JUMPING GENES FOR WATER DISTRIBUTION SYSTEM OPTIMIZATION

D.N. Raad¹, A.N. Sinske² and J.H. van Vuuren¹

¹Department of Logistics, University of Stellenbosch, Private Bag X1, Matieland, 7602, South Africa, emails: darianr@sun.ac.za, vuuren@sun.ac.za

²GLS Software (Pty) Ltd., PO Box 814, Technopark, Stellenbosch, 7599, South Africa, email: alex@gls.co.za

Abstract

The design of a water distribution system (WDS) is a challenging problem involving multiple conflicting objectives. The goal of multi-objective WDS design optimization is to find the set of solutions which embodies the Pareto-optimal trade-off between system cost and benefit (typically some measure of reliability). This provides decision-makers with a broad understanding of design possibilities, allowing them to identify a solution with the maximum cost-benefit, or maximize benefit subject to a given budget. A WDS must be designed in order to satisfy consumer demand at required pressures, handle multiple demand loading conditions, withstand component failure and degradation, and provide surplus capacity for growth. In this paper, a jumping-gene (JG) evolutionary algorithm is applied to multi-objective WDS design, and tested on a number of established WDS benchmarks. The JG algorithm is compared to the Non-dominating Sorting Genetic Algorithm II (NSGA-II). Results indicate that the JG algorithm is comparable to the NSGA-II, yet sufficiently different to warrant consideration, particularly in terms of finding high reliability looped configurations. It is argued that the JG mechanism is capable of mimicking how an engineer might exchange capacity between various parts of a system in the course of the design process, and as such is able to find robust alternative designs which may elude competing algorithms.

1. INTRODUCTION

Water distribution system (WDS) design optimization is a challenging research field. From the first computerized attempt (Schaake and Lai, 1969) to advanced modern meta-heuristics (Afshar et al., 2007; Kapelan et al., 2005; Keedwell and Khu, 2006; Vrught and Robinson, 2007; Wu and Simpson, 2001; Zecchin et al., 2007), it has witnessed remarkable developments in optimization theory. In its simplest form, WDS design is posed as the combinatorial optimization problem of selecting pipe sizes for a fixed layout water network from a discrete set of commercially available pipe diameters, in order to minimize cost. More complex forms of the problem may include the selection or sizing of reservoirs, tanks, valves, and pumps, the staged construction or rehabilitation of an existing WDS, minimizing running costs over the system lifetime, including energy and maintenance costs, the consideration of multiple conflicting objectives such as cost and reliability, WDS component placement, the consideration of uncertainty in demand and other data, water quality objectives, and the design of the physical network layout. The basic components of a WDS are shown in Figure 1(a) and a real WDS case study in Figure 1(b).

WDS design was originally approached from the perspective of cost minimization subject to demand and pressure constraints. However, researchers soon came to realize that this paradigm is inadequate, owing to the lack of robustness and flexibility in the resulting least-cost designs (Walski, 2001). Numerous measures of WDS reliability have subsequently been investigated, including techniques which constrain reliability to minimum levels (Goulter et al., 2000). During the last decade, a number of researchers have
pursued a multi-objective (MO) design approach, emphasizing the trade-off between the cost and the various benefits of a WDS, such as hydraulic reliability, surplus energy, and water quality (Walski, 2003).

Water distribution system (WDS) design by artificial evolution has been progressing for more than two decades. Genetic algorithms (GAs) are stochastic search heuristics inspired by Darwinian natural selection and population genetics, first proposed in (Holland, 1975). Recent success in multi-objective WDS design has been achieved through the use of multi-objective evolutionary algorithms (MOEAs), such as the Non-dominated Sorting Algorithm II (NSGA-II) (Kapelan et al., 2005). MOEAs attempt to find an approximation to the Pareto-Optimal set of a MO problem in objective function space, starting from an initial population of solutions and employing evolutionary search techniques. Solutions to the problem are encoded by means of genetic strings (or chromosomes), the genes of which may be mapped directly to values of decision variables. These solutions compete on the basis of a fitness function which is some measure of Pareto-dominance in multi-objective space.

