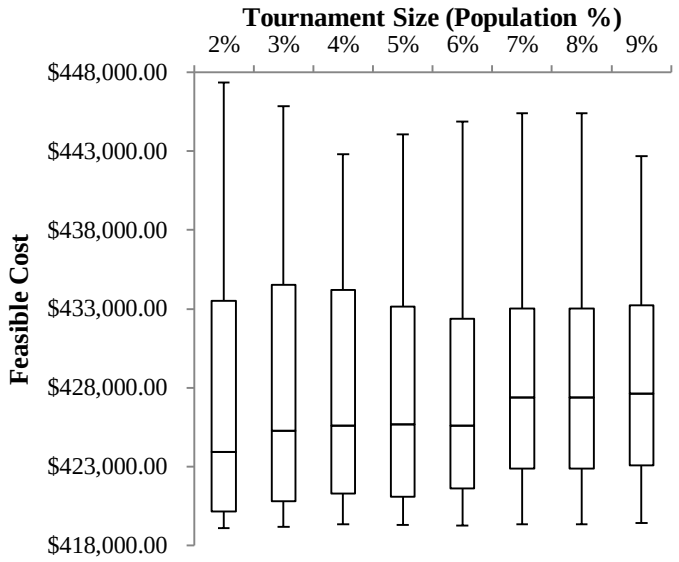


Figure 9-2. Box and whisker plot showing mean tournament size results for the Two Loop benchmark problem

Table 9-3. Mean mutation probability results for the Two Loop benchmark problem

Mean Mutation Rate							
Mutation Rate	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
0.032	2.38654E-06	$ 419,015.63	2.30485E-06	$ 434,559.69	$ 16,474.75	-	17%
0.064	2.38663E-06	$ 419,000.00	2.34573E-06	$ 426,621.88	$ 11,047.84	-	33%
0.096	2.38663E-06	$ 419,000.00	2.36728E-06	$ 422,583.13	$ 7,101.62	-	53%
0.128	2.38663E-06	$ 419,000.00	2.37592E-06	$ 420,972.50	$ 4,884.43	-	58%
0.16	2.38663E-06	$ 419,000.00	2.3745E-06	$ 421,194.06	$ 4,314.02	-	37%
0.192	2.38654E-06	$ 419,015.63	2.3492E-06	$ 425,792.19	$ 6,619.54	-	13%
0.224	2.38451E-06	$ 419,375.00	2.30902E-06	$ 433,284.38	$ 8,945.29	-	3%
0.256	2.37658E-06	$ 420,781.25	2.26718E-06	$ 441,324.06	$ 10,151.50	-	1%

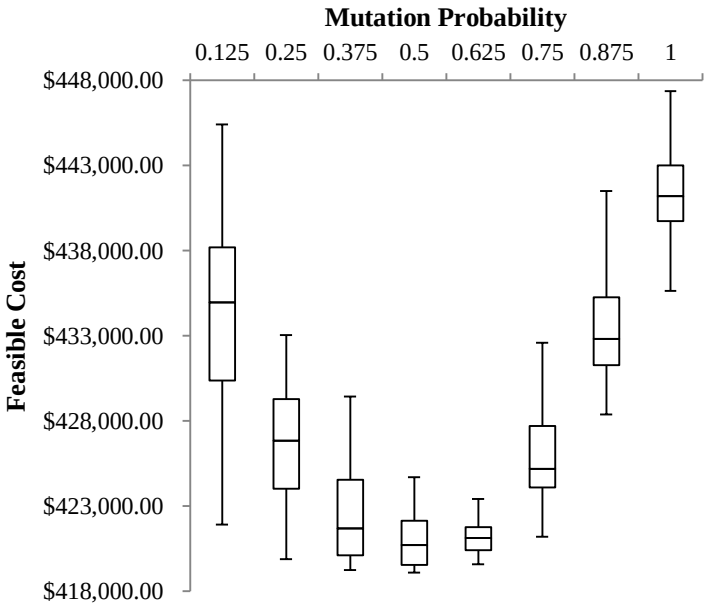


Figure 9-3. Box and whisker plot showing mean mutation probability results for the Two Loop benchmark problem

9.1.1.2 Foss Poly 1

Table 9-4. Mean penalty cost results for the Foss Poly 1 benchmark problem

Mean Penalty Cost								
Penalty	Best Feasible Fitness	Best Feasible Cost	Mean Best Feasible Fitness	Mean Best Feasible Cost	Mean Best Feasible Cost SD	Mean Iterations To Feasible	Best Known Solution (% of Solutions)
$ 2,000.00	1.76617E-05	$ 57,557.55	8.59506E-06	$ 145,843.89	$ 63,032.06	-	0%
$ 4,000.00	2.4426E-05	$ 42,513.52	1.26283E-05	$ 101,357.52	$ 50,435.81	-	0%
$ 6,000.00	2.59611E-05	$ 40,234.57	1.73755E-05	$ 69,907.05	$ 32,794.29	-	0%
$ 8,000.00	2.63485E-05	$ 39,922.50	2.03225E-05	$ 55,362.38	$ 18,692.10	-	0%
$ 10,000.00	2.64121E-05	$ 39,688.96	2.18693E-05	$ 49,341.63	$ 10,845.40	-	0%
$ 12,000.00	2.66782E-05	$ 39,311.88	2.25E-05	$ 47,307.21	$ 6,066.35	-	0%
$ 14,000.00	2.66286E-05	$ 39,406.85	2.2703E-05	$ 46,524.12	$ 4,239.21	-	0%
$ 16,000.00	2.64301E-05	$ 39,771.33	2.27942E-05	$ 46,417.86	$ 3,879.82	-	0%

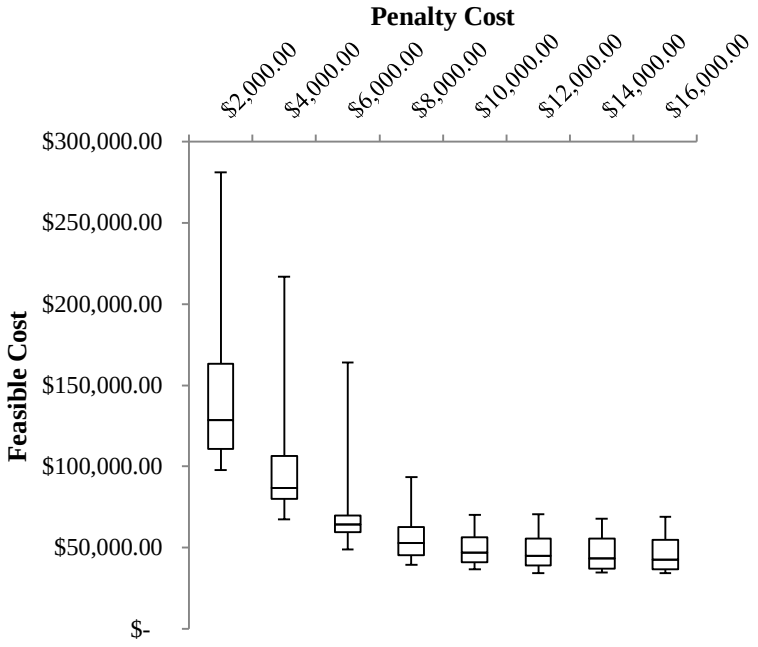


Figure 9-4. Box and whisker plot showing mean penalty cost results for the Foss Poly 1 benchmark problem

Table 9-5. Mean tournament size results for the Foss Poly 1 benchmark problem

Mean Tournament Size							
Tournament Size	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
2%	2.43075E-05	$ 43,863.12	1.88709E-05	$ 63,993.99	$ 16,204.45	-	0%
3%	2.47572E-05	$ 42,897.61	1.87367E-05	$ 66,820.99	$ 19,187.59	-	0%
4%	2.48672E-05	$ 42,538.58	1.86844E-05	$ 67,492.15	$ 19,814.87	-	0%
5%	2.50928E-05	$ 42,264.45	1.85773E-05	$ 70,526.98	$ 23,516.56	-	0%
6%	2.54205E-05	$ 41,711.41	1.86436E-05	$ 70,372.46	$ 25,602.82	-	0%
7%	2.54315E-05	$ 41,690.33	1.84197E-05	$ 74,143.83	$ 28,267.21	-	0%
8%	2.54315E-05	$ 41,690.33	1.84197E-05	$ 74,143.83	$ 28,267.21	-	0%
9%	2.52382E-05	$ 41,751.34	1.84355E-05	$ 74,567.42	$ 29,124.35	-	0%

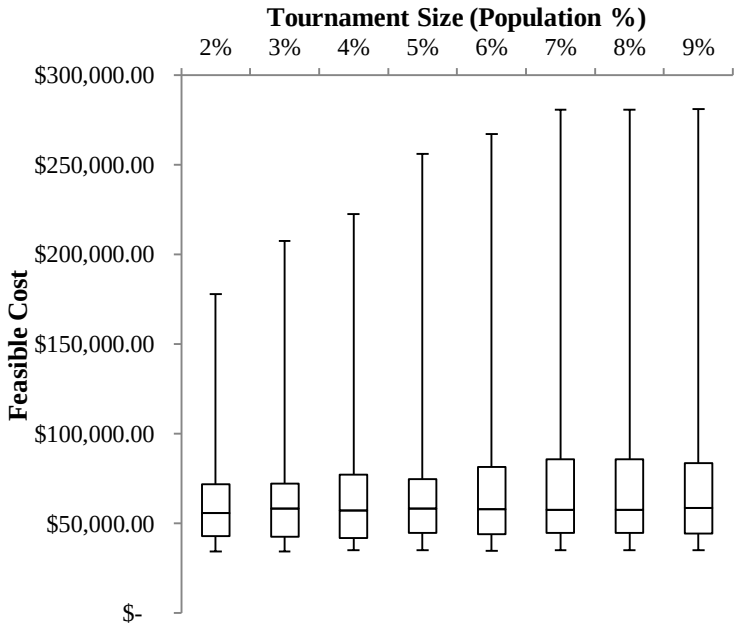


Figure 9-5. Box and whisker plot showing mean tournament size results for the Foss Poly 1 benchmark problem

