Volume 27 (1), pp. 17–43
http://www.orssa.org.za

ORiON
ISSN 0529-191-X
©2011

Water distribution systems design optimisation using metaheuristics and hyperheuristics

DN Raad[*] A Sinske[†] JH van Vuuren[‡]

Received: 4 February 2011; Revised: 22 May 2011; Accepted: 24 May 2011

Abstract

The topic of multi-objective water distribution systems (WDS) design optimisation using metaheuristics is investigated, comparing numerous modern metaheuristics, including several *multi-objective evolutionary algorithms*, an *estimation of distribution algorithm* and a recent *hyperheuristic* named AMALGAM (an evolutionary framework for the simultaneous incorporation of multiple metaheuristics), in order to determine which approach is most capable with respect to WDS design optimisation. Novel metaheuristics and variants of existing algorithms are developed, for a total of twenty-three algorithms examined. Testing with respect to eight small-to-large-sized WDS benchmarks from the literature reveal that the four top-performing algorithms are mutually non-dominated with respect to the various performance metrics used. These algorithms are NSGA-II, TAMALGAMJ$_{\text{ndu}}$, TAMALGAM$_{\text{ndu}}$ and AMALGAMS$_{\text{ndp}}$ (the last three being novel variants of AMALGAM). However, when these four algorithms are applied to the design of a very large real-world benchmark, the AMALGAM paradigm outperforms NSGA-II convincingly, with AMALGAMS$_{\text{ndp}}$ exhibiting the best performance overall.

Key words: Water distribution systems design optimisation, metaheuristic, hyperheuristic.

1 Introduction

A *water distribution system* (WDS)[1] is a network of components designed to supply treated drinking water to human settlements. *WDS design optimisation* (WDSDO) involves the specification of a layout and the sizing of components for a WDS design, in order to minimise costs and maximise various system benefits. Multi-objective WDSDO has gained in popularity, yielding a Pareto-optimal set of solutions which embody a trade-off between various objectives, and enables more informed decision-making. The design of a WDS is subject to performance constraints, most importantly the satisfaction of consumer water demands within acceptable pressure ranges throughout the system. The

[*]Corresponding author: Department of Logistics, University of Stellenbosch, Private Bag X1, Matieland, 7602, South Africa, email: darianraad@gmail.com

[†]GLS Software (Pty) Ltd, PO Box 814, Technopark, Stellenbosch, 7599, South Africa

[‡]**(Fellow of the Operations Research Society of South Africa)**, Department of Logistics, University of Stellenbosch, Private Bag X1, Matieland, 7602, South Africa

[1]A list of acronyms is provided in the appendix at the end of the paper.

primary WDS components to be sized and placed include pipes, pumps, valves, tanks, and reservoirs. Historically, the WDSDO problem was often simplified to that of *selecting pipe diameters for a fixed layout network* from a discrete set of commercially available options in order to *minimize capital cost*. The WDS pipe-sizing problem is a classical combinatorial optimisation problem, classified as NP-hard. The vast number of permutations of potential WDS designs makes brute force analysis impossible for most real-world WDS cases.

The WDSDO problem may also include the selection of operational modes or settings for components, the staged construction of the network over time, the rehabilitation of existing components (*e.g.* pipe cleaning and relining), and considerations of water quality maximisation and leakage abatement. WDSDO is significantly complicated by the dynamic nature of a WDS, as exhibited by constantly *varying water demands* and the natural growth and aging of the WDS over time. Finally, the optimization problem itself is complicated by the need to conduct *computationally expensive hydraulic simulations* for every candidate design, sometimes for a large number of demand loading conditions, especially if tanks are to be designed [29, 48]. Water demand is estimated from guidelines or historic records and grouped at the network nodes [48].

Apart from cost minimization, the objective of maximizing WDS reliability has received substantial attention in the literature. However, there is still no universally accepted definition for WDS reliability, and numerous different reliability quantification schemes exist [16]. Possibly the most scientific interpretation is *probabilistic hydraulic reliability*, whereby a *Monte Carlo simulation* (MCS) is conducted with stochastic demands, and the proportion of the time that a design solution is hydraulically feasible is taken as the reliability of the WDS [22]. A major problem regarding this methodology is that a full MCS implementation may call for thousands of hydraulic simulations, which becomes impractical as the WDSs grow in size. Alternative techniques for addressing reliability have included analytical approximation techniques, such as the *first order reliability method* (FORM) [53], the incorporation of such techniques within genetic algorithms [43], and integration-based methodologies [3]. These methods all have their limitations, particularly in terms of scalability. As an alternative to probabilistic reliability, some researchers have proposed *reliability surrogate measures*, which measure some desirable property, excess capacity or redundancy of the WDS, in order to quantify how resilient the system might be during stressed conditions. Examples include the *Resilience Index* of Todini [42], dating from 2000, and the *Network Resilience* metric of Prasad and Park [33], dating from 2004.

This paper is focused on multi-objective WDSDO, considering the design of network layouts and the assignment of pipe diameters with respect to a number of WDS benchmarks from the literature. More specifically, metaheuristics and hyperheuristics are used to solve the bi-objective problem of cost minimisation and reliability maximisation (employing the Network Resilience surrogate measure). It was the goal of this study to investigate novel algorithms towards WDSDO, expanding knowledge within the field and striving to identify algorithms which yield improved efficiency and solution quality. For this purpose twenty-three algorithms were selected or developed for comparison, on the basis of representing several different metaheuristic design paradigms (including traditional *multi-*

objective evolutionary algorithms (MOEAs), auto-adapting MOEAs, local-search heuristics, *estimation of distribution algorithms* (EDAs), metaheuristics derived from nature, and hyperheuristics). Of particular note is the hyperheuristic AMALGAM, developed by Vrugt and Robin [45] in 2007, which employs multiple metaheuristics simultaneously in a dynamic fashion in an attempt to improve performance. Several variants of AMALGAM were developed for this study in order to address the shortcomings of the original algorithm. This study updates a 2009 paper, namely Raad *et al.* [34], in which a smaller subset of the metaheuristics and WDS benchmarks were examined, and in which a less rigorous testing methodology was employed.

2 A Brief History of Multi-objective WDSDO

Prior to the late 1990s, WDSDO was focused on single-objective, least-cost design, subject to performance constraints, rarely considering the trade-off between cost and various system benefits. The failure to incorporate numerous important WDS evaluation criteria meant that WDSDO could not become a part of standard engineering practise [47].

The first widely available model that considered multiple objectives was the WADISO program by Walski and Gessler in 1985 [13], which used a *partial enumeration method* to produce solutions, showing the trade-off between cost and minimum pressure. Unfortunately this method quickly becomes impractical as the size of the WDS grows. Halhal *et al.* [17] were arguably the first to popularise multi-objective WDSDO in 1997, using a multi-objective *genetic algorithm* (GA) approach to address the problem of WDS rehabilitation under a limited budget. They accommodated the goals of maximizing benefit (carrying capacity, physical integrity and system flexibility) and minimizing cost, using the concepts of *Pareto rank* and *fitness sharing* introduced by Goldberg in 1989 [14]. They used the idea of *incremental solution building* to develop a *structured messy GA* (SMGA), which incrementally builds longer genetic chromosomes up to a maximum number of rehabilitation options, effectively pruning the search space enormously. Their technique is impressive, but suffers from several drawbacks, including a lack of scalability, and having to specify weighting factors between incommensurate system properties. Also in 1997, Savic and Walters [38] used a standard GA integrated with the EPANET hydraulic solver of Rossman [36], which they tested with respect to three WDS benchmarks from the literature. They found that the results were sensitive to Hazen-Williams head-loss coefficients of the pipes. Halhal *et al.* [17] enforced the structural restriction of recombining chromosomes of the same length only. In 2001, Wu and Simpson [51] demonstrated that this is unnecessary, applying the full version of the *fast messy GA* (FMGA) designed by Goldberg *et al.* [15] to least-cost WDS design. The FMGA 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. In 2004, Tolson *et al.* [43] combined the FORM of Xu and Goulter [53] with a basic GA to find Pareto-optimal solutions for WDS design by means of goal-programming (numerous single-objective problems solved for different reliability goals). This method is computationally expensive and requires intensive calculation of derivatives and matrix inversions.

In 2003, Farmani *et al.* [9] compared four MOEAs for WDSDO and concluded that the *Nondominated Sorting Genetic Algorithm II* (NSGA-II) [5] was the best. A self-adaptive penalty function was incorporated into a multi-objective version of the FMGA by Wu and Walski [50] in 2004, and applied to the optimisation of the Hanoi network. In the same year Nicolini [31] compared three MOEAs (the *Efficient Nondominated GA* (ENGA), NSGA-II, and a controlled elitist NSGA-II) towards the design of the two-loop network introduced by Alperovits and Shamir [2], finding that the latter two outperformed the ENGA. In 2005, Farmani *et al.* [11] compared the NSGA-II to the Strength Pareto Evolutionary Algorithm 2 (SPEA-II) with respect to three WDS benchmarks, and found that SPEA-II produced improved solution quality (at the cost of increased running time). They applied these algorithms to the large Exeter WDS benchmark [8] 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 spaces. In another study [10], they applied the NSGA-II to the multi-objective design of the Anytown WDS [46], which includes the design and placement of tanks, using the Resilience Index as an objective. In 2005, Kapelan *et al.* [22] 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 reliability. They employed RNSGA-II to solve the famous NYTUN problem [39] in a multi-objective fashion.

In 2009, Olsson *et al.* [32] compared three *estimation of distribution algorithms* (EDAs), concluding that the *Univariate Marginal Distribution Algorithm* (UMDA) was superior. Also in 2009, di Pierro *et al.* [7] analyzed two multi-objective hybrid algorithms, namely *ParEGO* [26] and *LEMMO* [21], with respect to the design of a real, medium-sized network in Southern Italy, and a real, large-sized network in the UK. Both algorithms employ machine learning techniques and were designed for dealing with expensive multi-objective optimisation problems, able to operate under a scenario of severely restricted function evaluations. They demonstrated that both hybrid algorithms were capable of dramatic speed enhancements, and although ParEGO was shown to be unsuitable for designing large systems, LEMMO could be successfully extended to the efficient design of large-scale WDSs.

A notable problem with all of these studies is the relatively few WDS benchmarks employed, making it difficult to draw general conclusions about algorithmic performance.