The jumping gene (JG) paradigm for MOEAs was examined in a recent study (Chan et al., 2007) which demonstrated the superiority of a jumping gene enabled MOEA over six other popular MOEAs for the majority of 13 common benchmark problems. Jumping gene transposition is a stochastic operation whereby gene substrings move from one location to another within chromosomes. In this paper, the JG operator is applied within the framework of the NSGA-II.
2. A BRIEF HISTORY OF WDS DESIGN WITH EVOLUTIONARY ALGORITHMS

Early attempts at WDS design optimization included the use of linear (Alperovits and Shamir, 1977; Quindry et al., 1981) and non-linear programming (Lansey and Mays, 1989), dynamic programming (Schaake and Lai, 1969), partial enumeration (Gessler, 1985), as well as other design heuristics (Monbaliu, 1990). These methods suffered from one or more drawbacks such as over-simplification, unrealistic output solutions (split-pipes or continuous pipe sizes), lack of redundancy (no loops), and premature convergence to local optima (Lansey, 2000). Researchers therefore gradually turned to stochastic search techniques for water network design, such as genetic algorithms (Goldberg, 1989), simulated annealing (Cunha and Sousa, 1999), and tabu search (Fanni et al., 2000), which are more efficient and robust, and able to represent discrete pipe diameter variables explicitly.

The standard genetic algorithm (GA) is a meta-heuristic which mimics natural evolution to generate a population of ‘healthy’ candidate solutions to a problem. It employs binary-coded chromosomes representing the decision variables, and the three stochastic genetic operators of mutation, reproduction, and crossover. Mutation typically occurs bitwise by flipping a bit with low probability. Reproduction promotes survival of the fittest through a selection mechanism which favours solutions of higher fitness. Crossover occurs when genetic information is exchanged between two individual solutions. This may be done by means of a single-point crossover, whereby two strings are cut at a random point along their length, and the tail ends exchanged to produce two new solutions (Goldberg, 1989).

The first study to apply genetic algorithms to the problem of gas pipeline optimization was that of (Goldberg, 1983). It was later demonstrated that GAs are able to find near-optimal solutions to pipeline operation problems after examining only a small fraction of the solution space (Goldberg and Kuo, 1987). A comparative study was conducted (Simpson et al., 1994) of the standard GA versus complete enumeration and nonlinear programming (using the generalized reduced gradient method). Enumeration was dismissed as a feasible possibility. Nonlinear programming was faster than the GA, but yielded only a single solution with continuous pipe sizes which had to be rounded up or down. While it may be feasible for small problems, there is no guarantee that the optimum will be found for large real-world systems. The GA was robust, finding the global optimum in 8 out of 10 runs, and yielded a large number of alternatives near to optimum. Another study used an improved genetic algorithm (Dandy et al., 1996) with variable power scaling of the fitness function and grey coding of genetic strings to avoid large Hamming distances (as opposed to binary coding). This GA outperformed the simple GA and other techniques (dynamic-, linear- and nonlinear programming), and was able to find an improved solution to a famous WDS benchmark.