Table 9-6. Mean mutation probability results for the Foss Poly 1 benchmark problem

Mean Mutation Rate							
Mutation Rate	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
0.0035	3.02239E-05	$ 35,076.21	1.93249E-05	$ 96,849.99	$ 56,581.02	-	0%
0.007	3.08028E-05	$ 34,173.94	2.20567E-05	$ 68,930.28	$ 32,517.62	-	0%
0.0105	2.93164E-05	$ 35,500.26	2.17314E-05	$ 63,422.99	$ 25,485.50	-	0%
0.014	2.65793E-05	$ 38,504.00	2.04084E-05	$ 61,201.15	$ 20,306.86	-	0%
0.0175	2.41002E-05	$ 42,247.63	1.86845E-05	$ 62,421.78	$ 16,244.68	-	0%
0.021	2.18319E-05	$ 46,148.28	1.69178E-05	$ 66,004.94	$ 14,466.62	-	0%
0.0245	1.97943E-05	$ 50,691.45	1.55039E-05	$ 69,081.28	$ 12,385.00	-	0%
0.028	1.78976E-05	$ 56,065.38	1.41603E-05	$ 74,149.25	$ 11,997.74	-	0%

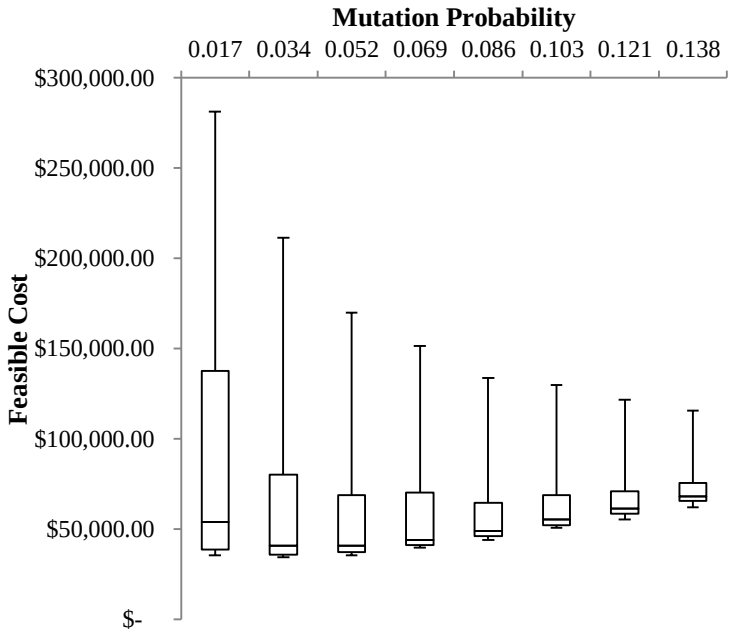


Figure 9-6. Box and whisker plot showing mean mutation probability results for the Foss Poly 1 benchmark problem

9.1.1.3 Hanoi

Table 9-7. Mean penalty cost results for the Hanoi benchmark problem

Mean Penalty Cost							
Penalty	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
$ 500,000.00	1.63578E-07	$ 6,113,631.88	1.58877E-07	$ 6,298,390.16	$ 117,962.98	249.61	2%
$ 1,000,000.00	1.636E-07	$ 6,112,699.69	1.58739E-07	$ 6,303,661.41	$ 114,756.34	251.17	2%
$ 1,500,000.00	1.63601E-07	$ 6,112,725.78	1.58812E-07	$ 6,300,633.75	$ 114,906.78	250.69	2%
$ 2,000,000.00	1.63593E-07	$ 6,113,027.03	1.58722E-07	$ 6,304,321.72	$ 118,038.95	251.24	2%
$ 2,500,000.00	1.63651E-07	$ 6,110,783.91	1.58654E-07	$ 6,307,037.66	$ 118,964.97	251.25	2%
$ 3,000,000.00	1.63467E-07	$ 6,117,838.59	1.58666E-07	$ 6,306,793.75	$ 116,336.45	251.29	2%
$ 3,500,000.00	1.63569E-07	$ 6,113,989.38	1.58612E-07	$ 6,309,069.38	$ 117,772.46	250.49	2%
$ 4,000,000.00	1.63567E-07	$ 6,114,033.59	1.58628E-07	$ 6,308,491.72	$ 118,967.78	250.83	2%

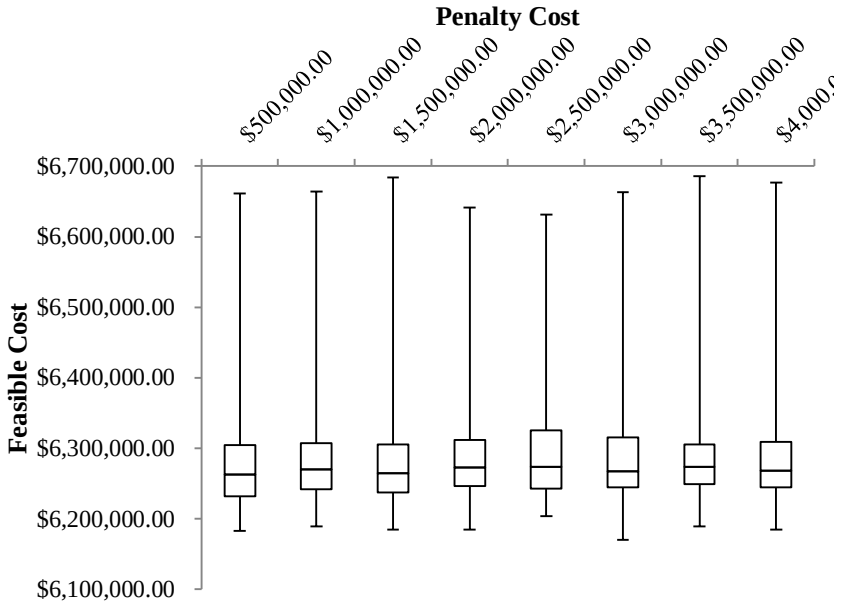


Figure 9-7. Box and whisker plot showing mean penalty cost results for the Hanoi benchmark problem

Table 9-8. Mean tournament size results for the Hanoi benchmark problem

Mean Tournament Size							
Tournament Size	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
2%	1.63604E-07	$ 6,112,581.72	1.59595E-07	$ 6,268,503.13	$ 103,194.13	431.31	3%
3%	1.63652E-07	$ 6,110,768.59	1.594E-07	$ 6,276,852.97	$ 108,471.44	330.72	2%
4%	1.63671E-07	$ 6,110,058.59	1.59094E-07	$ 6,289,173.59	$ 114,794.58	265.88	2%
5%	1.63572E-07	$ 6,113,766.72	1.58742E-07	$ 6,303,606.88	$ 117,801.97	232.77	2%
6%	1.63611E-07	$ 6,112,403.44	1.58448E-07	$ 6,315,366.56	$ 120,282.00	214.21	2%
7%	1.63524E-07	$ 6,115,659.84	1.58248E-07	$ 6,324,097.81	$ 124,176.58	178.82	1%
8%	1.63524E-07	$ 6,115,659.84	1.58248E-07	$ 6,324,097.81	$ 124,176.58	178.82	1%
9%	1.63466E-07	$ 6,117,831.09	1.57935E-07	$ 6,336,700.78	$ 124,809.43	174.04	1%

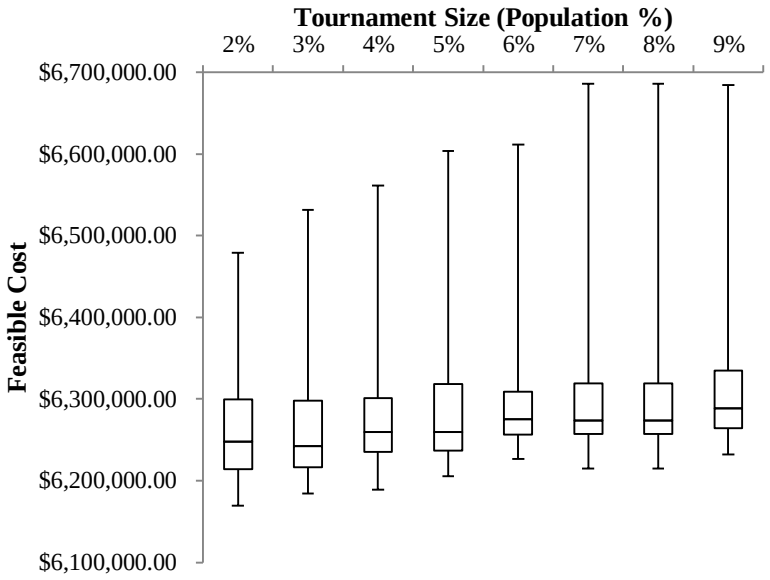


Figure 9-8. Box and whisker plot showing mean tournament size results for the Hanoi benchmark problem