3 Multi-objective WDSDO Model

Consider a WDS system with discrete variables $\boldsymbol{x}^{\rm d}$ (including $n_{\rm p}$ pipe diameters and other discrete component sizes, settings and locations), continuous variables $\boldsymbol{x}^{\rm c}$ (including tank dimensions and valve settings), $n_{\rm r}$ reservoirs and n nodes. A complete solution is therefore $\boldsymbol{x} = \{\boldsymbol{x}^{\rm d}, \boldsymbol{x}^{\rm c}\}$. The multi-objective WDSDO problem for such a system may be expressed as that of finding an approximation to the Pareto-optimal solution set of designs with the objectives of

$$\left.\begin{array}{lll}\text{minimizing} & C = C_{\mathrm{c}}(\boldsymbol{x}) + C_{\mathrm{o}}(\boldsymbol{x},\boldsymbol{d},\boldsymbol{e}), & [\textit{Cost}] \\ \text{minimizing} & \widehat{P} = \widehat{P}(\boldsymbol{d},\boldsymbol{h},\boldsymbol{q},\boldsymbol{o}), & [\textit{Penalty}] \\ \text{maximizing} & R = R(\boldsymbol{x},\boldsymbol{d},\boldsymbol{h},\boldsymbol{q}), & [\textit{Reliability}] \\ \text{maximizing} & \mathcal{K} = \{K_1, K_2, \ldots, K_m\}, & [\textit{Misc}] \\ \text{subject to} & \boldsymbol{x}^{\mathrm{d}} = [x_1^{\mathrm{d}}, x_2^{\mathrm{d}}, \ldots, x_r^{\mathrm{d}}],\ x_i^{\mathrm{d}} \in \mathcal{X}^i, & [\text{Discrete}] \\ & \boldsymbol{x}^{\mathrm{c}} = [x_1^{\mathrm{c}}, x_2^{\mathrm{c}}, \ldots, x_s^{\mathrm{c}}],\ x_{i,\min}^{\mathrm{c}} \le x_i^{\mathrm{c}} \le x_{i,\max}^{\mathrm{c}}, & [\text{Continuous}] \\ & \boldsymbol{g}(\boldsymbol{h},\boldsymbol{q}) = \boldsymbol{0}, & [\text{Hydraulic}] \\ & \boldsymbol{h}(\boldsymbol{d})_{\min} \le \boldsymbol{h}(\boldsymbol{d}) \le \boldsymbol{h}(\boldsymbol{d})_{\max}, & [\text{Pressure}] \\ & \boldsymbol{w}_{\min} \le \boldsymbol{w}(\boldsymbol{x},\boldsymbol{h},\boldsymbol{q},\boldsymbol{d},\boldsymbol{o}) \le \boldsymbol{w}_{\max}, & [\text{Other}] \end{array}\right\} \quad (1)$$

where $\boldsymbol{d}$ denotes the demand loading conditions at the nodes, $\boldsymbol{h}$ is a $1 \times n$ vector of computed nodal pressure heads, $\boldsymbol{q}$ is a $1 \times n_{\mathrm{p}}$ vector of pipe flows, $\boldsymbol{o}$ denotes other computed hydraulic properties such as tank capacity and water level deviations from those required at the end of a cycle. Here C denotes the cost of the network as a function of the decision variables, including the capital investment cost C_{c} and operational costs C_{o} (which may include the present value of energy costs $\boldsymbol{e}$ for pumping as well as maintenance and repair costs). $\widehat{P} = \widehat{P}(\boldsymbol{d},\boldsymbol{h},\boldsymbol{q},\boldsymbol{o})$ is a penalty function which depends on the magnitudes of pressure and other constraint violations. The objective R denotes some reliability measure (such as Network Resilience) as a function of the components, nodal heads and pipe flows, and $\mathcal{K}$ denotes a set of miscellaneous objective functions, such as maximizing water quality.

The hydraulic constraints $\boldsymbol{g} = \boldsymbol{0}$ ensure *continuity of flow* and *zero head loss around loops.* These are hard constraints and may be satisfied intrinsically by means of a hydraulic solver which is called to evaluate the flows and pressures for every network configuration. EPANET2 [36] was employed for this purpose in this paper. The remaining constraints are usually considered *soft constraints*, in that slight violations may still be acceptable. Therefore these constraints are typically handled by means of a penalty function, such as the objective $\widehat{P}$. If the penalty function is zero, then all constraints are satisfied. The nodal pressure constraints specify a vector of minimum heads $\boldsymbol{h}_{\min}$, which ensure a minimum customer service level, and a vector of maximum heads $\boldsymbol{h}_{\max}$, which guards against leakage and component damage. The discrete design constraints specify that every decision variable takes on a value from a discrete set $\mathcal{X}^i = \{x_{\mathrm{o}}^1, \ldots, x_{\mathrm{o}}^{\omega}\}$, where x_{o}^i is the i-th available discrete option. Continuous constraints impose maximum and minimum limits on $\boldsymbol{x}^{\mathrm{c}}$. The other constraints, denoted by $\boldsymbol{w}$, may include upper limits on flow velocity, tank operational constraints, and budgetary constraints. A truly generic formulation of this problem is difficult, owing to the large variety of potential objectives and constraints, and the option to incorporate phased design over time [48].

With conflicting objectives, one is interested in obtaining an approximation to the true Pareto-optimal set of solutions in cost/reliability space. This necessarily invokes the concept of Pareto-dominance, whereby a solution $\boldsymbol{x}$ *dominates* another solution $\boldsymbol{y}$, denoted by $\boldsymbol{x} \succ \boldsymbol{y}$, if $\boldsymbol{x}$ is better in terms of all objective function values than $\boldsymbol{y}$, and two solutions are *nondominated* with respect to each other when each is better than the other for some objective function value. The goal of MOO is to find a good approximation of the Pareto-optimal set. However, this is an idealization; since the true Pareto-set is typically not available in real world problems, it is not possible to test whether the algorithm has attained any Pareto-set members. There may exist many different 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. Performance assessment must necessarily take the form of comparative analysis between approximation sets, for which several indicators exist. A Pareto-approximation set A is said to dominate a set B if for each $y \in B$ there exists an $x \in A$, such that $x \succ y$. Similarly, a set A is said to *weakly dominate* a set B if for each $y \in B$, either $y \in A$ or there exists an $x \in A$, such that $x \succ y$ [57].

In evolutionary optimization it is not normally possible to handle constraints explicitly. A *penalty term* is often added to the basic cost in order to guide the optimization towards feasible solutions. This penalty is based on the magnitude of the constraint violation, multiplied by a *penalty factor* which scales the violation to the same order of magnitude as the cost. Ideally, this should result in an infeasible solution being slightly more expensive than any feasible one. The penalty factor typically requires trial and error fine-tuning, although some researchers have suggested methods for auto-adaptation thereof [1]. An example of such a *penalized cost function* is $C(\boldsymbol{h}, \boldsymbol{v}, \boldsymbol{d}) = C_c(\boldsymbol{d}) + P(\boldsymbol{h}, \boldsymbol{v})$, where $C_c(\boldsymbol{d})$ is the capital cost and $P(\boldsymbol{h}, \boldsymbol{v})$ is a penalty term as a function of nodal pressure heads and pipe velocities $\boldsymbol{v}$. The penalty term incorporated in this paper takes the form

$$\frac{P(\boldsymbol{h}, \boldsymbol{v})}{\alpha_p} = \sum_{j=1}^{p} \frac{v_j - v_{\min}}{v_{\max} - v_{\min}} - 1, \text{ if } v_j > v_{\max} + \sum_{i=1}^{n} \begin{cases} -\frac{h_i - h_{\min}}{h_{\max} - h_{\min}}, & \text{if } h_i < h_{\min} \\ \frac{h_i - h_{\min}}{h_{\max} - h_{\min}} - 1, & \text{if } h_i > h_{\max}, \end{cases} \tag{2}$$

where α_p is the penalty factor. This function has been designed to penalize constraint violations normalized by the size of the feasible range, which enables meaningful aggregation of minimum and maximum constraint violations. Velocity here refers to absolute velocity, since flow may occur in either direction. In this paper it was decided to take $v_{\min} = 0$ and pressure head limits are only enforced at nodes with non-zero demand.

4 Reliability Surrogates

Introduced by Todini [42] in 2000, the *Resilience Index* is an indicator of excess system power. Since internal energy losses will increase when demand increases or pipe failure occurs, it is desirable to provide more hydraulic power at each node in a looped network than is required, so that a sufficient surplus exists to be dissipated internally in case of failures. This surplus is used to characterize the resilience of the network.

If $P_{\text{tot}} = \gamma \sum_{k=1}^{n_r} Q_k H_k$ is the total available power supplied to the system, where Q_k is the flow and H_k is the pressure head supplied by the k^{th} reservoir, and γ is the specific weight of water, then $P_{\text{tot}} = P_{\text{int}} + P_{\text{ext}}$, where P_{int} is the power dissipated in the pipes, while $P_{\text{ext}} = \gamma \sum_{i=1}^{n} q_i h_i$ is the power delivered to users in terms of flow q_i and head h_i at node i, where n and n_{r} denote the number of nodes and the number of reservoirs in the network, respectively. Todini introduced the *Resilience Index* $I_r = 1 - P^*_{\text{int}}/P^*_{\max}$, where $P^*_{\text{int}} = P_{\text{tot}} - \gamma \sum_{i=1}^{n} q^*_i h_i$ is the power dissipated in the network to satisfy the total demand (at any given pressure — typically in excess of that required) and $P^*_{\max} = P_{\text{tot}} - \gamma \sum_{i=1}^{n} q^*_i h^*_i$ is the maximum power available to be dissipated internally if the constraints in terms of both demand and head are satisfied at the nodes. After appropriate substitutions, the Resilience Index may be written as

$$I_r = \frac{\sum_{i=1}^{n} q_i^*(h_i - h_i^*)}{\sum_{k=1}^{n_r} Q_k H_k - \sum_{i=1}^{n} q_i^* h_i^*}. \tag{3}$$

Prasad and Park [33] developed the *Network Resilience* metric in 2004 in response to the above measure by Todini. The advantage of Network Resilience is that it explicitly rewards *reliable loops* of similarly sized pipes by penalizing sudden changes in pipe diameter. The Todini Resilience Index does not explicitly reflect the effects of redundancy. A branched network with sufficient surplus head may therefore still appear desirable, providing insufficient conditions for a reliable network. Prasad and Park demonstrated that using Network Resilience as an objective instead of the Resilience Index produced more robust designs in terms of pipe failure, which is why it was selected for use in this paper.