Halhal et al. (1997) was the first study to consider a multi-objective GA approach to address the problem of WDS rehabilitation under a limited budget. They accommodated the goals of maximizing benefit and minimizing cost, using the concepts of Pareto rank and fitness sharing (Goldberg, 1989). They used the notion of incremental solution building in developing a structured messy GA (SMGA), a method which exploits the fact that for a fixed budget, only a small number of the total possible rehabilitation options may be implemented. Whereas the conventional GA represents all decision variables in a chromosome - even if many are inactive - solution representation in SMGA begins with a single decision variable (rehabilitation option), and incrementally builds longer genetic strings up to a maximum number of rehabilitation options, effectively pruning the search space enormously. Their technique was ground-breaking and is laudable for its dramatic improvements in efficiency and utility. However, their initialization procedure requires a complete enumeration of all single-option solutions, which may not be scalable for large systems. Furthermore, their benefit function is difficult to work with as it requires the specification of weights between a number of incommensurate system properties. Finally, the fitness sharing technique requires user-specified cost neighbourhoods. Another study in 1997 (Savic and Walters) used a standard GA integrated with the EPANet hydraulic solver (Rossman, 2000) which they tested on 3
WDS benchmarks from the literature. They found that the results were sensitive to Hazen-Williams head-loss coefficients of the pipes. The SMGA technique was augmented (Walters et al., 1999) to enable the selection of pumps and tanks, including the location of the latter at any node and inclusion of new operational constraints for tank water cycles. This was applied to create improved designs for the renowned “Anytown” benchmark (Walski et al., 1987). The SMGA has the structural restriction of recombining chromosomes of the same length only. Wu and Simpson (2001) demonstrated that this is unnecessary, applying the full version of the fast messy GA (Goldberg et al., 1993) to least-cost WDS design. The fast messy GA employs probabilistically complete initialization and explicit building-block (or schemata) filtering and juxtaposition to find good gene combinations very quickly. They demonstrated dramatically improved performance for the design of a real-world Moroccan WDS. The disadvantage of their method is that it is not formulated for multi-objective design. As opposed to the SMGA it can handle completely new designs. Tolson et al. (2004) combined the first-order reliability method (FORM) (Xu and Goulter, 1999) with a basic GA to find Pareto-optimal solutions for WDS design by means of goal-programming (numerous single-objective optimization problems are solved for different reliability goals).

This method may work for small problems, but is computationally expensive as it requires intensive calculation of derivatives and matrix inversions. Babayan et al. (2005) responded with a more effective integration-based methodology designed to accommodate uncertainty in demand inputs, which is still limited by its single-objective formulation. Farmani et al. (2003) compared four MOEAs for WDS design optimization and concluded that NSGA-II (Deb et al., 2002) was the best. Prasad and Park (2004) employed the NSGA for multi-objective optimization using objectives of cost and a novel surrogate measure of reliability called network resilience, designed to reward reliable loops in the network explicitly. They found that this produced more robust designs than previous methods. Farmani et al. (2005a) compared the NSGA-II to the SPEA-II algorithm on three WDS benchmarks, and found that the latter produced improved solution quality (at the cost of increased running time).

They applied these algorithms to the large Exeter WDS benchmark with three objectives, and concluded that while both algorithms were somewhat successful, further research was needed in locating better Pareto-optimal sets, particularly in high-dimensional objective spaces. In another study (Farmani et al., 2005b) they applied the NSGA-II to the multi-objective design of the Anytown WDS, which includes the design and placement of tanks, using a surrogate reliability measure called the resilience index. Kapelan et al. (2005) implemented an adapted robust version of the NSGA-II algorithm (RNSGA-II) which uses reduced sampling fitness evaluation (requiring fewer hydraulic simulations) to solve the stochastic WDS design problem with the objectives of minimizing cost and maximizing probabilistic hydraulic reliability. They employed this approach to solve the famous NYTUN problem in a multi-objective fashion. Keedwell and Khu (2006) developed a cellular automata approach towards initializing GA searches for WDS optimization with healthy individuals (feasible or near feasible) as opposed to using random initial configurations, which they called the Cellular Automaton Network Design Algorithm (CANDA). The algorithm iteratively evaluates the pressure deficit or excess at every demand node based on a target pressure. Incoming pipes are downsized or upsized depending on whether a node experiences a pressure deficit or pressure excess. These changes are implemented for every node before the next hydraulic simulation is conducted. It was demonstrated that this method converges rapidly to semi-realistic configurations, but was of limited use in further refining designs (Keedwell and Khu, 2006).