Table 9-9. Mean mutation probability results for the Hanoi benchmark problem

Mean Mutation Rate							
Mutation Rate	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
0.01	1.61476E-07	$ 6,193,254.22	1.52234E-07	$ 6,575,597.19	$ 199,225.19	228.41	0%
0.02	1.64046E-07	$ 6,095,890.31	1.58142E-07	$ 6,325,686.88	$ 117,623.28	216.32	1%
0.03	1.64344E-07	$ 6,084,804.84	1.59538E-07	$ 6,269,939.84	$ 105,510.89	223.41	3%
0.04	1.64388E-07	$ 6,083,188.75	1.60149E-07	$ 6,246,024.84	$ 104,582.60	230.84	6%
0.05	1.64393E-07	$ 6,082,995.31	1.60684E-07	$ 6,225,125.63	$ 102,081.24	245.52	4%
0.06	1.64059E-07	$ 6,095,393.28	1.60355E-07	$ 6,237,960.31	$ 103,957.04	259.53	0%
0.07	1.63415E-07	$ 6,119,417.97	1.59812E-07	$ 6,259,094.22	$ 104,110.79	298.07	0%
0.08	1.62503E-07	$ 6,153,785.16	1.58798E-07	$ 6,298,970.63	$ 100,615.70	304.45	0%

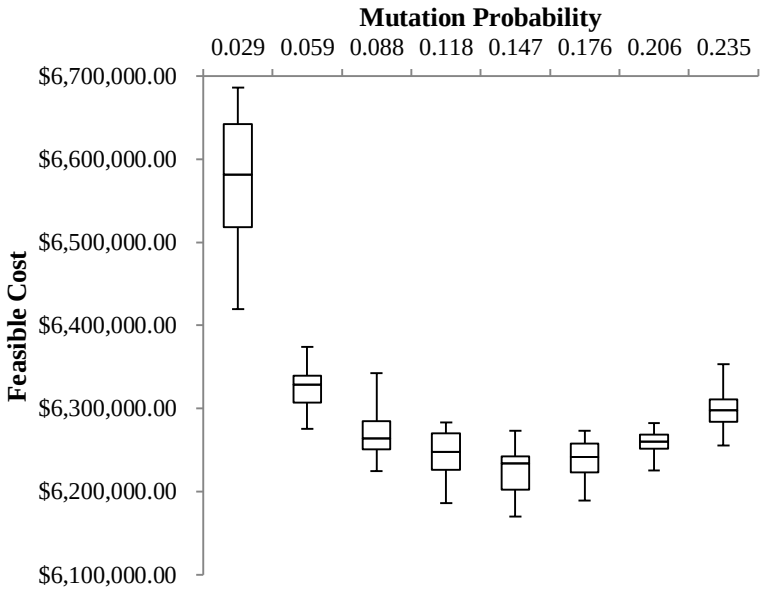


Figure 9-9. Box and whisker plot showing mean mutation probability results for the Hanoi benchmark problem

9.1.1.4 New York Tunnels

Table 9-10. Mean penalty cost results for the New York Tunnels benchmark problem

Mean Penalty Cost							
Penalty	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
$ 1,000,000.00	2.45421E-08	$ 40,807,485.94	2.18658E-08	$ 46,076,482.81	$ 2,955,729.98	-	0%
$ 2,000,000.00	2.50625E-08	$ 39,978,590.63	2.37362E-08	$ 42,282,942.19	$ 1,255,711.51	-	2%
$ 3,000,000.00	2.50795E-08	$ 39,946,759.38	2.40668E-08	$ 41,742,443.75	$ 1,105,459.96	-	13%
$ 4,000,000.00	2.51916E-08	$ 39,753,339.06	2.41832E-08	$ 41,545,868.75	$ 992,283.71	-	18%
$ 5,000,000.00	2.50877E-08	$ 39,935,000.00	2.41389E-08	$ 41,637,690.63	$ 1,114,315.42	-	18%
$ 6,000,000.00	2.51119E-08	$ 39,893,528.13	2.41307E-08	$ 41,659,303.13	$ 1,155,312.70	-	20%
$ 7,000,000.00	2.51395E-08	$ 39,845,567.19	2.40692E-08	$ 41,774,118.75	$ 1,226,517.44	-	21%
$ 8,000,000.00	2.51751E-08	$ 39,778,031.25	2.40531E-08	$ 41,807,190.63	$ 1,223,406.41	-	21%

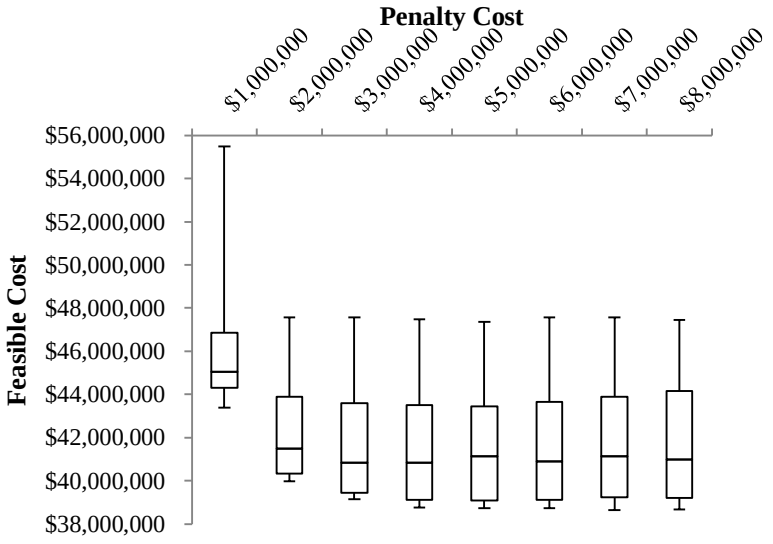


Figure 9-10. Box and whisker plot showing mean penalty cost results for the New York Tunnels benchmark problem

Table 9-11. Mean tournament size results for the New York Tunnels benchmark problem

Mean Tournament Size							
Tournament Size	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
2%	2.48737E-08	$ 40,294,915.63	2.36743E-08	$ 42,514,120.31	$ 1,159,685.24	-	16%
3%	2.50119E-08	$ 40,057,870.31	2.37448E-08	$ 42,388,026.56	$ 1,313,729.47	-	15%
4%	2.49511E-08	$ 40,161,285.94	2.37496E-08	$ 42,360,810.94	$ 1,302,264.60	-	14%
5%	2.50551E-08	$ 39,979,826.56	2.38052E-08	$ 42,261,412.50	$ 1,320,471.69	-	15%
6%	2.50202E-08	$ 40,034,209.38	2.37694E-08	$ 42,351,242.19	$ 1,393,054.09	-	14%
7%	2.51554E-08	$ 39,810,035.94	2.38337E-08	$ 42,210,390.63	$ 1,528,018.04	-	13%
8%	2.51554E-08	$ 39,810,035.94	2.38337E-08	$ 42,210,390.63	$ 1,528,018.04	-	13%
9%	2.5167E-08	$ 39,790,121.88	2.38331E-08	$ 42,229,646.88	$ 1,483,495.95	-	14%

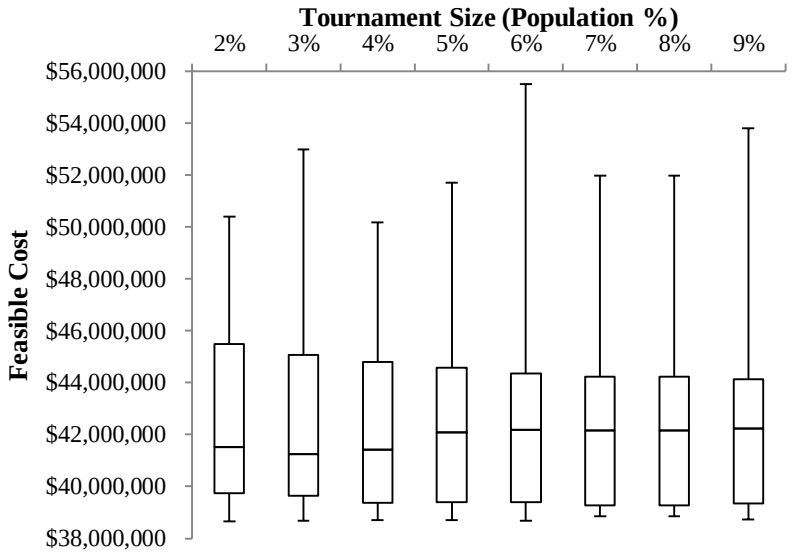


Figure 9-11. Box and whisker plot showing mean tournament size results for the New York Tunnels benchmark problem

Table 9-12. Mean mutation probability results for the New York Tunnels benchmark problem

Mean Mutation Rate							
Mutation Rate	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
0.012	2.56693E-08	$ 38,979,357.81	2.35434E-08	$ 43,052,400.00	$ 3,856,647.77	-	11%
0.024	2.57721E-08	$ 38,809,570.31	2.50719E-08	$ 40,066,618.75	$ 1,139,355.19	-	46%
0.036	2.57995E-08	$ 38,764,223.44	2.51969E-08	$ 39,803,987.50	$ 796,294.90	-	46%
0.048	2.5792E-08	$ 38,774,978.13	2.49838E-08	$ 40,106,073.44	$ 767,791.05	-	10%
0.06	2.54756E-08	$ 39,258,398.44	2.42439E-08	$ 41,301,076.56	$ 946,360.71	-	0%
0.072	2.47713E-08	$ 40,382,171.88	2.33497E-08	$ 42,862,175.00	$ 1,037,115.35	-	0%
0.084	2.39723E-08	$ 41,727,125.00	2.23928E-08	$ 44,693,145.31	$ 1,151,269.07	-	0%
0.096	2.31377E-08	$ 43,242,476.56	2.14615E-08	$ 46,640,564.06	$ 1,333,903.09	-	0%