The notion of Network Resilience incorporates the effects of both surplus power and reliable loops. The surplus power at node i is given by $P_i = \gamma q_i(h_i - h_i^*)$. A loop may be considered reliable if the pipes incident with a node are not widely varying 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 $C_i = (d_1 + d_2 + d_3)/(3d_1)$ and in generalized form as

$$C_i = \frac{\sum_{j=1}^{n_p^i} d_j}{n_p^i \times \max\left\{d_1, \ldots, d_{n_p}\right\}},$$

where n_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 $C_i < 1$ otherwise. For nodes incident with only one pipe, the value of C_i is taken to be 1. 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

$$X = \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} C_i P_i = \sum_{i=1}^{n} C_i q_i (h_i - h_i^*).$$

This expression may be normalized by dividing by the maximum surplus power to obtain the Network Resilience,

$$I_n = \frac{X}{X_{\max}} = \frac{\sum_{i=1}^{n} C_i q_i^*(h_i - h_i^*)}{\sum_{k=1}^{n_r} Q_k H_k - \sum_{i=1}^{n} q_i^* h_i^*}, \tag{4}$$

where $X_{\max}$ is the maximum surplus power. Network resilience may also be viewed as equivalent to the Resilience Index with surplus power at node i given a weight of C_i based on the uniformity in diameter of pipes incident with it [33].

5 Multi-objective Algorithms

Twenty-three WDSDO algorithms were selected or developed for comparison purposes in this paper. This selection includes three typical MOEAs, namely the NSGA-II [5], the *Strength Pareto Algorithm II* (SPEA-II) [56], and generalised *Differential Evolution* (DE) [27]; two estimation of distribution algorithms, namely the (UMDA) [32] and the novel *Partitioned UMDA* (PUMDA); two self-adaptive MOEAs, namely *Another Dynamic Multi-objective Evolutionary Algorithm* (ADMOEA) and a novel algorithm called ANIMA; a population-based greedy heuristic algorithm, appropriately entitled GREEDY; a multi-objective *Particle Swarm Optimization* (PSO) algorithm [24], and finally fourteen different implementations of AMALGAM, including eleven implementations of novel variants developed for this study. These algorithms are discussed in the following sections.

5.1 Non-dominated Sorting Genetic Algorithm II

The NSGA-II was developed by Deb *et al.* [5] in 2002 in an attempt to improve upon the performance of its predecessor, the NSGA. Its design objectives were to reduce the number of optimization parameters, improve its elitism scheme, and improve upon the computational complexity of the non-dominated sorting algorithm used in most MOEAs at the time. It is still considered an advanced MOEA, and has outperformed many of its brethren for numerous optimization problems. NSGA-II is also frequently used as a benchmark algorithm in MOO studies (Farmani *et al.* 2003 [9]; Kapelan *et al.* 2005 [22]; Olsson *et al.* 2009 [32]; Prasad and Park 2004 [33]).

The NSGA-II is notable for the relatively low computational complexity of its dominance rank (depth of front to which a solution belongs) sorting algorithm, called the *Fast Non-dominated Sorting Algorithm* (FNSA). The NSGA-II further uses the concept of crowding distance to distinguish between solutions of identical rank. This yields a measure of how isolated a solution is in objective space, and solutions with larger crowding distance values are favoured in order to explore less crowded regions of that space. NSGA-II was implemented in this paper using an SBX crossover [6] for all variable types (including the discrete variables, for which rounding to the nearest integer was employed), since it was found that this produced superior results to binary crossover, including the standard SBX operator (with $n_c = 2$ and a crossover probability of 0.5 for each corresponding gene pair) and TD mutation (using a component-wise mutation probability of 0.005) [35].

5.2 Strength Pareto Algorithm II

SPEA-II was developed by Zitzler *et al.* [56] in order to improve upon its predecessor and take advantage of new MOEA techniques. It has proven very competitive when compared to the NSGA-II algorithm, particularly in terms of solution diversity. SPEA-II employs an evolutionary population of solutions and a fixed-size archive to store non-dominated solutions. Fitness assignment uses the concept of dominance strength (the number of solutions an individual dominates), the distance to the k-th nearest neighbour in objective space, as well as a special truncation operator, designed to prevent clustering and preserve boundary solutions [56]. Although SPEA-II has a similar computational complexity to NSGA-II, it generally performs slower than NSGA-II in comparative studies [11]. However, SPEA-II generally achieves a more even solution distribution in objective space. The same variational operators as used for NSGA-II were employed here for SPEA-II.

5.3 Differential Evolution

DE was first proposed by Storn and Price in 1997 [41] as a generic metaheuristic for the optimization of nonlinear and non-differentiable continuous space functions, and has proven very robust and competitive with respect to other evolutionary algorithms. At the heart of its success lies a very simple differential operator, whereby a trial solution vector is generated by mutating a random target vector by some multiple of the difference vector between two other random population members. For three distinct random indices i, j and k, this has the form

$$\boldsymbol{y}_i = \boldsymbol{x}_i + \hat{f} \times (\boldsymbol{x}_j - \boldsymbol{x}_k),$$

where $\boldsymbol{x}_i$ is the target vector, $\boldsymbol{y}_i$ is the trial vector and $\hat{f}$ is a constant factor in the range $[0, 2]$ which controls the amplification of differential variation, typically taken as 0.5. If the trial vector has a better objective function value, then it replaces its target vector. The original DE method was formulated for single-objective optimization only. Several adaptations of DE have been proposed in order to extend it for MOO, such as *Generalized DE* in various versions [27, 28]. The version used in this paper creates two offspring as the addition and subtraction of the solution difference vector, one of which replaces the target vector if it is nondominated with respect to it. In a sensitivity analysis, it was found that a difference factor $\hat{f}$ of 0.7 produced the best results overall for WDSDO.

5.4 Univariate Marginal Distribution Algorithm

The UMDA is a simple EDA, that makes no assumptions regarding interaction between variables. In each generation it builds anew separate probability distributions for each gene using allele frequencies in the population. These univariate probability density functions are then stochastically sampled in order to generate new gene values for offspring creation. The probability of generating a particular individual is the product of the individual's allele probabilities. This simple technique is surprisingly effective when combined with a Pareto-based selection scheme and an anti-crowding mechanism, such as that of the NSGA-II which is employed here.

In 2009 Olsson *et al.* [32] compared three EDAs for MO WDSDO, including the *Hierarchical Bayesian Optimization* (HBO) algorithm, the *Chi-square matrix method* (CSM) for building block identification, and the UMDA. They found that HBO was ineffective for designing large WDSs, CSM maintained good performance for the larger systems, demonstrating a better solution spread than UMDA, but UMDA was clearly the best algorithm overall in terms of Pareto-dominance.

A variant of the UMDA, called the *Partitioned UMDA* (PUMDA), was developed for this paper, whereby in each generation the objective space is partitioned along the reliability axis. The size of each partition is generated independently using half of the absolute value of a normal distribution sample with a mean of zero and a standard deviation equal to one third of the reliability range ($r_{\min}$,$r_{\max}$), sampled iteratively until the full range is partitioned. For each sub-range of the reliability range, all the solutions that fall into that partition are then employed to generate a UMD probability model for that partition. In order to generate new solutions, a partition is selected at random and its UMD model sampled to produce offspring. Different levels of reliability correspond to different design paradigms (described by critical solution schemata), which can only be exploited fully by honing in on these regions of the objective space. By allowing variable-sized partitions and redefining them during each generation, PUMDA allows for paradigm overlap and mixing. PUMDA was found to outperform the UMDA for many of the test cases. Both the UMDA and PUMDA algorithms were implemented within the NSGA-II framework.

5.5 A Multi-objective Greedy Algorithm

A WDS design heuristic named GREEDY was developed for this study. It was adapted from four prior WDS design heuristics, namely those of Keedwell and Khu [23] in 2006, Afshar *et al.* [1] in 2005, Todini [42] in 2000, and Saldarriga *et al.* [37] in 2008, using design strategies similar to those that might be used by a human engineer. In addition to these heuristics, it also employs several practical adjustment steps to improve performance based on engineering judgement. The new combined algorithm is greedy in the sense that it conducts a neighbourhood search in which the best improvement step is followed for each of the different heuristic rules. The practise of incorporating different search mechanisms reduces the probability of becoming trapped in local optima.

Todini presented a goal-programming design heuristic for rapidly approximating the Pareto-optimal curve in cost/resilience space. For a given solution, if no pressure deficit occurs, a reduction of diameters is performed with respect to the pipe p_i^* for which the largest decrease in cost per unit of power dissipation occurs during a single step reduction in pipe diameters ($j^\lambda \to j^{\lambda-1}$). That is,

$$p_i^* = \max_{i=1,\ldots,p}\left\{-\frac{C_i^{\lambda-1} - C_i^{\lambda}}{P_{\mathrm{r},i}^{\lambda-1} - P_{\mathrm{r},i}^{\lambda}}\right\},$$

where C_i^λ and $P_{\mathrm{r},i}^\lambda$ are the cost and power, respectively, of pipe i at the current diameter $x_i = j^\lambda$. The diameter reduction only takes place if velocity constraints are not exceeded, if the Resilience Index does not fall below the current target value, and if the new diameter does not cause a nodal pressure deficit. When a pressure deficit occurs in the system, the inverse procedure is followed: Pipe diameters are iteratively increased according to the largest decrease of internal power dissipation per unit cost [42]. A similar procedure may be followed for the unitary power metric of Saldarriga *et al.* [37], which is defined as pipe discharge q_i, multiplied by the difference between the pressure head at the pipe's initial ($h_{i,\mathrm{init}}$) and final ($h_{i,fin}$) nodes, such that $h_{i,\mathrm{init}} - h_{i,fin} > 0$. The pipe unitary power is therefore $P_{\mathrm{ru},i} = q_i(h_{i,\mathrm{init}} - h_{i,fin})$. The diameter of the pipe with the largest $P_{\mathrm{ru},i}$ value may be increased to the next diameter size, and similarly, the pipe with the smallest unitary power may be decreased to the next smallest commercial diameter.

The heuristic of Afshar *et al.* [1] identifies a node with the maximum head deficit. All paths conveying water from any source to this node are established using flow direction, and a pipe is selected for diameter increase that results in the maximum decrease in a total penalized system cost (for which a function similar to (2) is employed). This algorithm may also be applied in the reverse direction, by decreasing pipe diameters on paths towards a node with a maximum head violation. A similar process is used for pipe velocity violations, considering all pipes that exit the source node of the violating pipe.