When using GAs, constraints are typically incorporated by means of a penalty term added to the basic cost of a solution. This requires an appropriate penalty factor to scale the magnitude of a constraint violation to that of the basic cost, ideally such that an infeasible solution is always more expensive than a feasible one, thereby applying enough pressure to find feasible solutions. Traditionally this factor was found by trial and error. Wu and Simpson (2002) developed a method to adapt the penalty factor automatically during the course of optimization.
A simpler and more effective method was developed by Afshar et al. (2007) based on multiplying the penalty factor by the ratio of the most expensive feasible solution to the most expensive infeasible solution in every generation. They used this scheme to improve search efficiency in the design optimization of several WDS benchmarks. A possible alternative to penalty factors is to conduct tournament selection using constraint violation magnitudes.

3. MODEL FORMULATION FOR WDS DESIGN OPTIMIZATION

The multi-objective WDS design optimization problem for a system comprising \( p \) pipes and \( n \) nodes may be expressed as that of finding an approximation to the Pareto-optimal solution set with objectives of

\[
\text{minimizing } C = C(h, d, q) \quad \text{[Cost]}
\]
\[
\text{and maximizing } R = R(h, d, q) \quad \text{[Reliability]}
\]

subject to
\[
g(q, d) = 0, \quad \text{[Hydraulic]}
\]
\[
h_0 \leq h \leq h_1, \quad \text{[Pressure]}
\]
\[
d = [d_1, d_2, \ldots, d_p], d_i \in D, \quad \text{[Design]}
\]
\[
w_0 \leq w(h, d, q) \leq w_i \quad \text{[Other]}
\]

where \( d \) is a \( 1 \times p \) vector of decision variables representing the pipe diameters, where \( h \) is a \( 1 \times n \) vector of computed nodal pressure heads, where \( C \) denotes the cost of the network as a function of pipe diameters, nodal heads and flow \( q \), and where \( R \) denotes some reliability measure. The hydraulic constraints \( g(q, d) = 0 \) ensure continuity of flow and zero head loss around loops. These may be satisfied intrinsically by means of a hydraulic solver which is called to evaluate the flows and pressures for every network configuration. EPANet2 (Rossman, 2000) is employed in this paper. The nodal pressure constraints specify a vector of minimum heads \( h_0 \), which ensure a minimum customer service level, and a vector of maximum heads \( h_1 \), which guards against leaks and component damage. The design constraints specify that every decision variable takes on a value from the discrete set \( D = \{d^1, \ldots, d^m\} \) where \( d^i \) is the \( i \)-th available pipe diameter. Other constraints may include upper limits on pipe velocity. A penalized cost function of the form \( C = C(h, v, d) = C_d(d) + P(h, v) \) is used, where \( C_d(d) \) is the capital cost and \( P(h, v) \) is a penalty term as a function of demand node pressure heads and absolute pipe velocities \( v \). The penalty term incorporated in this paper takes the form

\[
P = \alpha_p \left[ \sum_{j=1}^{k} \left( \frac{v_j - v_{\min}}{v_{\max} - v_{\min}} - 1 \right) [A] + \sum_{i=1}^{n} \left( - \frac{h_i - h_{\min}}{h_{\max} - h_{\min}} \right) [B] + \sum_{i=1}^{n} \left( - \frac{h_i - h_{\min}}{h_{\max} - h_{\min}} - 1 \right) [C] \right]
\]