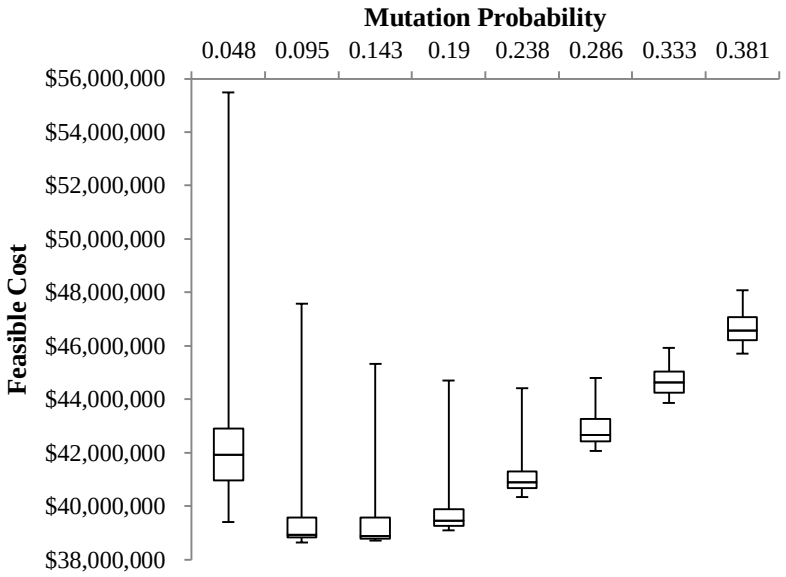


Figure 9-12. Box and whisker plot showing mean mutation probability results for the New York Tunnels benchmark problem

9.1.1.5 Modena

Table 9-13. Mean penalty cost results for the Modena benchmark problem

Mean Penalty Cost							
Penalty	Best Feasible Fitness	Best Feasible Cost	Mean Best Feasible Fitness	Mean Best Feasible Cost	Mean Best Feasible Cost SD	Mean Iterations To Feasible	Best Known Solution (% of Solutions)
$ 10,000.00	1.55118E-05	$ 64,559.21	1.47296E-05	$ 68,395.99	$ 3,464.75	-	4%
$ 20,000.00	1.54902E-05	$ 64,661.50	1.47538E-05	$ 68,192.68	$ 3,015.87	-	4%
$ 30,000.00	1.55178E-05	$ 64,526.94	1.47034E-05	$ 68,538.13	$ 3,486.27	-	3%
$ 40,000.00	1.54889E-05	$ 64,654.65	1.4679E-05	$ 68,631.70	$ 3,449.01	-	3%
$ 50,000.00	1.54679E-05	$ 64,756.67	1.46488E-05	$ 68,826.92	$ 3,623.48	-	3%
$ 60,000.00	1.54899E-05	$ 64,653.78	1.4673E-05	$ 68,675.37	$ 3,567.33	-	2%
$ 70,000.00	1.5478E-05	$ 64,706.05	1.46573E-05	$ 68,775.18	$ 3,553.96	-	3%
$ 80,000.00	1.54946E-05	$ 64,638.96	1.46447E-05	$ 68,856.09	$ 3,762.71	-	3%

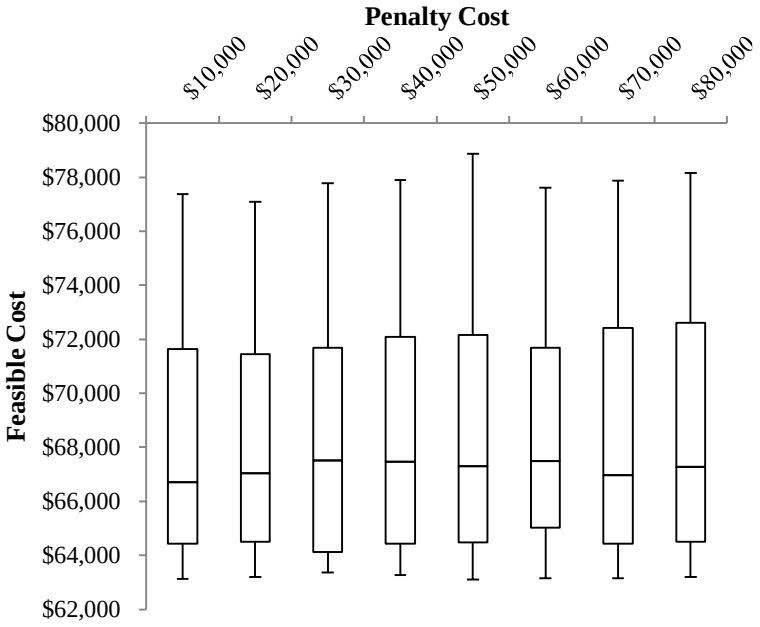


Figure 9-13. Box and whisker plot showing mean penalty cost results for the Modena benchmark problem

Table 9-14. Mean tournament size results for the Modena benchmark problem

Mean Tournament Size							
Tournament Size	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
2%	1.53675E-05	$ 65,219.34	1.46975E-05	$ 68,469.56	$ 2,366.29	-	3%
3%	1.53683E-05	$ 65,218.70	1.47015E-05	$ 68,494.15	$ 2,866.68	-	3%
4%	1.54856E-05	$ 64,674.21	1.47148E-05	$ 68,420.57	$ 3,055.49	-	3%
5%	1.55108E-05	$ 64,558.97	1.47124E-05	$ 68,453.73	$ 3,420.92	-	3%
6%	1.55115E-05	$ 64,548.75	1.46914E-05	$ 68,642.27	$ 3,864.00	-	4%
7%	1.55562E-05	$ 64,352.28	1.46595E-05	$ 68,801.30	$ 4,127.24	-	3%
8%	1.55562E-05	$ 64,352.28	1.46595E-05	$ 68,801.30	$ 4,127.24	-	3%
9%	1.5583E-05	$ 64,233.22	1.46531E-05	$ 68,809.18	$ 4,095.52	-	3%

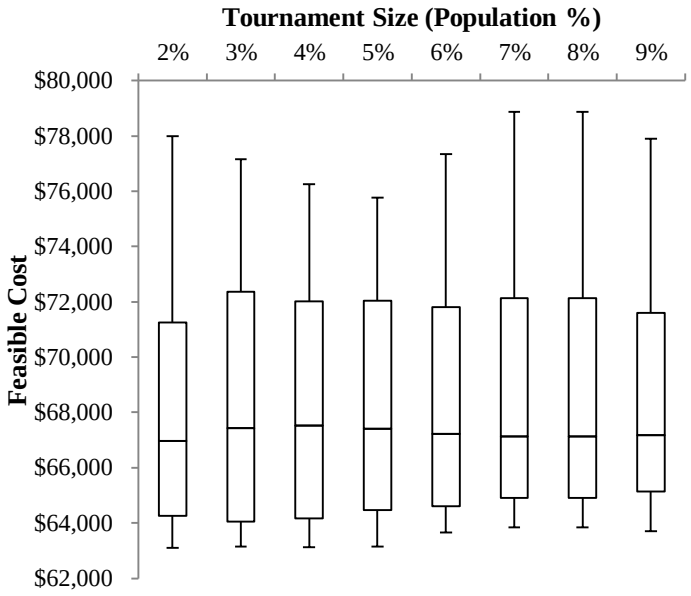


Figure 9-14. Box and whisker plot showing mean tournament size results for the Modena benchmark problem

Table 9-15. Mean mutation probability results for the Modena benchmark problem

Mean Mutation Rate							
Mutation Rate	**Best Feasible Fitness**	**Best Feasible Cost**	**Mean Best Feasible Fitness**	**Mean Best Feasible Cost**	**Mean Best Feasible Cost SD**	**Mean Iterations To Feasible**	**Best Known Solution (% of Solutions)**
0.011	1.5937E-05	$ 62,746.98	1.38915E-05	$ 73,600.09	$ 11,645.53	-	3%
0.022	1.59546E-05	$ 62,677.70	1.54585E-05	$ 64,898.37	$ 3,747.02	-	11%
0.033	1.59544E-05	$ 62,678.44	1.56659E-05	$ 63,924.72	$ 2,360.63	-	10%
0.044	1.59058E-05	$ 62,870.51	1.55886E-05	$ 64,199.70	$ 1,721.27	-	0%
0.055	1.56794E-05	$ 63,780.40	1.5151E-05	$ 66,045.67	$ 1,627.36	-	0%
0.066	1.52765E-05	$ 65,468.27	1.45689E-05	$ 68,694.50	$ 1,822.64	-	0%
0.077	1.48574E-05	$ 67,328.59	1.39108E-05	$ 71,969.71	$ 2,287.70	-	0%
0.088	1.43741E-05	$ 69,606.86	1.32543E-05	$ 75,559.29	$ 2,711.24	-	0%