Keedwell and Khu [23] developed a cellular automata approach towards initializing WDS optimization searches with healthy designs instead of using random initial configurations. The method is called the *Cellular Automaton Network Design Algorithm* (CANDA). It considers each demand node as a cell in an automaton and iteratively evaluates the head deficit or excess of that node. If a node experiences a pressure deficit, all the pipes supplying water to that node are increased to the next largest size. Similarly, if a node experiences a head excess, the incoming pipes are downsized. These changes are implemented for every node in the network before the next hydraulic simulation is conducted. It was shown that this method converges rapidly to semi-realistic configurations, but was of limited use in further refining designs.

Finally, the additional heuristic steps implemented in GREEDY are: incrementing / decrementing the diameter of the pipe which has the largest / smallest head loss, and similar steps for head loss per unit length. During each generation GREEDY generates offspring by selecting solutions without replacement from the population, and then executing each of the aforementioned heuristic transition steps (in both the increasing and decreasing capacity directions) on each solution to generate new offspring, forming a child population. These offspring are selected for propagation into the next generation using an NSGA-II environmental selection framework. Duplicate solutions are replaced by a random solution modified using randomly between five and ten CANDA steps.

5.6 Multi-objective Particle Swarm Optimisation

PSO is a metaheuristic inspired by the flocking behaviour of birds and insect swarms. Kennedy and Eberhart [24] proposed the original PSO algorithm in 1995, and it has steadily gained popularity, owing to its features of robustness and rapid convergence. Although PSO was originally developed for continuous optimisation, it may be adapted for discrete optimisation [20].

In PSO, solutions in a population are treated as particles flying through the decision space, each associated with a current *velocity* $\boldsymbol{v}$, a memory of its previous *personal best position* $\boldsymbol{p}_{\text{best}}$, knowledge of the *global best position* $\boldsymbol{g}_{\text{best}}$ and sometimes, a local best position $\boldsymbol{l}_{\text{best}}$, within some neighbourhood — defined either in terms of Euclidian distance in objective space, or by some *neighbourhood topology*. Particles are initialized with a random velocity at a random starting position. For particle i at position $\boldsymbol{x}_i^t$ during iteration t, velocity and position are updated as

$$\begin{aligned} \boldsymbol{v}_i^{t+1} &= w\boldsymbol{v}_i^t + c_1u_1(\boldsymbol{p}_{i,\text{best}} - \boldsymbol{x}_i^t) + c_2u_2(\boldsymbol{g}_{\text{best}} - \boldsymbol{x}_t^i) \quad \text{and} \qquad (5) \\ \boldsymbol{x}_i^{t+1} &= \boldsymbol{x}_t^i + \boldsymbol{v}_i^{t+1} + u_3\boldsymbol{x}_i^t, \qquad (6) \end{aligned}$$

respectively, where w denotes the *inertial weight*, controlling the effect of a particle's previous velocity, where c_1 and c_2 are the *learning factors* for *cognitive* and *social* learning respectively, where $u_1, u_2 \in [0, 1]$ are uniform random variables, and where $u_3 \in [-\frac{1}{2}, \frac{1}{2}]$ is a uniform random variable controlling turbulence [45].

A basic multi-objective version of PSO was developed for inclusion in this paper. This version calculates PSO fitness as the crowding distance of a solution divided by the square root of its Pareto-rank. No global best position is used, only a local best, which is identified for each dominated individual as the Pareto-solution which yields the largest PSO fitness value normalized by the Euclidean distance between the solutions in objective space. Particle collisions are accommodated by randomly generating a new solution with a random initial velocity. In this paper, MOPSO is implemented using an inertial weight of $w = 0.75$ and learning factors $c_1 = c_2 = 2$. Finally, the algorithm was adapted to round continuous positions to discrete component values.

5.7 Dynamic Multi-objective Evolutionary Algorithm

The DMOEA, developed in 2003 by Yen and Lu [54], is a cellular multi-objective evolutionary algorithm. It is 'cellular' because the objective space is divided into a grid of a user-specified granularity with the intention of improving algorithmic efficiency. The grid is used as an environmental model to store solution quality information (Pareto-rank and density) organized per grid cell, such that the solutions need not be compared directly to one another, but rather interact only with the grid in order to determine their comparative quality. Grid cells may dominate each other in the usual Pareto fashion and a solution takes on as rank the domination count of the cell in which it lies. Similarly, the number of solutions in a particular grid cell provides a density estimate for these solutions. The DMOEA has a slightly inauspicious name, since this could easily pertain to an entire class of MOEAs that exhibit some dynamic behaviour. DMOEA employs population growth and decline strategies in order to obtain a so-called 'optimal' population size, where size optimality is defined in terms of user solution density preferences rather than in terms of algorithmic performance.

An algorithm was developed for this paper based on the original DMOEA [54], using an identical grid model for the cellular rank and density information, but differing in terms of the population growth and decline strategies, since the original DMOEA failed to produce satisfactory convergence to reasonable sizes. This algorithm was called *Another DMOEA* (ADMOEA) to distinguish it from its forerunner. The algorithm incorporates a number of advanced features, including:

1. A growth strategy whereby a number of new solutions are generated as a function of the grid-size (a 200×200 grid size was employed) and the current Pareto-set size.
2. This growth strategy incorporates three different search mechanisms (a differential evolution operator with difference factor $f = 0.7$, an SBX operator with exponent $n_c = 2$, and a PUMDA operator) and a probability vector to control mechanism selection, updated iteratively based on each mechanism's success rate.
3. A solution age that is incremented during each generation. Solutions may not be removed from the population before they reach a certain age (taken as $a_t = 5$ in this paper), allowing them sufficient time to propagate their genes.
4. A probabilistic population decline strategy that selects solutions for removal on the basis of their age, cellular rank and density.
5. A regeneration strategy that recreates the entire population using PUMDA once limited hypervolume improvement has occurred for fifty consecutive generations.
6. An epsilon-archive to hold Pareto-optimal solutions, updated before regeneration.

7. Compression and growth mechanisms to alter the dimensions of the grid in order to zoom in on the important region of the objective space, or to accommodate new solutions outside of the current grid dimensions.

The finer details of its operation are not provided in this paper, since ADMOEA did not outperform the other algorithms. For full details, the reader is referred to Raad [35].

5.8 ANIMA: A Self-adaptive Evolutionary Algorithm

ANIMA is an auto-adaptive MOEA based on the NSGA-II framework and was developed for this paper. It employs two different variation mechanisms, namely the SBX crossover with triangular mutation and a differential evolution operator. What makes ANIMA unique is that it encodes the variation operator parameters and solution genes together, effectively making each solution an agent carrying both the search instructions and the solution information. These parameter values are generated randomly within reasonable ranges for the initial population and any duplicate replacement solutions, but are passed on to newly created offspring by their parent solutions, either with or without mutation. At the time of solution creation, a solution is randomly assigned an evolution state — indicating either the SBX operator or the DE operator as the primary. Variation is achieved by performing both an SBX operation and a DE operation, using three parents. The search parameter values of the primary parent are copied to the offspring, with the primary operator parameters having a 20% chance of variation. Separating the evolution states enables the parameters to settle on favourable values. Good parameter value combinations are found by randomizing and fixing one parameter value, and adjusting the second by means of evolutionary variation in order to suit the first. For the SBX difference index, the minimum and maximum were chosen as $p_{\text{sbxmin}} = 1$ and $p_{\text{sbxmax}} = d/2$, respectively, where d is the dimensionality of the design variable. For the DE difference factor the minimum and maximum were taken as $[p_{\text{dfmin}}, p_{\text{dfmax}}] = [0.6, 1.2]$. For the triangular mutation, the probability range was taken as $[p_{\text{tmin}}, p_{\text{tmax}}] = [0, 0.02]$. The finer details of ANIMAs operation are provided in Raad [35], since it did not outperform the other algorithms compared here.

5.9 AMALGAM: A Hyperheuristic

The AMALGAM algorithm, developed by Vrugt and Robinson [45] in 2007, is a generic evolutionary meta-algorithmic framework which incorporates k sub-algorithms in the solution of a MOO problem. The algorithm employs a population of N solutions whose offspring are created in a *genetically adaptive* manner by dividing the creation of N offspring amongst the sub-algorithms in a way that is proportional to the success of these sub-algorithms during previous generations. Each sub-algorithm has access to the entire population to generate its share of the offspring. The philosophy behind AMALGAM is that the strengths of different metaheuristics can be combined and exploited dynamically to produce a faster, more reliable search than is possible with any one of the algorithms on its own. AMALGAM borrows largely from the NSGA-II [5] to construct a generic multi-method framework. Vrugt and Robinson demonstrated impressive performance enhancements using four sub-algorithms within the AMALGAM framework, namely NSGA-II, Adaptive Metropolis, PSO and DE.

Sub-algorithms in AMALGAM are made to generate offspring in proportion to their reproductive success during previous generations. For this purpose it is required to count, the number of solutions S_{t+1}^j contributed by the j-th sub-algorithm to the population P_{t+1} during generation t. If N_t^j is the number of offspring that sub-algorithm j must generate during generation t, then

$$N_{t+1}^j = N\frac{S_{t+1}^j}{N_t^j} \Bigg/ \sum_{h=1}^{k} \frac{S_{t+1}^h}{N_t^h}, \tag{7}$$

where the ratio of the number of sub-algorithm j's successful offspring in the new population to the number generated is scaled to the combined success ratios of the entire algorithm set. The implementation by Vrugt and Robinson employed a minimum size of $N_t^j = 5$ to avoid inactivating any of the algorithms. New offspring are generated by each algorithm to create the child population Q_{t+1}, which is then subjected to the NSGA-II selection scheme. The process is repeated until a termination condition is satisfied.

Alternative formulations for AMALGAM were devised for this paper, in order to address some of the shortcomings exposed during testing. The AMALGAM scheme need not necessarily be deployed within the NSGA-II framework. An alternative formulation was developed, called *AMALGAMS*, which employs the SPEA-II environmental selection strategy. One important issue not addressed in AMALGAM is sub-algorithm efficiency. In the origional version a very slow sub-algorithm which produces the best offspring will dominate the search. However, it may be possible that a faster algorithm is capable of producing equally good solutions if allocated a larger portion of the offspring, resulting in significant speed enhancements overall. The offspring partitioning formula (7) may be adapted to

$$N_{t+1}^j = N\left(S_{t+1}^j/(N_t^j T_t^j)\right) \Bigg/ \sum_{h=1}^{k} \left(S_{t+1}^h/(N_t^h T_t^h)\right)$$

in order to include the individual running times of the sub-algorithms such that relatively longer running times are penalized, where T_t^j denotes the running time of algorithm j during iteration t. This variant was called *TAMALGAM*. A more intelligent offspring partitioning scheme might include the use of performance metrics, such as hypervolume or dominance rank information, in order to better quantify offspring successes. Two additional formulations were devised. In the first formulation, *AMALGAMI*, the offspring success count per algorithm S_{t+1}^j is replaced by the sum of the squared inverse dominance rankings of the offspring that survive to the next generation, *i.e.*

$$\hat{S}_{t+1}^j = \sum_{i \in O_{t+1}^j} \left(\frac{1}{\text{rank}(i)}\right)^2,$$

where O_{t+1}^j is the set of successful offspring produced by algorithm j. The second formulation, *AMALGAMJ*, also replaces N_t^j by $(N_t^j)^{\frac{1}{2}}$ in order to reduce the importance of the number of offspring generated. The two considerations of sub-algorithm efficiency and finely-grained performance evaluation may be combined to produce additional variants

TAMALGAMI and *TAMALGAMJ*. The five best formulations, AMALGAM, AMALGAMS, TAMALGAM, AMALGAMI and TAMALGAMJ, are included in this paper. For sake of brevity, these are denoted in shortened form as A, AS, TA, AI, and TAJ, respectively.