where \( \alpha_p \) is the penalty factor, terms \( [A] \) are included where \( v_j > v_{\max} \), terms \( [B] \) are included if \( h_i < h_{\min} \), and terms \( [C] \) are included if \( h_i > h_{\max} \). This has been designed to penalize constraint violations normalized by the size of the feasible range, which enables more meaningful aggregation of minimum and maximum constraint violations. Such a penalty term may be used directly as an indication of feasibility, since it is zero when all velocity and head values are in their feasible ranges. In this paper it was decided to take \( v_{\min} = 0, v_{\max} = 2 \text{ m/s}, \) and \( h_{\min} = 25, h_{\max} = 95 \text{ m}, \) in accordance with common
design practise (GLS Software (Pty) Ltd., 2008). Accurate estimation of the hydraulic reliability of a WDS requires Monte-Carlo simulation in order to calculate the probability that customer requirements are met for the simulation duration throughout the network. This is a computationally intensive process, and is avoided here by employing a surrogate reliability measure called network resilience \( I_n \) attributed to Prasad and Park (2004) as an enhancement of the resilience index measure (Todini, 2000). Here \( I_n \) is an indicator of excess system power (energy per unit time) and explicitly rewards reliable loops of similarly sized pipes. Internal energy losses increase when demand increases or pipe failure occurs; therefore it is desirable to provide surplus power at each node. The surplus power at any node \( i \) is given by 
\[
P_i = \gamma q_i (h_i - h_i^*)
\]
where \( q_i \) is the demand (flow) at the node, \( h_i \) is the nodal head, \( h_i^* \) is the minimum nodal head, and \( \gamma \) is the specific weight of water. A loop may be considered reliable if the pipes incident with a node do not vary widely in diameter. If \( d_1 > d_2 > d_3 \) are the diameters of three pipes incident with node \( i \), then the uniformity of that node is given by 
\[
C_i = \left( \sum_{j=1}^{p_i} d_j \right) \left( p_i \times \max\{d_1, \ldots, d_p\} \right)
\]
where \( p_i \) is the number of pipes incident with node \( i \). Note that \( C_i = 1 \) if pipes incident with a node all have the same diameter, while \( C_i < 1 \) if pipes incident with a node have different diameters. For nodes incident with only one pipe, the value of \( C_i \) is taken to be one. The combined effect of both surplus power and connecting pipe uniformity of node \( i \), called weighted surplus power, is expressed as 
\[
X_i = C_i P_i.
\]
For the network as a whole, it is given by 
\[
X = \sum_{i=1}^{n} X_i.
\]
This may be normalized by dividing with maximum surplus power to obtain network resilience as
\[
I_n = \frac{X}{X_{\text{max}}} = \frac{\sum_{i=1}^{n} C_i q_i^* (h_i - h_i^*)}{\sum_{k=1}^{n} Q_k H_k^* - \sum_{i=1}^{n} q_i^* h_i^*}
\]
where \( X_{\text{max}} \) is the maximum surplus power of the system, where \( Q_k \) is the flow and \( H_k \) is the pressure head supplied by the \( k \)-th reservoir, and \( q_i^* \) is the demand at the \( i \)-th node (Prasad and Park, 2004).

4. **MOO, MOEAS AND JUMPING GENES**

Multi-objective optimization (MOO) yields a set of compromised solutions, which should approximate the true Pareto-optimal solution set. These solutions are non-dominated in the sense that no other solution is better with respect to all objective function values, and moving from one solution to another results in the improvement of one objective but the degradation of another. A solution \( x^1 \) is said to dominate an inferior solution \( x^2 \) (denoted by \( x^1 \succ x^2 \)) if \( x^1 \) is better in terms of at least one objective function value (for objective function \( f(x) = y = (y_1, y_2) \); there is an \( i \in [1,2] \), \( y_i^1 > y_i^2 \)) and no worse in terms of all other objectives (for \( j \neq i \), \( y_i^1 \geq y_i^2 \)) (Zitzler et al., 2003).
The curve in objective space which can be drawn through the Pareto-optimal (PO) solutions is commonly known as the Pareto-optimal front. Several sub-fronts may exist into which a set of solutions may be classified, each lower front being dominated by the one above it. Dominance may be characterized in a number of different ways, such as the count of solutions dominated by or which dominate an individual, or the rank of the sub-front to which a solution belongs (the PO front has rank 1). MOEAs are evolutionary algorithms designed to find an approximation to the full PO set. They employ a population of solutions evolving according to typical genetic operations, with a selection scheme based on Pareto-dominance. There may exist many different PO set approximations, and different algorithms may produce sets of differing quality in terms of their closeness to the true Pareto optimal front, and their diversity along it, for which various quality metrics exist (Zitzler et al., 2003).