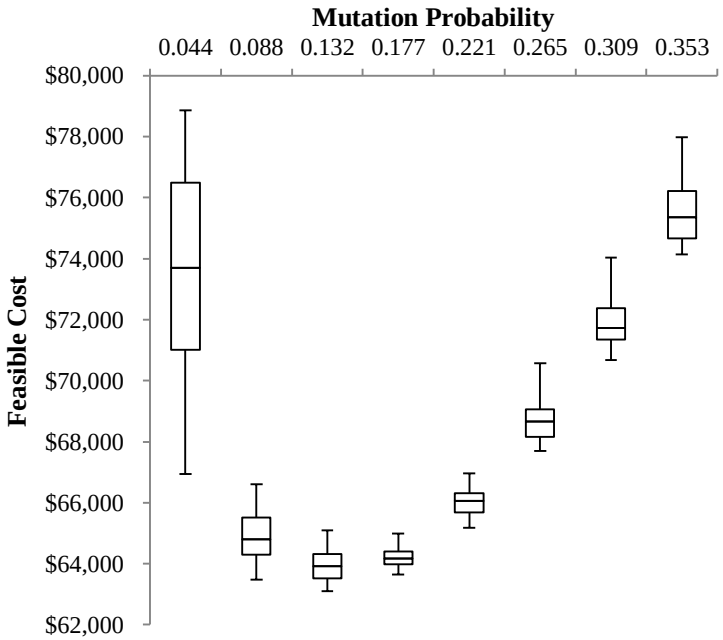


Figure 9-15. Box and whisker plot showing mean mutation probability results for the Modena benchmark problem

9.1.2 Replacement Strategy

9.1.2.1 Two Loop

Table 9-16. Mean replacement strategy results for the Two Loop benchmark problem

	Mean Best Feasible Cost	Mean Best Feasible Cost SD
Worst / Conditional	$ 419,100.00	$ 303.05
Oldest / Conditional	$ 424,080.00	$ 18,505.20
Random / Conditional	$ 419,360.00	$ 692.82
Worst / Unconditional	$ 419,320.00	$ 471.21
Oldest / Unconditional	$ 484,380.00	$ 16,256.73
Random / Unconditional	$ 482,960.00	$ 18,370.30

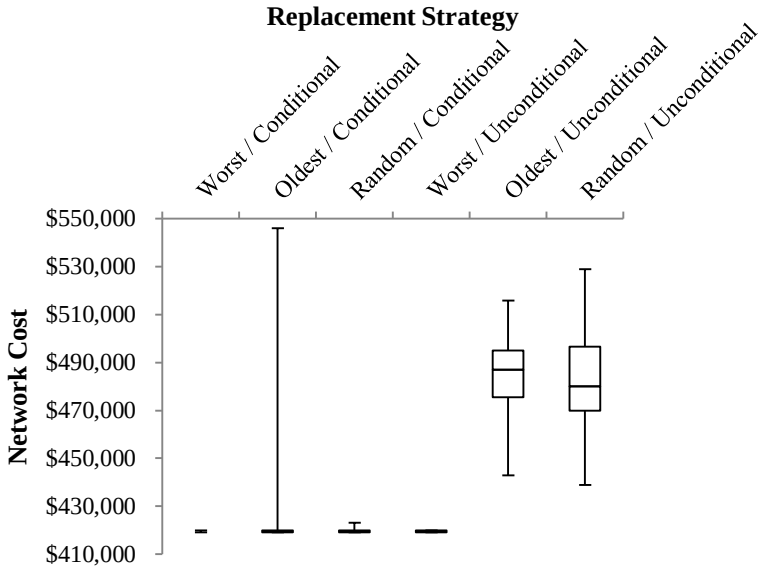


Figure 9-16. Box and whisker plot showing replacement strategy results for the Two Loop benchmark problem

9.1.2.2 Foss Poly 1

Table 9-17. Mean replacement strategy results for the Foss Poly 1 benchmark problem

	Mean Best Feasible Cost	Mean Best Feasible Cost SD
Worst / Conditional	$ 43,783.45	$ 3,885.27
Oldest / Conditional	$ 45,734.77	$ 3,641.25
Random / Conditional	$ 44,411.77	$ 3,798.98
Worst / Unconditional	$ 43,292.16	$ 3,496.99
Oldest / Unconditional	$ 201,773.72	$ 12,839.57
Random / Unconditional	$ 202,280.88	$ 11,732.34

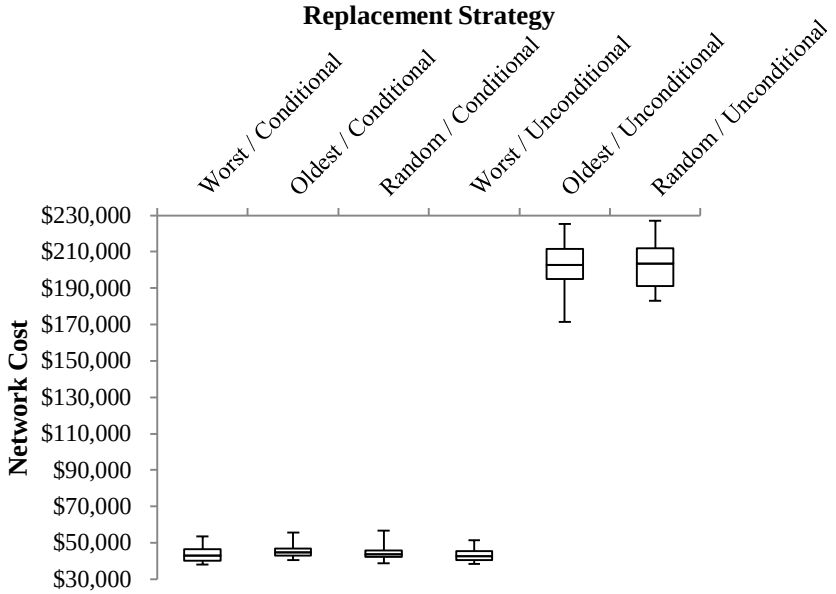


Figure 9-17. Box and whisker plot showing replacement strategy results for the Foss Poly 1 benchmark problem

9.1.2.3 Hanoi

Table 9-18. Mean replacement strategy results for the Hanoi benchmark problem

	Mean Best Feasible Cost	Mean Best Feasible Cost SD
Worst / Conditional	$ 6,183,407.60	$ 99,800.06
Oldest / Conditional	$ 6,195,220.00	$ 90,184.37
Random / Conditional	$ 6,206,049.80	$ 104,525.92
Worst / Unconditional	$ 6,205,100.00	$ 95,292.54
Oldest / Unconditional	$ 7,520,530.00	$ 118,098.27
Random / Unconditional	$ 7,486,498.20	$ 153,245.31

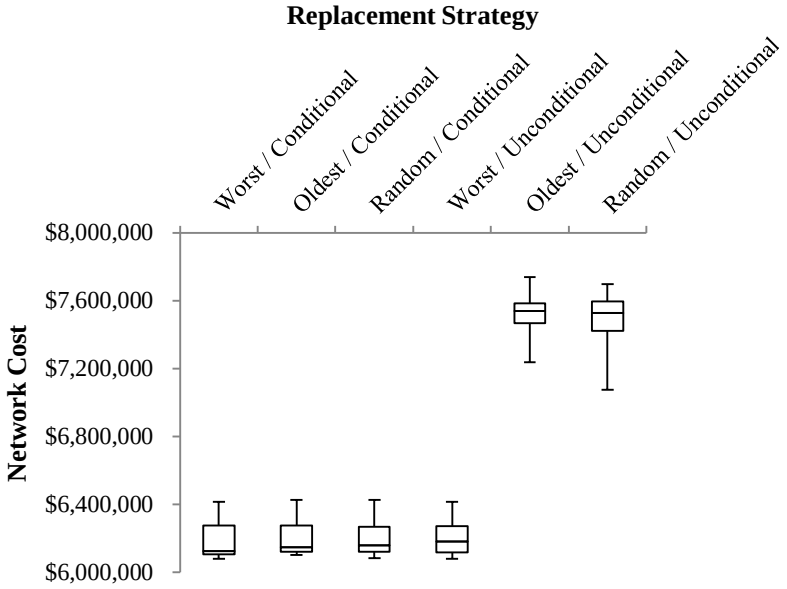


Figure 9-18. Box and whisker plot showing replacement strategy results for the Hanoi benchmark problem

9.1.2.4 New York Tunnels

Table 9-19. Mean replacement strategy results for the New York Tunnels benchmark problem

	Mean Best Feasible Cost	Mean Best Feasible Cost SD
Worst / Conditional	$ 38,916,008.00	$ 321,846.53
Oldest / Conditional	$ 39,041,774.00	$ 423,130.51
Random / Conditional	$ 38,835,564.00	$ 266,068.69
Worst / Unconditional	$ 38,832,386.00	$ 259,893.85
Oldest / Unconditional	$ 47,642,064.00	$ 1,165,752.69
Random / Unconditional	$ 47,257,584.00	$ 1,085,688.89

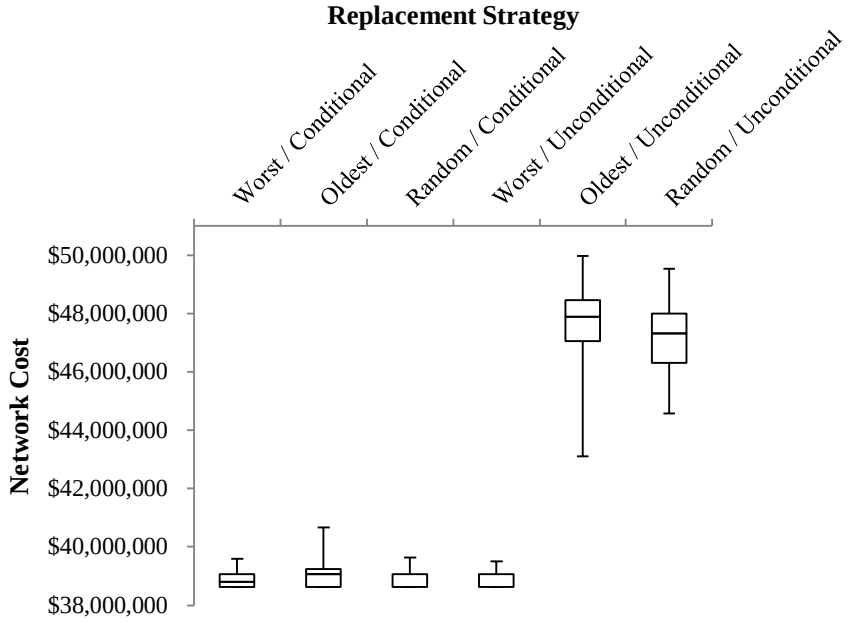


Figure 9-19. Box and whisker plot showing replacement strategy results for the New York Tunnels benchmark problem