The AMALGAM variants are implemented with sub-algorithms indicated by the first-letter subscript (*i.e.* a subscript 'n' denotes NSGA-II, 'd' denotes DE, 'u' denotes UMDA, 'p' denotes PUMDA, and 'g' denotes GREEDY). AMALGAM has been implemented with three different selections of sub-algorithms, the first two comprising three sub-algorithms ('ndu' and 'ndp'), and the final selection comprising four sub-algorithms ('ndug'). The following variants of AMALGAM were implemented in this study: $\mathrm{A_{ndp}}$, $\mathrm{A_{ndu}}$, $\mathrm{A_{ndug}}$, $\mathrm{AS_{ndp}}$, $\mathrm{AS_{ndu}}$, $\mathrm{TA_{ndp}}$, $\mathrm{TA_{ndu}}$, $\mathrm{TA_{ndug}}$, $\mathrm{AI_{ndp}}$, $\mathrm{AI_{ndu}}$, $\mathrm{AI_{ndug}}$, $\mathrm{TAJ_{ndp}}$, $\mathrm{TAJ_{ndu}}$ and $\mathrm{TAJ_{ndug}}$.

6 Algorithmic Performance Evaluation

Two Pareto-compliant measures of solution quality were selected for this paper. The use of multiple performance measures in MOO, including ones that are Pareto-compliant, is recommended by Knowles *et al.* [25] in a tutorial on performance assessment for MOEAs. A Pareto-compliant performance indicator is one that only gives preference to one approximation set, $\boldsymbol{A}$, over another set, $\boldsymbol{B}$, if $\boldsymbol{B}$ does not weakly dominate $\boldsymbol{A}$. Most unary indicators are non-Pareto-compliant, but the *hypervolume* metric [55] is a popular exception.

The first performance measure employed here is a *dominance rank* quantifier that uses a *binary weak-domination indicator* to determine the dominance relationship of one approximation set with respect to another. One may say that a set $\boldsymbol{A}$ is better than a set $\boldsymbol{B}$ if it weakly dominates $\boldsymbol{B}$, and is dissimilar from $\boldsymbol{B}$. Symbolically this is expressed by $\boldsymbol{A} \triangleleft \boldsymbol{B}$. A given approximation set $\boldsymbol{A}$ is compared to every other approximation set in the total pool $\mathcal{P}$ of approximation sets produced by the various algorithms. The dominance rank of $\boldsymbol{A}$ is then

$$\mathrm{rank}(\boldsymbol{A}) = 1 + \left|\{\boldsymbol{B}_i \in \mathcal{P} : \boldsymbol{B}_i \triangleleft \boldsymbol{A}\}\right|.$$

The lower the ranking, the better the approximation set is with respect to the entire set pool. The rank must be calculated for each set $\boldsymbol{A}_i^j$ $(i = 1, \ldots, m)$ produced by each algorithm $(j = 1, \ldots, n)$, forming a set of rank samples for each algorithm,

$$d_{\mathrm{R}} = \left[\left\{\mathrm{rank}(\boldsymbol{A}_1^1), \mathrm{rank}(\boldsymbol{A}_2^1), \ldots, \mathrm{rank}(\boldsymbol{A}_m^1)\right\}, \ldots, \left\{\mathrm{rank}(\boldsymbol{A}_1^n), \mathrm{rank}(\boldsymbol{A}_2^n), \ldots, \mathrm{rank}(\boldsymbol{A}_m^n)\right\}\right].$$

The second performance measure used is the unary hypervolume metric by Zitzler and Thiele [58], a fine-grained analysis which measures the total hypervolume of the objective space dominated by a given approximation set, relative to a reference point. Higher hypervolumes are more desirable, since more space is dominated. This metric has the advantage of representing both closeness to the true Pareto-front and solution diversity. Any reference point used should be dominated by the entire set of known solutions. Hypervolume is often provided in normalized form.

A third novel non-Pareto compliant performance metric was developed for this paper, named the *ϵ-archive size*. An ϵ-archive is a grid of user-specified granularity placed over the objective space, such that a single best solution is retained in each grid cell. The number of solutions (size) of an ϵ-archive gives a good indication of the solution spread and evenness along the Pareto-front, but should only be used in combination with a Pareto-compliant performance measure.

It is traditional to compare the performance of evolutionary algorithms by executing different algorithms with similar population sizes for an equal number of generations [3, 38, 40]. This allows results from different studies to be compared independently of the computer software and hardware used. However, this method becomes impractical when using algorithms which require different population sizes (such as EDAs), or metaheuristics with adaptive population sizing. Additionally, it may be highly unfair, since an evolutionary generation may entail an arbitrary amount of numerical processing, possibly including a local search subcomponent, resulting in very different generational processing times from one algorithm to the next.

An alternative testing methodology, and the one adopted in this paper, entails using a time budget, such that an algorithm must do its best within an allotted time-frame. The identification of a fair time limit was achieved as follows: Algorithms were executed independently until they 'converged,' where convergence is defined as failure to improve results above a specified threshold over a required number of consecutive generations. In this paper a threshold of 0.05% change in hypervolume was used, which had to occur for 200 consecutive generations for the algorithm to have 'converged.' These convergence time trials were completed thirty times for each algorithm to yield an average convergence time. The 90$^{\text{th}}$ percentile of the convergence times amongst all the algorithms was used as the limit in ensuing full time trials. This method may also be used to transfer experiments to a new computer platform (*i.e.* attaining new time limits for the new system), provided similar convergence thresholds and software implementations are employed.

7 WDSDO Experiment Implementation Details

In this paper the following nine WDS benchmarks documented in the literature were employed to test the relevant algorithms: the *New York Tunnel* problem (NYTUN), proposed by Schaake and Lai [39] in 1969, the *Two-loop Network* (TLN), introduced by Alperovits and Shamir [2] in 1977, the *Hanoi Network* (HANOI), first presented by Fujiwara and Khang [12] in 1990, the *Two Reservoir Problem* (TRP), proposed by Simpson *et al.* [40] in 1994, the *Blacksburg* (BLACK), *Fossolo* (FOSS), *Pescara* (PESCA) and *Modena* (MOD) networks presented by Bragalli *et al.* [4] in 2008, and the *Exeter Water Network* (EXNET), documented by the Centre for Water Systems in Exeter, UK [8]. These WDS benchmarks represent different aspects of real-world systems, varying in numbers and dimensions of components. Detailed descriptions of the benchmarks may be found in Raad [35].

In order to test the algorithms with respect to WDSDO on these benchmarks, deterministic demands were utilised as stated in their problem descriptions. The Network Resilience surrogate measure was employed, and the optimisation paradigm was taken as the bi-objective problem of cost minimisation and reliability maximisation. The penalty term method was used, as per (2), with penalty factors determined as $p_{\text{f}} = C_{\max}/0.01$, where $C_{\max}$ denotes the maximum cost of a WDS design (ensuring that a 1% or greater total normalized constraint violation will yield a penalty greater than the most expensive network cost). This method of selecting the penalty term produces large penalty values, which are scaled to the magnitude of the cost. Smaller penalty factors would be more appropriate if the focus was on locating the least-cost design, but this would also result in the final population containing a large proportion of infeasible solutions. The above approach works very well for obtaining entire Pareto-optimal solution sets of feasible solutions. Penalty factors for the various benchmarks are presented in Table 1. Fair time trials are conducted, using a *convergence time trial* followed by a *full time trial*, which employs the time limits obtained in the convergence trial. Thirty convergence trials were conducted in order to compute average times, with initial populations generated stochastically by means of stochastic Latin hypercube sampling [19]. The *replacement of duplicate solutions* in a population with randomly generated solutions was found to improve algorithm performance, and was therefore used for all algorithms in this study. Where multiple demand loadings were present, the minimum reliability surrogate value obtained for all the demand scenarios was taken as the objective to be maximized. Solution quality metrics were calculated in cost-Network Resilience space, using only feasible solutions in the final approximation sets.

Population sizes for the different algorithms and benchmark combinations were generated in accordance with Goldberg's sizing method of competing populations as described by Harik and Lobo [18]. In this analysis, UMDA and PUMDA were treated separately from the other algorithms during population sizing, since they are EDAs and typically require a population size which is a different order of magnitude, effectively defining two population size groups. Goldberg's sizing method was applied ten times for each of the algorithms in the two size groups, obtaining the most prevalent 'optimal' sizes. The largest such size within each group was then employed for all algorithms within that group, both for convergence and time trials. The resulting population sizes are shown in Table 1. Hypervolume reference points and ϵ-precision values were required to compute the hypervolume in cost-reliability space and calculate the ϵ-archive size metric. Values for maximum cost, minimum reliability, and ϵ-precision were generated for each benchmark by considering the represented ranges of cost and reliability in the achieved approximation sets. These values also appear in Table 1. All numerical computations were performed on an Intel Core 2 Duo processor (E6850 3.00GHz) with 3.25 GB of RAM, running on a Windows XP SP3 operating system. All hydraulic simulations were conducted using the OOTEN Toolkit for EPANET2 [44].