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a well-known MOEA (Deb et al., 2002), and has proven highly successful for a broad range of optimization problems. NSGA-II employs a fast non-dominated sorting procedure to sort the solutions of a population into their non-domination ranks (front depths). Solutions of identical rank are distinguished by means of a crowding distance metric which quantifies how isolated they are in objective space, with a larger crowding distance being more desirable. A detailed description of NSGA-II appears in (Deb et al., 2002). In all algorithms used in this paper, offspring creation occurs by means of binary tournament selection, single-point genetic crossover, and component-wise mutation with a probability of 0.01, using a binomial distribution for the number of mutation steps away from the current gene value. In a binary tournament, two solutions are selected at random and compete using a crowded comparison operator. This operator selects the fitter of the two solutions firstly as the one with the lower Pareto-rank, or, provided the ranks are equal, selects the one with the largest crowding distance. Furthermore a WDS network is encoded by means of an array of integer genes (genome), each of which represents an option for a particular component (such as a pipe diameter), and may be used directly to index such options.

The jumping gene phenomenon was discovered by Barbara McClintock (1950) during the course of her work on maize genetics. The jumping gene, also known as a transposon, is a gene or subset of gene elements in a chromosome, which undergoes a transposition (movement or copy action) within the chromosome or between two different chromosomes. This movement of genetic information may be likened to a rearrangement of existing genetic capacity. A transposon which contributes positively in its current location may prove even more beneficial in another location, improving chromosome fitness. When genetic information is transferred between chromosomes it is known as horizontal transmission (as opposed to vertical transmission from parents to offspring), and may even occur inter-species. JG transposition is complex in nature, including non-transposable elements, transposition agents, and autonomous / non-autonomous transposons. For a more detailed coverage of the biological aspects the reader may consult the work of Chan et al. (2007), and the original work of McClintock (1950). The transposition of genetic material translates to WDS design in that a rearrangement of existing pipe capacity may enable more efficient water delivery.

For example, a large size for a pipe supplying water from a first reservoir may be more effective when assigned to a pipe supplying water from a second reservoir. This is intuitively agreeable with the notion that an engineer might rearrange WDS capacity during the course of a design exercise. It is worthwhile noting that if genetic operations are limited to mutation and crossover alone, a population of solutions which is dominated by certain high-frequency alleles (specific values for certain genes, such as the large pipe at the first reservoir), may struggle to move away from the local optima which have these alleles as building blocks. Crossover is unable to produce the required swap of genetic information, and the swapped configuration is highly unlikely to be found by mutation. The JG operation provides fresh mobility to the usual genetic operations, hopefully enabling more effective exploration of a genetic niche (local subspace of the full search space within which a population is contained). The use of JG in the context of evolutionary computation is a relatively recent phenomenon. The first mention of the operator
occurs in (Costa and Simoes, 1999). A handful of researchers have used JG to enhance the performance of the NSGA-II algorithm, though it may easily be adopted in many MOEAs. Chan et al. (2007) conducted a systematic statistical analysis comparing the JG algorithm to six other MOEAs and found highly favourable results. They concluded that the JG algorithm provided improved convergence and diversity on the majority of 13 constrained and unconstrained benchmark test functions, and that it outperformed its peers for binary and real-coded problems. They demonstrated the dissimilarity of JG from the random mutation operator, and confirmed that it was particularly adept at spreading solutions along the Pareto-Optimal front. Some of the practical applications of JG infused MOEAs have been antenna design (Yeung et al., 2007), the design of heat exchangers (Agarwal and Gupta, 2008), and the optimization of fuel blending (Khosla et al., 2007).

The JG operation may take the form of a *copy-and-paste* action whereby the transposon remains in its original position and is copied to overwrite gene values at a new position. Alternatively, a *cut-and-paste* action is when the transposon is removed from its initial position, with the chromosome genes shifting to take its place. It is then inserted at a new position in-between existing genes (*i.e.* no overwrite occurs). The copy-and-paste operation is illustrated in Figure 2, applied *a)* in the same chromosome and *b)* between different chromosomes. Similarly, the cut-and-paste operation is shown in Figure 3.