9.1.2.5 Modena

Table 9-20. Mean replacement strategy results for the Modena benchmark problem

	Mean Best Feasible Cost	Mean Best Feasible Cost SD
Worst / Conditional	$ 6,125,819.40	$ 124,548.09
Oldest / Conditional	$ 6,514,257.80	$ 225,005.51
Random / Conditional	$ 6,164,987.60	$ 151,459.31
Worst / Unconditional	$ 6,099,923.60	$ 136,019.69
Oldest / Unconditional	$ 8,994,304.60	$ 150,495.73
Random / Unconditional	$ 9,009,283.20	$ 143,062.54

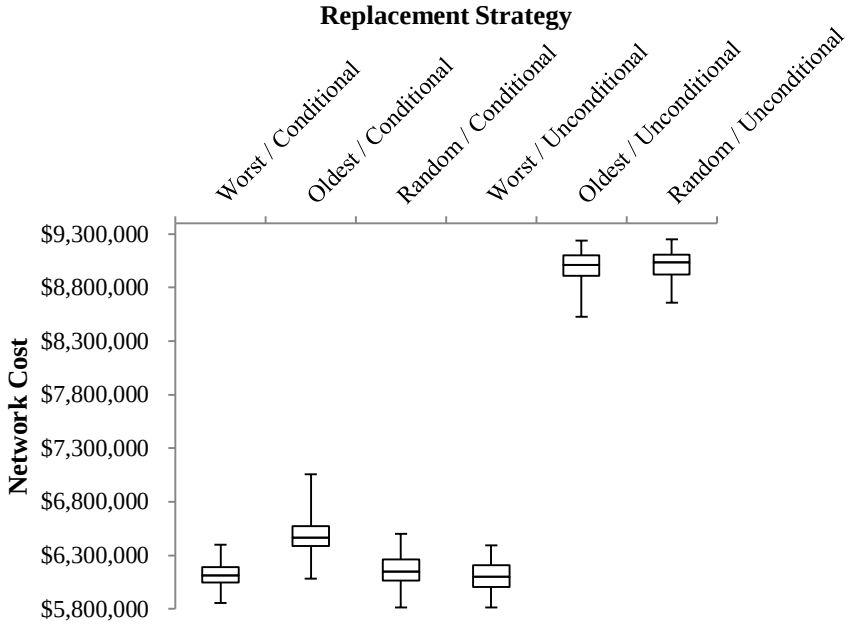


Figure 9-20. Box and whisker plot showing replacement strategy results for the Modena benchmark problem

Bibliography

[1] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[2] J. H. Holland, *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975.

[3] I. Rechenberg, “Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution,” 1971.

[4] R. Storn and K. Price, “Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces,” *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1995.

[5] G. S. Hornby, A. Globus, D. S. Linden, and J. D. Lohn, “Automated antenna design with evolutionary algorithms,” *AIAA Sp.*, pp. 19–21, 2006.

[6] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose, “Multiobjective evolutionary computation for supersonic wing-shape optimization,” *Evol. Comput. IEEE Trans.*, vol. 4, no. 2, pp. 182–187, 2000.

[7] K.-S. Shin and Y.-J. Lee, “A genetic algorithm application in bankruptcy prediction modeling,” *Expert Syst. Appl.*, vol. 23, no. 3, pp. 321–328, 2002.

[8] S. DiPaola and G. Liane, “Incorporating characteristics of human creativity into an evolutionary art algorithm,” *Genet. Program. Evolvable Mach.*, vol. 10, no. 2, pp. 97–110, 2009.

[9] J. A. Biles, “GenJam: A genetic algorithm for generating jazz solos,” *ICMC*, vol. 94, pp. 131–137, 1994.

[10] J. Zhang, H. S. H. Chung, A. W. L. Lo, and T. Huang, “Extended ant colony optimization algorithm for power electronic circuit design,” *Power Electron. IEEE Trans.*, vol. 24, no. 1, pp. 147–162, 2009.

[11] R. Le Riche and R. T. Haftka, “Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm,” *AIAA J.*, vol. 31, no. 5, pp. 951–956, 1993.

[12] D. A. Savic and G. A. Walters, “Genetic Algorithms for Least-Cost Design of Water Distribution Networks,” *J. Water Resour. Plan. Manag.*, vol. 123, no. 2, pp. 67–77, 1997.

[13] A. Marchi, E. Salomons, A. Ostfeld, Z. S. Kapelan, and A. R. Simpson, “Battle of the Water Networks II,” *Water Resour. Plan. Manag.*, vol. 140, no. 7, 2014.

[14] S. D. Keedwell E, Johns M, “Spatial and Temporal Visualisation of Evolutionary Algorithm Decisions in Water Distribution Network Optimisation,” in *VizGEC Workshop, Genetic and Evolutionary Computation Conference*, 2015.

[15] L. V Kantorovich, “A new method of solving of some classes of extremal problems,” *Dokl. Akad. Nauk SSSR*, vol. 28, pp. 211–214, 1940.

[16] N. Karmarkar, “A New Polynomial Time Algorithm for Linear Programming,” *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.

[17] H. Robbins and S. Monro, “A Stochastic Approximation Method,” *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400–407, 1951.

[18] L. Rastrigin, “The convergence of the random search method in the extremal control of a many parameter system,” *Autom. Remote Control*, vol. 24, no. 10, pp. 1337–1342, 1963.

[19] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward, “A Classification of Hyper-heuristics Approaches,” in *Handbook of Metaheuristics*, International Series in Operations Research & Management Science: Springer, 2009.

[20] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1983.

[21] I. H. Osman and G. Laporte, “Metaheuristics: A bibliography,” *Ann. Oper. Res.*, vol. 63, pp. 513–623, 1996.

[22] S. Voß, S. Martello, I. H. Osman, and C. Roucairol, *Meta-Heuristics-Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, 1999.

[23] T. Stützle, *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications*. DISKI Dissertationen zur Künstliken Intelligenz, 1999.

[24] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[25] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983.

[26] F. Glover, “Tabu Search - Part 1,” *ORSA J. Comput.*, vol. 1, no. 2, pp. 190–206, 1989.

[27] F. Glover, “Tabu Search - Part 2,” *ORSA J. Comput.*, vol. 2, no. 1, pp. 4–32, 1990.

[28] M. Dorigo, “Optimization, Learning and Natural Algorithms,” Politecnico di Milano, Italie, 1992.

[29] H. P. Schwefel, *Numerische Optimierung von Computer-Modellen*. 1974.

[30] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[31] L. . Fogel, A. . Owens, and M. . Walsh, *Artificial Intelligence Through Simulated Evolution*. New York, USA: John Wiley & Sons, 1966.

[32] I. Rechenberg, *Cybernetic Solution Path of an Experimental Problem*. Ministry of Aviation, Royal Aircraft Establishment, 1965.

[33] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Stuttgart: Frommann-Holzboog, 1973.

[34] H. P. Schwefel, *Numerische Optimierung uon Computer-Modellen mittels der Ezdutionsstrategie*. Interdisciplinary systems research, 1977.

[35] H. P. Schwefel, *Numerical optimization of computer models*. Chichester: Wiley.

[36] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, Mass: Addison-Wesley, 1989.

[37] M. Mitchell, *An Introduction to Genetic Algorithms*. United States of

America: Massachusetts Institute of Technology, 1996.

[38] A. Oyama, S. Obayashi, and T. Nakamura, “Real-coded adaptive range genetic algorithm applied to transonic wing optimization,” *1*, no. 3, pp. 179–187, 2001.

[39] A. Colorni, M. Dorigo, and V. Maniezzo, “Metaheuristics for high school timetabling,” *Comput. Optim. Appl.*, vol. 9, no. 3, pp. 275–298, 1998.

[40] J. Nicklow *et al.*, “State of the Art for Genetic Algorithms and Beyond in Water Resources Planning and Management,” *J. Water Resour. Plan. Manag.*, vol. 136, no. 4, pp. 412–432, 2010.

[41] P. Reed, B. Minsker, and D. E. Goldberg, “Designing a competent simple genetic algorithm for search and optimization,” *Water Resour. Res.*, vol. 36, no. 12, pp. 3757–3761, 2000.

[42] J. H. Yoon and C. A. Shoemaker, “Improved real-coded GA for groundwater bioremediation,” *J. Comput. Civ. Eng.*, vol. 15, no. 3, pp. 224–231, 2001.

[43] P. Bayer and M. Finkel, “Evolutionary algorithms for the optimization of advective control of contaminated aquifer zones,” *Water Resour. Res.*, vol. 40, no. 6, 2004.

[44] L. C. Chang, F. J., Chen, L., & Chang, “Optimizing the Reservoir Operating Rule Curves by Genetic Algorithms,” *Hydrol. Process.*, vol. 19, no. 11, pp. 2277–2289, 2005.

[45] D. E. Dougherty and R. A. Marryott, “Optimal Groundwater-Management .1. Simulated Annealing,” *Water Resour. Res.*, vol. 27, no. 10, pp. 2493–2508, 1991.