8 WDSDO Experimental Results

The results from the comparative analysis are summarised in tabular form, including averages and standard deviations of hypervolume, dominance rank, epsilon archive size, and normalised convergence times, averaged across the first eight benchmarks, and then treated separately for the EXNET benchmark. Hypervolume is normalized (NHV) by the hypervolume of the best attainment set produced after combining all algorithms' attainment sets, in order to express hypervolume performance as a percentage of best known hypervolume. Algorithms are ranked firstly on the basis of average dominance rank,

WDS	Penaltyf	Size1	Size2	ϵ_C	ϵ_R	Mx Cst	Mn NR
TLN	440 000 000	64	128	50 000	0.0005	5 000 000	0
TRP	286 123 962	64	128	15 000	0.002	100 000 000	0
NYTUN	29 410 320 000	64	64	1000 000	0.004	300 000 000	0
HANOI	1 096 979 760	128	64	100 000	0.002	12 000 000	0
BLACK	187 010 000	64	256	10 000	0.0015	1 500 000	0.4
FOSS	166 192 258	64	256	10 000	0.0015	2 000 000	0.4
PESC	1 900 444 000	64	256	75 000	0.0015	15 000 000	0.4
MOD	2 808 336 962	64	256	75 000	0.002	15 000 000	0
EXNET	9 751 805 500	128	N/A	500 000	0.004	5 000 000 000	0

Table 1: *Parameters for the WDS benchmarks: Penalty factors (Penaltyf), Group 1 population size (Size1), Group 2 population size (Size2), ϵ-precision values for cost (ϵ_C) and surrogate reliability (ϵ_R), Hypervolume reference points for cost (Mx Cst) and Network Resilience (Mn NR).*

and secondly using average hypervolume, then by the standard deviation of hypervolume, and finally by epsilon archive size. Time to convergence is expressed in normalised form (NT) by dividing convergence time by the average for a particular benchmark.

8.1 Summary and Analysis of First Eight Benchmarks in Phase 1

The results for the first eight benchmark tests (TRP, TLN, HANOI, NYTUN, BLACK, FOSS, PESC and MOD) are summarized in Tables 2 and 3, showing results for the convergence analyses and time trial analyses, respectively, averaged across all eight benchmarks. Table 2 is sorted in terms of increasing average NT, and reports the averages of average and standard deviation for d_R, NHV and NT. Table 3 is sorted in terms of increasing average dominance rank, and reports the average rank, the averages of average and standard deviation for d_R, NHV and ϵ-archive size A_S.

The results from the convergence analysis indicate that ADMOEA was the fastest algorithm, requiring on average 0.5279 of the average time to converge in order to achieve on average 0.9063 of the best known NHV. This can be attributed to its dynamic population sizing and offspring generation methodologies which typically process fewer solutions per evolutionary generation. However, an average dominance rank of 43.23 and the highest standard deviation of 0.0230 for NHV indicates that the ADMOEA speed enhancement comes at the cost of missed Pareto-optimal solutions. The closest competitor in terms of time is A_{ndp} with 0.7327 of the average time to convergence for an average NHV of 0.9426, and the best SD NT of 0.1945, revealing that it is the most consistent performer in terms of convergence time. With an average dominance rank of 6.43 A_{ndp} still lagged behind the leading algorithms, showing that the speed enhancement comes at the cost of reduced performance, but that it may be suitable for use in time critical applications. The algorithm achieving the best average NHV is A_{ndu}, the original formulation which achieves an average of NHV of 0.9502 in 0.8218 of the average convergence time. With an average dominance rank of 2.39, this algorithm would seem suitable for general use. The algorithm with the best average and standard deviation of dominance rank is TA_{ndu}, with values of 1.78 and 1.82, respectively. It also attains a good average NHV of 0.9345 with a standard deviation of 0.0107. This is achieved at an average convergence time of 0.8754 of the mean. TAJ_{ndu} and NSGA-II also achieve good performance in below average time,

Algorithm	**Avg d_R**	**SD d_R**	**Avg NHV**	**SD NHV**	**Avg NT**	**SD NT**
ADMOEA	43.23	59.02	0.9063	*0.0230*	**0.5279**	0.4057
A_{ndp}	6.43	6.73	0.9426	0.0113	0.7327	**0.1945**
AI_{ndp}	8.36	10.90	0.9429	0.0111	0.7451	0.2161
TA_{ndp}	4.35	4.83	0.9418	0.0100	0.7490	0.1978
TAJ_{ndp}	7.99	9.75	0.9402	0.0116	0.7817	0.2449
AI_{ndu}	7.98	11.31	0.9323	0.0109	0.8175	0.2119
A_{ndu}	2.39	2.58	**0.9502**	0.0138	0.8218	0.2091
TAJ_{ndu}	2.28	3.48	0.9341	0.0105	0.8413	0.2224
DE	27.49	23.38	0.9289	0.0147	0.8576	0.2449
NSGA-II	2.52	3.89	0.9356	0.0110	0.8673	0.2548
TA_{ndu}	**1.78**	**1.82**	0.9345	0.0107	0.8754	0.2299
TAJ_{ndug}	20.49	16.48	0.9277	0.0103	0.9247	0.2615
TA_{ndug}	25.20	17.74	0.9250	0.0100	0.9484	0.2621
ANIMA	4.23	8.40	0.9340	0.0130	0.9587	0.2875
PSO	360.65	43.33	*0.7492*	0.0223	1.0213	0.3592
A_{ndug}	50.00	21.92	0.9215	0.0127	1.0743	0.3171
AI_{ndug}	45.09	23.98	0.9227	0.0111	1.1240	0.3307
GREEDY	*397.72*	55.81	0.7571	0.0226	1.1876	0.3220
UMDA	251.35	*85.58*	0.8148	0.0211	1.2166	0.4595
PUMDA	123.21	68.33	0.8899	0.0198	1.2738	0.2254
AS_{ndp}	1.91	2.34	0.9493	0.0147	1.3158	0.4031
AS_{ndu}	7.51	9.38	0.9414	0.0101	1.6331	0.5011
SPEA-II	2.45	2.63	0.9467	**0.0096**	*1.7043*	*0.5546*

Table 2: *Summary statistics of convergence analyses, with average performance metrics computed over the eight benchmarks TRP, TLN, HANOI, NYTUN, BLACK, PESC, FOSS and MOD.*

and may also be considered candidates for general use. The worst performing algorithm in terms of time is SPEA-II, which requires 1.7043 of the mean convergence time, but yields good results with a dominance rank of 2.45 and an average NHV of 0.9467. SPEA-II is also the best in terms of NHV standard deviation, achieving a value of less than 1%. The worst performing algorithm in the convergence trials is GREEDY with an average dominance rank of 397.72 and average NHV of 0.7571. It is also slower than the average convergence rate at 1.1876. The Pareto-ranked algorithms in terms of average convergence time and average NHV are A_{ndu}, AI_{ndp}, A_{ndp} and ADMOEA. The non-dominated algorithms with respect to average convergence time and dominance rank are TAJ_{ndu}, TA_{ndu}, TA_{ndp}, A_{ndu} and A_{ndp}.

The results from the time trial analyses indicate that NSGA-II is the top performing algorithm in terms of average dominance rank, achieving a value of 1.1. This seems surprising given its simplicity. However, its average rank is only 9, compared to TA_{ndu} with an average rank value of 5.125. TA_{ndu} also achieved the lowest standard deviation for NHV of 0.26, compared to NSGA-II's 0.52. The best algorithm in terms of average NHV is AS_{ndp}, achieving a value of 0.9512. The algorithm with the lowest standard deviation for NHV is TAJ_{ndu}, achieving a value of 0.0085. These four algorithms are non-dominated with respect to the various performance criteria, and may be considered candidates for general usage. The performance results for all algorithms with an average dominance rank lower than 4 are graphed in Figure 1, with the Pareto-front solutions labelled. SPEA-II has also been labelled due to its smaller standard deviation for NHV than AS_{ndp}.

Algorithm	Avg Rank	Avg d_R	SD d_R	Avg HV	SD HV	Avg A_S
NSGA-II	9.000	**1.11**	0.52	0.9421	0.0092	64.61
TAJ$_\text{ndu}$	5.875	1.15	0.35	0.9423	**0.0085**	64.82
TA$_\text{ndu}$	**5.125**	1.19	**0.26**	0.9439	0.0092	64.90
AS$_\text{ndp}$	9.125	1.21	0.61	**0.9512**	0.0128	63.43
AI$_\text{ndu}$	6.000	1.25	0.50	0.9428	0.0086	64.55
SPEA-II	12.375	1.36	0.89	0.9476	0.0089	61.61
A$_\text{ndu}$	9.250	1.37	0.98	0.9423	0.0105	64.55
A$_\text{ndp}$	8.000	1.73	1.06	0.9474	0.0099	65.13
ANIMA	12.375	1.77	2.71	0.9416	0.0112	64.30
AI$_\text{ndp}$	9.750	2.57	4.43	0.9483	0.0095	64.91
ADMOEA	12.125	3.83	10.97	0.9321	*0.0223*	73.41
TAJ$_\text{ndp}$	10.750	6.37	7.58	0.9463	0.0113	65.37
AS$_\text{ndu}$	12.125	6.97	5.70	0.9360	0.0088	62.17
TA$_\text{ndp}$	12.375	7.63	8.75	0.9460	0.0117	65.57
TAJ$_\text{ndug}$	10.875	20.87	12.90	0.9330	0.0090	64.51
TA$_\text{ndug}$	10.125	23.12	12.24	0.9322	0.0095	64.51
DE	12.250	27.81	17.86	0.9470	0.0147	64.48
AI$_\text{ndug}$	13.250	59.45	13.42	0.9240	0.0094	63.94
A$_\text{ndug}$	13.625	62.96	15.05	0.9245	0.0093	64.22
PUMDA	17.500	114.73	78.33	0.8933	0.0190	129.06
UMDA	21.000	234.76	*90.15*	0.8218	0.0193	57.10
PSO	20.875	360.43	41.87	*0.7600*	0.0190	42.71
GREEDY	*22.25*	*410.64*	46.19	0.7614	0.0205	56.23

Table 3: *Summary statistics of full time trials, with average performance metrics computed over the eight benchmarks TRP, TLN, HANOI, NYTUN, BLACK, PESC, FOSS and MOD.*

ANIMA performed reasonably well, but did not make it into the top five algorithms. Its adaptive mechanisms may potentially be refined in a more rational manner in order to improve performance, possibly by means of machine learning techniques. ADMOEA fell just short of a top-ten positions, with its greatest weakness being high performance variability (it has the highest standard deviation for NHV of 0.0223). However, its fast convergence time may be reason enough to consider it for general usage. The worst performing algorithms are PUMDA, UMDA, PSO and GREEDY, where the latter is undeniably the worst algorithm for WDSDO given this sample of benchmarks. The poor performance of UMDA is probably expected due to its lack of innovation, and that of GREEDY due to its being a local search.