![Figure 2: Copy-and-paste transposition operation. (a) Same chromosome. (b) Different chromosome.](image1)

![Figure 3. Cut-and-paste transposition operation. (a) Same chromosome. (b) Different chromosome](image2)

Transposons may consist of a few bits, a whole gene, or may span several genes. There is no planning in these JG transpositions (as is true of the natural phenomenon); the selection of the transposon and insertion / paste positions is random and opportunistic. Furthermore, in order for it to be beneficial, JG must occur with a low frequency, otherwise it may lead to excessive *genetic drift* resulting in overall fitness degradation and possible extinction. A careful balance must be maintained, because gene transposition has a higher probability of degrading fitness than finding a better solution. It is recommended that the jumping gene operator be inserted after selection but prior to crossover and mutation for best performance (Chan et al., 2007). A jumping rate of 0.04 (4 out of every 100 transposons) was employed, which is also adopted here. In this paper only a single gene (representing a single
component) is allowed to jump at a time, although a genome may undergo multiple jumps, each with a small probability. To the best knowledge of the authors, this is the first study investigating the use of the JG algorithm for WDS design optimization. The JG algorithm is presented in detail by Chan et al. (2007).

5. JUMPING AROUND: PERFORMANCE EVALUATION FOR 4 WDS BENCHMARKS

Competing genetic algorithms are frequently compared by running several trials (e.g. 20 - 50) on multiple benchmark tests for an equal number of generations. However, it should be noted that computation per evolutionary generation may differ substantially between different algorithms, and that there is a time trade-off for the extra genetic operation performed by the JG algorithm. Therefore, it is advised to either use a common time limit, or execute the algorithms until they fail to produce improvement above some threshold percentage (e.g. 1% of the population size), in which case running times must be reported. The first method focuses more on computational efficiency, and the second on solution quality and long-term solution discovery, although both can be achieved with a large enough time limit. The first method is used here because of its simplicity. All time limits were established by trial and error.

The algorithms were tested using three documented WDS benchmark problems and a real case study. The benchmarks include the New York Tunnel problem (Schaake and Lai, 1969), the Two-loop Network (Alperovits and Shamir, 1977), and the Hanoi Network (Fujiwara and Khang, 1990). The real case study is a new WDS development project in Ekurhuleni (East Rand, Gauteng Province, South Africa) called the R21 Corridor development area (GLS Software (Pty) Ltd., 2008). This system is large relative to the other benchmarks, consisting of 87 pipes. Further information is available from the authors on request. The benchmarks will be referenced by the shorthand NYTUN, TLN, HANOI and R21 respectively. For each benchmark, a set of thirty different random initial populations was provided. These initial populations gave rise to thirty optimization runs of each algorithm for each benchmark. A population size of 100 individuals was enforced throughout. The penalty factors for the benchmarks were determined experimentally with the assistance of an empirical equation, \( \alpha_p = (C_{max} - C_{min})/(0.01 \times n) \), where \( n \) is the number of demand nodes and \( C_{max}, C_{min} \) are the minimum and maximum possible cost of a network.

This equation ensures that a head deficit greater than 0.01 m at all the nodes will result in a configuration which is more expensive than the most expensive system. The penalty factor values used were \( \alpha_p = 10^9 \) for NYTUN and R21, \( \alpha_p = 3 \times 10^7 \) for HANOI, and \( \alpha_p = 5 \times 10^7 \) for TLN. The time limits involved for the various benchmarks were 2 minutes per optimization run for TLN, 3 minutes per run for NYTUN, 4 minutes per run for HANOI, and 30 minutes per run for R21.