[46] D. K. Karpouzos, F. Delay, K. L. Katsifarakis, and G. de Marsily, “A Multipopulation Genetic Algorithm to Solve the Inverse Problem in Hydrogeology,” *Water Resour. Res.*, vol. 37, no. 9, pp. 2291–2302, 2001.

[47] H. J. Shieh and R. C. Peralta, “Optimal in Situ Bioremediation Design by Hybrid Genetic Algorithm-Simulated Annealing,” *J. Water Resour. Plan. Manag. ASCE*, vol. 131, no. 1, pp. 67–78, 2005.

[48] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. Bristol, UK: IOP Publishing Ltd. and Oxford University Press, 2000.

[49] D. Thierens, D. E. Goldberg, and A. G. Pereira, “Domino Convergence, Drift, and the Temporal-Salience Structure of Problems,” in *IEEE International Conference on Evolutionary Computation*, 1998, pp. 535–540.

[50] H. P. Schwefel and G. Rudolph, *Contemporary Evolution Strategies*. Granada, Spain: Spanish RIG IEEE Neural Networks Council, 1995.

[51] K. Deb and R. B. Agrawal, *Simulated Binary Crossover for Continuous Search Space*. Department of Mechanical Engineering, Indian Institute of Technology, 1994.

[52] K. Deb, D. Joshi, and A. Anand, “Real-Coded Evolutionary Algorithms with Parent-Centric Recombination,” in *IEEE World Congress on Computational Intelligence*, 2002, pp. 61–66.

[53] N. Hansen and A. Ostermeier, “Completely Derandomized Self-

Adaptation in Evolution Strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[54] J. B. Kollat and P. M. Reed, "Comparing State-of-the-Art Evolutionary Multi-Objective Algorithms for Long-Term Groundwater Monitoring Design," *Adv. Water Resour.*, pp. 792–807, 2006.

[55] H. P. Schwefel, *Evolution and Optimum Seeking*. New York, USA: Wiley, 1995.

[56] A. H. Aly and P. C. Peralta, "Comparison of a genetic algorithm and mathematical programming to the design of groundwater cleanup systems," *Water Resour. Res.*, vol. 35, no. 8, pp. 2415–2425, 1999.

[57] A. H. Aly and R. C. Peralta, "Optimal design of aquifer cleanup systems under uncertainty using a neural network and a genetic algorithm," *Water Resour. Res.*, vol. 35, no. 8, pp. 2523–2532, 1999.

[58] P. Reed, B. S. Minsker, and D. E. Goldberg, "Simplifying multiobjective optimization: An automated design methodology for the nondominated sorted genetic algorithm-II," *Water Resour. Res.*, vol. 39, no. 7, pp. 21–25, 2003.

[59] M. S. Gibbs, G. C. Dandy, and H. R. Maier, "A genetic algorithm calibration method based on convergence due to genetic drift," *Inf. Sci. (Ny).*, vol. 178, no. 14, pp. 2857–2869, 2008.

[60] A. Charnes, W. W. Cooper, and R. O. Ferguson, "Optimal estimation of executive compensation by linear programming," *Manage. Sci.*, vol. 1, no. 2, pp. 138–151, 1955.

[61] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization," *ICGA*, vol. 93, pp. 416–423, 1993.

[62] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.

[63] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 2000, pp. 849–858.

[64] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *Evol. Comput.*, vol. 3, no. 4, pp. 257–271, 1999.

[65] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Zurich, 2001.

[66] J. D. Knowles and D. W. Corne, "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation," in *1999 Congress on Evolutionary Computation*, 1999, pp. 98–105.

[67] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization," in *Parallel Problem Solving from Nature VI Conference*, 2000, pp. 839–848.

[68] C. A. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, USA: Springer, 2007.

[69] P. R. Bhave, *Optimal Design of Water Distribution Networks*. Pangbourne: Alpha Science International Ltd, 2003.

[70] M. M. Eusuff and K. E. Lansey, “Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm,” *Water Resour. Plan. Manag.*, vol. 129, no. 3, pp. 210–225, 2003.

[71] I. C. Goulter, “Systems Analysis in Water Distribution Network Design: From Theory to Practice,” *J. Water Resour. Plan. Manag.*, vol. 118, no. 3, pp. 238–248, May 1992.

[72] J. Reca, J. Martínez, C. Gil, and R. Baños, “Application of Several Meta-Heuristic Techniques to the Optimization of Real Looped Water Distribution Networks,” *Water Resour. Manag.*, vol. 22, no. 10, pp. 1367–1379, Dec. 2008.

[73] J. Reca and J. Martínez, “Genetic algorithms for the design of looped irrigation water distribution networks,” *Water Resour. Res.*, vol. 42, no. 5, May 2006.

[74] T. M. Walski, “Optimization and pipe sizing decisions,” *Water Resour. Plan. Manag.*, vol. 121, no. 4, pp. 340–343, 1995.

[75] D. T. Lauria, “Discussion of ‘Systems analysis in water-distribution network design: From theory to practice,’ by I.C Goulter,” *Water Resour. Plan. Manag.*, vol. 119, no. 6, pp. 720–722, 1993.

[76] A. R. Simpson, G. C. Dandy, and L. J. Murphy, “Genetic Algorithms Compared to Other Techniques for Pipe Optimization,” *Water Resour. Plan. Manag.*, vol. 120, no. 4, pp. 423–443, 1994.

[77] J. Gessler, “Pipe Network Optimization by Enumeration,” in *Computer Applications for Water Resources*, ASCE, 1985, pp. 572–581.

[78] I. C. Goulter and D. R. Morgan, “An Integrated Approach to the Layout and Design of Water Distribution Networks,” *Civ. Eng. Syst.*, vol. 2, no. 2, pp. 104–113, 1985.

[79] G. Syswerda, “Uniform crossover in genetic algorithms,” in *Third International Conference on Genetic Algorithms*, 1989, pp. 2–9.

[80] G. C. Dandy, A. R. Simpson, and L. J. Murphy, “An Improved Genetic Algorithm for Pipe Network Optimization,” *Water Resour. Res.*, vol. 32, no. 2, p. 449, 1996.

[81] D. Halhal, G. A. Walters, D. Ouazar, and D. A. Savic, “Water network rehabilitation with structured messy genetic algorithm,” *Water Resour. Plan. Manag.*, vol. 123, no. 3, pp. 137–146, 1997.

[82] C. Xu and I. C. Goulter, “Reliability-Based Optimal Design of Water Distribution Networks,” *Water Resour. Plan. Manag.*, vol. 125, no. 6, pp. 352–362, 1999.

[83] V. Pareto, *Cours D’Economie Politique*. Lausanne, Switzerland: Rouge, 1896.

[84] S. E. Cieniawski, J. W. Eheart, and S. Ranjithan, “USING GENETIC ALGORITHMS TO SOLVE A MULTIOBJECTIVE GROUNDWATER MONITORING PROBLEM,” *Water Resour. Res.*, vol. 31, no. 2, pp. 399–409, 1995.

[85] P. M. Reed and B. S. Minsker, “Striking the balance: Long-term

groundwater monitoring design for conflicting objectives," *Water Resour. Plan. Manag.*, vol. 130, no. 2, pp. 140–149, 2004.

[86] R. Farmani, D. A. Savic, and G. A. Walters, "Evolutionary multi-objective optimization in water distribution network design," *Eng. Optim.*, vol. 37, no. 2, pp. 167–183, Mar. 2005.

[87] K. Deb, *Multi-objective optimization using evolutionary algorithms*. New York, USA: Wiley, 2001.

[88] J. B. Kollat and P. M. Reed, "A computational scaling analysis of multiobjective evolutionary algorithms in long-term groundwater monitoring applications," *Adv. Water Resour.*, vol. 30, no. 3, pp. 408–419, 2007.

[89] A. K. Morales and C. V. Quezada, "A universal eclectic genetic algorithm for constrained optimization," in *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing*, 1998, pp. 518–522.

[90] W. Siedlecki and J. Sklansky, "Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition," in *Proceedings of the third international conference on genetic algorithms*, 1989, pp. 141–150.

[91] Z. Michalewic and N. . Attia, "Evolutionary optimization of constrained problems," in *3rd Annual Conference on Evolutionary Programming*, 1994, pp. 98–108.

[92] T. Back, F. Hoffmeister, and H. P. Schwefel, "A survey of evolution strategies," in *Fourth International Conference on Genetic Algorithms*, 1991, pp. 2–9.

[93] R. . Le Riche, C. Knopf-Lenoir, and R. . Haftka, "A segregated genetic algorithm for constrained structural optimization," in *Sixth International Conference on Genetic Algorithms*, 1995, pp. 558–565.

[94] C. A. . Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civ. Eng. Environ. Syst.*, vol. 17, pp. 319–346, 2000.

[95] C. a Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Comput. Methods Appl. Mech. Eng.*, vol. 191, no. 11–12, pp. 1245–1287, Jan. 2002.

[96] Z. Michalewicz, *Genetic Algorithms+Data Structures = Evolution Programs*. Berlin: Springer, 1992.

[97] J. . Bean, "Genetics and random keys for sequencing and optimization," 1992.

[98] J. . Bean, "Genetics and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, no. 2, pp. 154–160, 1994.

[99] V. Kumar, "Algorithms for constraint-satisfaction problems: a survey," *AI Mag*, pp. 32–44, 1992.

[100] B. Paechter, R. Rankin, A. Cumming, and T. Fogarty, "Timetabling the classes of an entire university with an evolutionary algorithm," in *5th International Conference on Parallel Problem Solving from Nature (PPSN V)*, 1998, pp. 865–874.