8.2 EXNET Benchmark Time Trials

EXNET is very large in comparison to the other eight WDS benchmarks, and consequently requires a significant increase in computational processing time and memory resources to undergo WDSDO. Conducting full time trials for all the algorithms would take several months of computing time. Only the four top performing algorithms for the first eight WDS benchmarks in the full time trials were therefore considered for the EXNET analysis, namely NSGA-II, TAJ$_\text{ndu}$, TA$_\text{ndu}$ and AS$_\text{ndp}$. The combined convergence and time trial analysis with thirty optimisation runs required more than a month of computing time. The point (0, 5 000 000 000) was selected as the hypervolume reference point.

The results of the convergence analysis are shown in Table 4. AS$_\text{ndp}$ demonstrated the

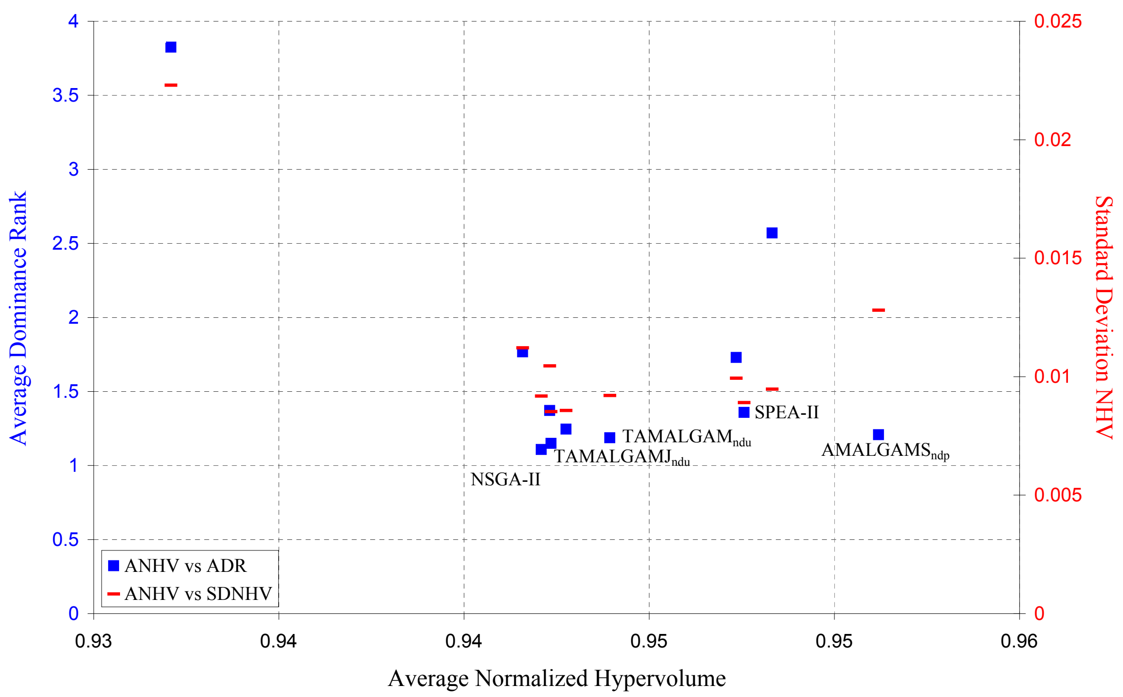

Figure 1: *Summary statistics for time trial analyses: Average normalised hypervolume vs average dominance rank.*

best performance in terms of all metrics, at the cost of the longest average convergence time of 19 507 seconds (5.4 hours). In addition to the lowest average dominance rank of 1.4 and the highest average hypervolume of 0.9014, it also provided the most reliable performance with the lowest standard deviations of 0.84, 0.0444 and 2 638 seconds for dominance rank, hypervolume and convergence time, respectively. The algorithm with the fastest convergence time was TA$_{ndp}$, achieving a time of 10 312 seconds (2.9 hours); however this comes at the cost of significantly reduced performance, with average dominance rank and hypervolume values of 15.7 and 0.7086, respectively. The results produced by TAJ$_{ndu}$ and NSGA-II lie somewhere inbetween, and are mutually non-dominated with respect to dominance rank, hypervolume and convergence time.

Full-length time trials were conducted using thirty optimisation runs and a time limit of 19 507 seconds for each algorithm. The results of these trials are shown in Table 5. AS$_{ndp}$ maintained a convincing lead, achieving an average dominance rank of 3.23 and an average hypervolume of 0.8632. It once again demonstrated the lowest standard deviation values of 3.62 and 0.0429 for dominance rank and hypervolume, respectively, and obtained a significantly larger average ϵ-archive size of 36.53, indicating in combination with dominance rank that it located a much larger portion of the Pareto-front. The closest competitor was TA$_{ndu}$, achieving an average dominance rank of 12.80 and an average hypervolume of 0.7143. TAJ$_{ndu}$ takes third position with an average dominance rank of 14.13 and an average hypervolume of 0.6090. NSGA-II was the worst of the four algorithms by a substantial margin, achieving an average dominance rank of 49.03 and an average hypervolume of 0.5648.

The attainment sets achieved by the various algorithms are graphed in NR-Cost space in Figure 2. Here it is clear that AS_{ndp} located a much larger portion of the Pareto-front, exclusively providing the global front from NR-values of 0.469 to 0.539. However, in the region of NR-values from approximately 0.54 to 0.561, TA_{ndu} found the majority of the non-dominated solutions, exclusively representing most of that section of the global Pareto-front. TAJ_{ndu} and NSGA-II located dominated sub-fronts, with NSGA-II exhibiting the most localised and dominated attainment front. The algorithms' attainment fronts converge at an NR-value of approximately 0.551 and diverge again thereafter. From an NR of approximately 0.555 onwards, AS_{ndp} found only dominated solutions, suggesting that it may not be the best method for discovering solutions of high reliability. The least expensive solution (having the lowest Network Resilience), found by AS_{ndp}, has a cost of 23 168 000 and a Network Resilience of 0.469 758. The most resilient solution, found by TAJ_{ndu}, has a cost of 38 757 400, and a Network Resilience of 0.561 725.

Algorithm	**Avg d_{R}**	**SD d_{R}**	**Avg HV**	**SD HV**	**Avg T (s)**	**SD T (s)**
TA_{ndu}	15.7	4.42	0.7086	0.0529	**10312**	4652
TAJ_{ndu}	14.5	8.68	0.6526	0.1381	11344	3575
NSGA-II	15.6	6.02	0.7256	0.0715	10821	2942
AS_{ndp}	**1.4**	**0.84**	**0.9014**	**0.0444**	19507	**2638**

Table 4: *Time (T) to convergence (taking as stopping criterion less than 0.05% change in HV over 200 generations) for the EXNET benchmark, computed over thirty optimisation runs.*

Algorithm	**Rank**	**Avg d_{R}**	**SD d_{R}**	**Avg HV**	**SD HV**	**Avg A_{S}**	**SD A_{S}**
TA_{ndu}	2	12.80	23.33	0.7143	0.0856	24.27	6.16
TAJ_{ndu}	3	14.13	26.05	0.6090	0.1302	29.37	8.70
NSGA-II	4	49.03	20.62	0.5648	0.0432	16.70	**2.39**
AS_{ndp}	1	**3.23**	**3.62**	**0.8632**	**0.0429**	**36.53**	6.46

Table 5: *Mean and Standard Deviation of performance metrics for the full time trial analysis on the EXNET benchmark, computed over thirty optimisation runs.*

9 Conclusion

In this paper, twenty-three alternative algorithms for multi-objective WDSDO were compared with respect to nine WDS benchmarks from the literature. Convergence time trials were conducted for each algorithm-benchmark pair in order to compare algorithmic efficiency. Full optimisation time trials were then conducted, employing the time limits determined in the convergence trials, in order to compare the solution quality produced by the various algorithms. The top ten performing algorithms applied in the full time trials with respect to the first eight benchmarks included NSGA-II, seven variants of AMALGAM [45], SPEA-II and ANIMA. Attempts to develop state-of-the-art algorithms which consistently outperform conventional multi-objective evolutionary algorithms (such as NSGA-II and SPEA-II) were unsuccessful. Compelling evidence was nevertheless provided that advanced hyperheuristics may be superior for the design of very large or complex WDSs.

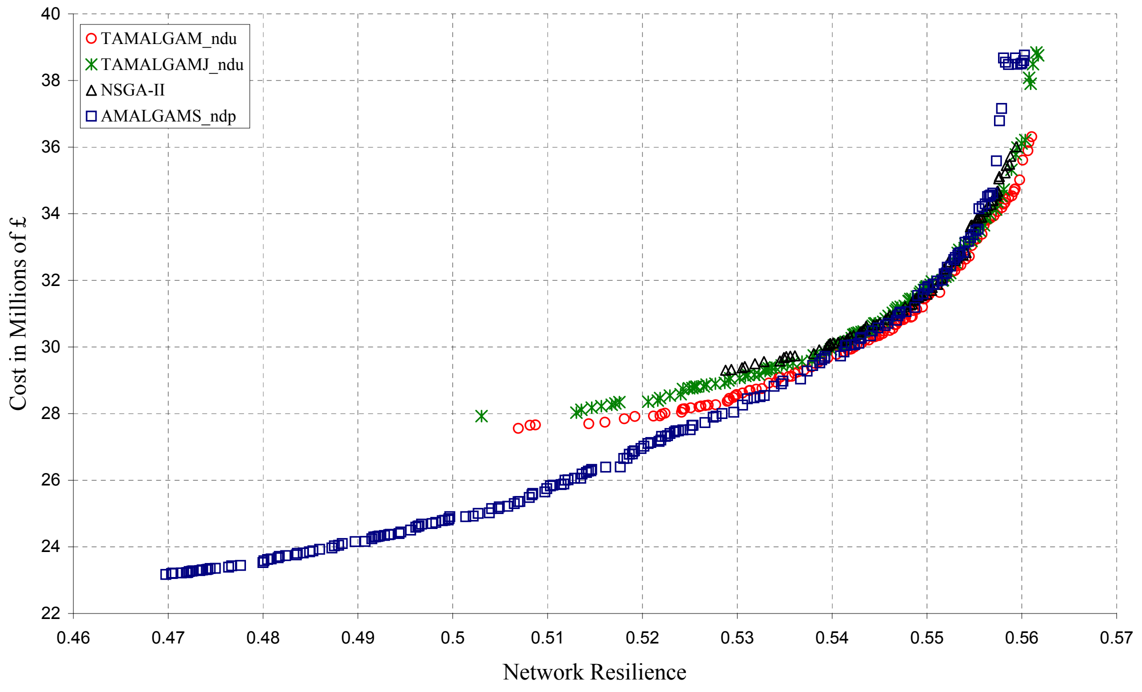

Figure 2: *Attainment sets found by TA_{ndu}, TAJ_{ndu}, NSGA-II and AS_{ndp} for the EXNET benchmark.*

The $\mathrm{AMALGAMS}_{\mathrm{ndp}}$ variant convincingly outperformed the common "industry standard" NSGA-II algorithm at designing the large EXNET benchmark. This provides the research community with a new design algorithm for large / difficult WDSDO problems.