Two performance evaluation metrics were applied to the test results, as recommended by Knowles et al. (2006). The first metric is the dominance count rank, which is simply one plus the number of approximation sets which weakly dominates a given set (each objective vector is either matched or dominated), such that a lower rank is more desirable. An algorithm with a lower average rank can ordinarily be classified as superior to another (Knowles et al., 2006). There was no significant difference between algorithms in terms of this metric for any of the benchmarks. The hypervolume metric is a more fine-grained analysis which measures the total hypervolume of the objective space dominated by a given approximation set relative to some reference point, such that a higher value is more desirable (Zitzler et al., 2003). This metric has the advantage of representing both closeness to the true Pareto-front and solution diversity. Normalized average hypervolumes and their standard deviations over thirty optimization runs for each of the benchmarks are presented in Table 1. The JG algorithm outperforms NSGA-II in terms of hypervolume for the NYTUN and R21 benchmarks, is similar for the TLN
benchmark, and worse for the HANOI benchmark. Thus, there is no clear winner between the two and further research with additional benchmarks may be required.

The reason for the inferior performance in the HANOI benchmark is that the JG algorithm was less adept than the NSGA-II at finding low cost branched designs (true for all benchmarks). In contrast, the JG algorithm was better at finding high cost, high reliability looped configurations. In order to fulfil their functions, water systems tend naturally to have large pipes near the sources with sizes decreasing towards the nodes furthermost from the sources. Therefore, in the case of a pure branched network, a transposition of pipe size from further along the branch to nearer the source and vice versa is likely to cause degradation in system performance. Thus, branched systems may be considered unstable configurations in view of the jumping gene paradigm, and are less likely to propagate successfully. It is therefore argued that JG is better suited to systems with loops and multiple sources. The JG algorithm might possibly be adapted to take this natural ordering of pipe sizes into account by restricting how transposition may occur. The superior performance of JG for R21 case study is also important. Seemingly small differences in results become more valuable as water systems grow in size, because they produce large savings relative to the cost of many other municipal projects. Both algorithms were able to improve significantly on a baseline engineered design of the R21 system which was found by the partial enumeration method (Gessler, 1985). This baseline design has a cost of R 72,100,100 and a network resilience of 0.191356. The new algorithms were able to locate solutions of equivalent resilience for a cost of R 50,000,000 - a phenomenal saving of 30%.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Benchmark</th>
<th>Mean HV</th>
<th>SD HV</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGA-II</td>
<td>NYTUN</td>
<td>0.657</td>
<td>0.016</td>
</tr>
<tr>
<td>JG</td>
<td>NYTUN</td>
<td><strong>0.684</strong></td>
<td>0.016</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>TLN</td>
<td>0.808</td>
<td>0.009</td>
</tr>
<tr>
<td>JG</td>
<td>TLN</td>
<td>0.807</td>
<td>0.004</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>HANOI</td>
<td><strong>0.337</strong></td>
<td>0.002</td>
</tr>
<tr>
<td>JG</td>
<td>HANOI</td>
<td>0.326</td>
<td>0.003</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>R21</td>
<td>0.734</td>
<td>0.034</td>
</tr>
<tr>
<td>JG</td>
<td>R21</td>
<td><strong>0.738</strong></td>
<td>0.026</td>
</tr>
</tbody>
</table>

6. CONCLUSION

In this paper a jumping gene adaptation of the NSGA-II was compared to the ordinary NSGA-II in their ability to optimize WDS designs for the objectives of minimizing cost and maximizing network resilience. Tests were conducted for three well-known WDS benchmarks and one real case study. Neither algorithm outclassed the other in terms of Pareto-dominance, yet the algorithms did reveal different strengths. NSGA-II was found to be better at identifying low cost branched designs, while the JG algorithm was better at finding designs of high reliability and cost. The JG algorithm produced slightly better results for the real case study. It is recommended that further research be conducted using more benchmarks, particularly of large, complex water systems. Potential areas of research could include sensitivity analysis on gene jumping rates, investigating transposons of length greater than one gene, examining the effect of using binary coded chromosomes (with bit strings as transposons) instead of integer coding, and restricting gene jumping to only occur within similar jumping classes (e.g. one class might consists of all pipes within three pipes distance from any source).
7. REFERENCES