[101] G. . Liepins and M. . Vose, “Representational issues in genetic optimization,” *Theor. Comput. Sci.*, vol. 2, no. 2, pp. 4–30, 1990.

[102] G. . Liepins and W. . Potter, “A genetic algorithm approach to multiple-fault diagnosis,” in *Handbook of Genetic Algorithms*, New York, USA: Van Nostrand Reinhold, 1991, pp. 237–250.

[103] A. Smith and D. Coit, “Constraint handling techniques – penalty functions,” in *Handbook of Evolutionary Computation*, T. Baeck, D. . Fogel, and Z. Michalewicz, Eds. Taylor & Francis, 1997.

[104] C. Voudouris and E. P. Tsang, *Guided local search*. Springer, 2003.

[105] C. Voudouris and E. Tsang, “Partial constraint satisfaction problems and guided local search,” in *Practical Application of Constraint Technology*, 1996, pp. 337–56.

[106] C. Voudouris and E. Tsang, “Guided local search and its application to the traveling salesman problem,” *Oper. Res.*, vol. 113, pp. 469–499, 1999.

[107] P. Kilby, P. Prosser, and P. Shaw, “Guided local search for the vehicle routing problem with time windows,” in *Meta-heuristics advances and trends in local search paradigms for optimization*, 1999, pp. 473–486.

[108] E. Tsang and C. Voudouris, “Fast local search and guided local search and their application to British telecom’s workforce scheduling problem,” *Oper. Res. Lett.*, vol. 20, pp. 119–127, 1997.

[109] Q. Zhang, J. Sun, and E. Tsang, “An evolutionary algorithm with guided mutation for the maximum clique problem,” *Evol. Comput. IEEE Trans.*, vol. 9, no. 2, pp. 192–200, 2005.

[110] S. Baluja, ““Population-based incremental learning: a method for integrating genetic search based function optimzation and competitive learning,” 1994.

[111] B. A. Tolson and C. A. Shoemaker, “Dynamically dimensioned search algorithm for computationally efficient watershed model calibration,” *Water Resour. Res.*, vol. 43, no. 1, pp. 208–214, 2007.

[112] M. S. Gibbs, H. R. Maier, and G. C. Dandy, “Using characteristics of the optimisation problem to determine the genetic algorithm population size when the number of evaluations is limited,” *Environ. Model. Softw.*, vol. 69, pp. 226–239, 2015.

[113] B. A. Tolson, M. Asadzadeh, H. R. Maier, and A. Zecchin, “Hybrid discrete dynamically dimensioned search (HD-DDS) algorithm for water distribution system design optimization,” *Water Resour. Res.*, vol. 45, no. 12, 2009.

[114] A. Khedr and B. A. Tolson, “Comparing optimization techniques with an engineering judgment approach to WDN design,” *Water Resour. Plan. Manag.*, vol. 142, no. 5, 2015.

[115] H. R. Maier, Z. Kapelan, J. Kasprzyk, J. B. Kollat, and L. S. Matott, “Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions,” *Environ. Model. Softw.*, vol. 62, pp. 271–299, 2014.

[116] B. A. Tolson, A. Khedr, and M. Asadzadeh, “The battle of the water networks (BWN-II): PADDS based solution approach,” in *14th Water*

Distribution Systems Analysis Symp., 2012.

[117] Y. S. Ong and A. J. Keane, “A domain knowledge based search advisor for design problem solving environments,” *Eng. Appl. Artif. Intell.*, vol. 15, no. 1, pp. 105–116, 2002.

[118] S. M. Sapuan, “A knowledge-based system for materials selection in mechanical engineering design,” *Mater. Des.*, vol. 22, no. 8, pp. 687–695, 2001.

[119] E. Keedwell and S.-T. Khu, “A novel evolutionary meta-heuristic for the multi-objective optimization of real-world water distribution networks,” *Eng. Optim.*, vol. 38, no. 3, pp. 319–333, 2006.

[120] F. Zheng, A. R. Simpson, and A. Zecchin, “A combined NLP-differential evolution algorithm approach for the optimization of looped water distribution systems,” *Water Resour. Res.*, vol. 47, no. 8, pp. 2924–2930, 2011.

[121] D. Kang and K. E. Lansey, “Revisiting optimal water-distribution system design: Issues and a heuristic hierarchical approach,” *Water Resour. Plan. Manag.*, vol. 138, no. 3, pp. 208–217, 2012.

[122] W. Bi, G. C. Dandy, and H. R. Maier, “Improved genetic algorithm optimization of water distribution system design by incorporating domain knowledge,” *Environ. Model. Softw.*, vol. 69, pp. 370–381, 2015.

[123] T. M. Walski, “The wrong paradigm—Why water distribution optimization doesn’t work,” *Water Resour. Plan. Manag.*, vol. 127, no. 4, pp. 203–205, 2001.

[124] A. Singh, B. S. Minsker, and A. J. Valocchi, “An interactive multi-objective optimization framework for groundwater inverse modeling,” *Adv. Water Resour.*, vol. 31, no. 10, pp. 1269–1283, Oct. 2008.

[125] H. Takagi, “Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation,” *Proc. IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.

[126] W. Bi, G. C. Dandy, and H. R. Maier, “Use of Domain Knowledge to Increase the Convergence Rate of Evolutionary Algorithms for Optimizing the Cost and Resilience of Water Distribution Systems,” *Water Resour. Plan. Manag.*, vol. 142, no. 9, 2016.

[127] E. Alperovits and U. Shamir, “Design of optimal water distribution systems,” *Water Resour. Res.*, vol. 13, no. 6, p. 885, 1977.

[128] D. Whitley and J. Kauth, “GENITOR: A different genetic algorithm,” in *Rocky Mountain Conference on Artificial Intelligence*, 1988, pp. 118–130.

[129] L. Whitley, “The GENITOR Algorithm and Selection Pressure,” in *Third International Conference on Genetic Algorithms*, 1989, pp. 116–121.

[130] G. Syswerda, “A study of reproduction in generational and steady-state genetic algorithms,” in *Foundation of Genetic Algorithms*, G. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 94–101.

[131] K. A. De Jong, “Analysis of the behavior of a class of genetic adaptive systems,” University of Michigan, 1975.

[132] D. Goldberg and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms,” in *Foundation of Genetic Algorithms*, G.

Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 69–93.

[133] A. El-Bahrawy and A. Smith, “Application of MINOS to water collection and distribution networks,” *Civ. Eng. Syst.*, vol. 2, no. 1, pp. 38–49, 1985.

[134] N. Duan, L. Mays, and K. Lansey, “Optimal reliability-based design of pumping and distribution systems,” *J. Hydraul. Eng.*, vol. 116, no. 2, pp. 249–268, 1990.

[135] S. Forrest, “Documentation for prisoner’s dilemma and norms programs that use the genetic algorithm,” 1985.

[136] J. E. Baker, “Adaptive selection methods for genetic algorithms,” in *International Conference on Genetic Algorithms and Their Applications*, 1985, pp. 100–111.

[137] J. Wakunda and A. Zell, “Median-selection for parallel steady-state evolution strategies,” in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 2000, pp. 405–414.

[138] M. Atiquzzaman, S. Y. Liong, and X. Yu, “Alternative decision making in water distribution network with NSGA-II,” *J. water Resour. Plan. Manag.*, vol. 132, no. 2, pp. 122–126, 2006.

[139] C. Bragalli, C. D’Ambrosio, and J. Lee, “On the optimal design of water distribution networks: a practical MINLP approach,” *Optim. Eng.*, vol. 13, no. 2, pp. 219–246, 2012.

[140] O. Fujiwara and D. Khang, “A two-phase decomposition method for optimal design of looped water distribution networks,” *Water Resour. Res.*, vol. 26, no. 4, pp. 539–549, 1990.

[141] E. Keedwell and S.-T. Khu, “A hybrid genetic algorithm for the design of water distribution networks,” *Eng. Appl. Artif. Intell.*, vol. 18, no. 4, pp. 461–472, Jun. 2005.

[142] Á. Eiben, R. Hinterding, and Z. Michalewics, “Parameter control in evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, 1999.

[143] A. E. Eiben, M. C. Schut, and A. R. de Wilde, “Boosting Genetic Algorithms with Self-Adaptive Selection,” in *IEEE International Conference on Evolutionary Computation*, 2006, pp. 477–482.

[144] R. A. Fisher, “Theory of Statistical Estimation,” *Proc. Camb. Philol. Soc.*, vol. 22, pp. 700–725, 1925.

[145] M. Johns, E. C. Keedwell, D. Savic, “Multi-objective Pipe Smoothing Genetic Algorithm for Water Distribution Network Design,” in *11th International Conference on Hydroinformatics*, 2014.

[146] L. A. Rossman, *EPANET users manual*. United States Environmental Protection Agency, 2000.

[147] E. Zitzler and L. Thiele, “Multiobjective optimization using evolutionary algorithms—a comparative case study,” *Parallel Probl. solving from nature—PPSN V*, 1998.

[148] E. Bader, J., Deb, K., & Zitzler, “Faster Hypervolume-based Search using Monte Carlo Sampling,” in *Conference on Multiple Criteria Decision Making (MCDM 2008)*, 2008, pp. 313–326.

[149] M. Johns, E. C. Keedwell, D. Savic, “Interactive 3D Visualisation of

Optimisation for Water Distribution Systems," in *11th International Conference on Hydroinformatics*, 2014.

[150] C. D. Wickens, J. D. Lee, Y. D. Liu, and S. Gordon-Becker, *Introduction to Human Factors Engineering*. Pearson; 2 edition, 2003.