ADMOEA proved the fastest algorithm in terms of convergence, but exhibited unstable performance and fared relatively poorly in terms of dominance rank. The only algorithms that were non-dominated with respect to average convergence time, dominance rank and average hypervolume were $\mathrm{A}_{\mathrm{ndp}}$ and $\mathrm{A}_{\mathrm{ndu}}$. But $\mathrm{TA}_{\mathrm{ndu}}$ was the strongest performer during the convergence trials in terms of dominance rank, which it achieved consistently in below average time, suggesting that it may be the preferred algorithm amongst these for time critical WDSDO. The top four performing algorithms in the full-length time trials were NSGA-II, $\mathrm{TA}_{\mathrm{ndu}}$, $\mathrm{TAJ}_{\mathrm{ndu}}$ and $\mathrm{AS}_{\mathrm{ndp}}$, and were mutually non-dominated with respect to each other for the various performance metrics, with NSGA-II exhibiting the best average dominance rank, and $\mathrm{AS}_{\mathrm{ndp}}$ yielding the best average NHV. The GREEDY algorithm exhibited the worst performance overall, demonstrating that an algorithm which mimics localised engineering judgement cannot compete with modern meta/hyperheuristics. The four best algorithms were executed for the very large EXNET benchmark for thirty 5.4-hour runs each. $\mathrm{AS}_{\mathrm{ndp}}$ proved the best algorithm on the whole, finding a much broader section of the Pareto-front than the other algorithms, although it was outperformed in the high Network Resilience region by $\mathrm{TA}_{\mathrm{ndu}}$, and failed to locate Pareto-optimal solutions of very high values of Network Resilience. $\mathrm{AS}_{\mathrm{ndp}}$ would seem to be the best algorithm at providing a broad range of solutions for difficult WDSDO problems, while $\mathrm{TA}_{\mathrm{ndu}}$ appears to be one of the best choices for efficient, consistent performance and achieving solutions of high reliability. The AMALGAM hyperheuristic and its improved variants have been shown to be very effective within the context of WDSDO, demonstrating that the hybridisation of metaheuristics is a valuable tool.

The findings of this paper were moderately different from those of Raad *et al.* 2009 [34], with AMALGAM no longer demonstrating consistently superior performance over the other algorithms. This is attributed to the new experimental methodology which included more iterations and much stricter convergence criteria, as well as the four additional WDS benchmarks used (BLACK, PESC, FOSS, and MOD), and improved programming techniques in terms of memory management. Finally, this paper may serve the purpose of providing a novel WDSDO framework for the rational investigation of algorithmic efficiency and solution quality that future multi-objective algorithmic comparison studies might consider adopting or improving upon.

References

[1] Afshar MH, Akbari M & Mario MA, 2005, *Simultaneous layout and size optimisation of water distribution networks: Engineering approach*, Journal of Infrastructure Systems, **11(4)**, pp. 221–230.

[2] Alperovits E & Shamir U, 1977, *Design of optimal water distribution systems*, Water Resources Research, **13(6)**, pp. 885–900.

[3] Babayan AV, Kapelan Z, Savic DA & Walters GA, 2005, *Least cost design of water distribution network under demand uncertainty*, Journal of Water Resources Planning and Management, **131(5)**, pp. 375–382.

[4] Bragalli C, D'Ambrosio C, Lee J, Lodi A & Toth P, 2008, *Water network design by MINLP*, IBM Research Report, RC24495 (W0802-056) (Mathematics).

[5] Deb K, Pratap A, Agarwal S & Meyarivan T, 2002, *A fast and elitist multi-objective genetic algorithm — NSGA-II*, IEEE Transactions on Evolutionary Computation, **6(2)**, pp. 182–197.

[6] Deb K & Agrawal RB, 1994, *Simulated binary crossover for continuous search space*, (Unpublished) Technical Report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur.

[7] di Pierro F, Khu S-T, Savic D & Berardi L, 2009, *Efficient multi-objective optimal design of water distribution networks on a budget of simulations using hybrid algorithms*, Environmental Modelling & Software, **24**, pp. 202–213.

[8] Exeter Centre for Water Systems, 2007, *Centre for Water Systems, University of Exeter — Projects Page*, [Online], [Cited March 2nd 2007], Available from `http://www.projects.ex.ac.uk/..`

[9] Farmani R, Savic DA & Walters GA, 2003, *Multi-objective optimisation of water system: A comparative study*, Paper presented at the Conference on Pumps, Electromechanical Devices and Systems Applied to Urban Water Management, 2003, Institute for Water Technology, Valencia.

[10] Farmani R, Walters GA & Savic DA, 2005, *Trade-off between total cost and reliability for Anytown water distribution network*, Journal of Water Resources Planning and Management, **131(3)**, pp. 161–171.

[11] Farmani R, Walters GA & Savic DA, 2005, *Evolutionary multi-objective optimisation in water distribution network design*, Engineering Optimisation, **37(2)**, pp. 167–183.

[12] Fujiwara O & Khang D, 1990, *A two-phase decomposition method for optimal design of looped water distribution networks*, Water Resources Research, **26(4)**, pp. 539–549.

[13] Gessler J, 1985, *Pipe network optimisation by enumeration*, Proceedings of the Speciality Conference on Computer Applications in Water Resources, American Society of Civil Engineers, New York (NY).

[14] Goldberg DE, 1989, *Genetic algorithms in search, optimisation, and machine learning*, Addison Wesley, San Francisco (CA).

[15] Goldberg DE, Deb K, Kargupta H & Harik G, 1993, *Rapid, accurate optimisation of difficult problems using fast messy genetic algorithms*, (Unpublished) Report No. 93004, Illinois Genetic Algorithms Laboratory, University of Illinois (IL).

[16] Goulter IC, Walski TM, Mays LW, Sekarya ABA, Bouchart R & Tung YK, 2000, *Reliability analysis for design*, in Mays LW (Ed), *Water distribution systems handbook*, McGraw-Hill, New York (NY).

[17] Halhal D, Walters GA, Ouazar D & Savic DA, 1997, *Water network rehabilitation with structured messy genetic algorithms*, Journal of Water Resources Planning and Management, **123(3)**, pp. 137–146.

[18] Harik GR & Lobo FG, 1999, *A parameter-less genetic algorithm*, in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), Morgan Kaufmann, pp. 258–267.

[19] Hess S, Train KE & Polak JW, 2004, *On the use of a modified latin hypercube sampling (MLHS) method in the estimation of a mixed logit model for vehicle choice*, (Unpublished) Manuscript, Imperial College, London.

[20] Izquierdo J, Montalvo I, Peacuterez R & Iglesias PL, 2008, *A diversity-enriched variant of discrete PSO applied to the design of water distribution networks*, Engineering Optimisation, **40(7)**, pp. 655–668.

[21] Jourdan L, Corne DW, Savic DA & Walters G, 2006, *LEMMO: Hybridising rule induction and NSGA II for multi-objective water systems design*, Procceedings of the Eighth International Conference on Computing and Control for the Water Industry, **2**, pp. 45–50.

[22] Kapelan ZS, Savic DA & Walters GA, 2005, *Multiobjective design of water distribution systems under uncertainty*, Water Resources Research, **41(1)**, pp. 1–15.

[23] Keedwell E & Khu ST, 2006, *Novel cellular automata approach to optimal water distribution network design*, Journal of Computing in Civil Engineering, **20(1)**, pp. 49–56.

[24] Kennedy J & Eberhart RC, 1995, *Particle swarm optimisation*, Proceedings of the 1995 IEEE International Conference on Neural Networks, Piscataway (NJ), pp. 1942–1948.

[25] Knowles JD, Thiele L & Zitzler E, 2006, *A tutorial on the performance assessment of stochastic multiobjective optimizers*, (Unpublished) TIK-Report No. 214, Computer Engineering and Network Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich.

[26] Knowles J, 2006, *ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimisation problems*, IEEE Transactions on Evolutionary Computation, **10(1)**, pp. 50–66.

[27] Kukkonen S & Lampinen J, 2004, *An extension of generalized differential evolution for multi-objective optimisation with constraints*, Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN 2004), Birmingham, pp. 752–761.

[28] Kukkonen S & Lampinen J, 2005, *GDE3: The third evolution step of generalized differential evolution*, (Unpublished) KanGAL Report No. 2005013, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, Kanpur.

[29] Lansey KE, 2000, *Optimal design of water distribution systems*, in Mays LW (Ed), *Water distribution systems handbook*, McGraw-Hill, New York (NY).

[30] Mays LW (Ed), 2000, *Water distribution systems handbook*, McGraw-Hill, New York (NY).

[31] Nicolini M, 2004, *Evaluating performance of multi-objective genetic algorithms for water distribution system optimisation*, in Phoon S-Y, Liong K-K & Babovic V (Eds), Sixth International Conference on Hydroinformatics, World Scientific Publishing Company, **1**, pp. 850–857.

[32] Olsson RJ, Kapelan Z & Savic DA, 2009, *Probabilistic building block identification for optimal design and rehabilitation of water distribution systems*, Journal of Hydroinformatics, **11(2)**, pp. 89–105.

[33] Prasad TD & Park N-S, 2004, *Multiobjective genetic algorithms for design of water distribution networks*, Journal of Water Resources and Planning Management, **130(1)**, pp. 73–82.

[34] Raad DN, Sinske A & Van Vuuren JH, 2009, *Robust multi-objective optimization for water distribution system design using a meta-meta-heuristic*, International Transactions in Operational Research, **16**(5), pp. 595–626.

[35] Raad DN, 2010, *Multi-objective optimisation of water distribution systems design using metaheuristics*, PhD Dissertation, Stellenbosch University, Stellenbosch.

[36] Rossman LA, 2000, *Computer models/EPANET*, in Mays LW (Ed), *Water distribution systems handbook*, McGraw-Hill, New York (NY).

[37] Saldarriaga JG, Bernal A & Ochoa S, 2008, *Optimized design of water distribution network enlargements using resilience and dissipated power concepts*, Proceedings of the 10th Annual Water Distribution Systems Analysis Conference (WDSA 2008), Kruger National Park, South Africa, pp. 298–312.

[38] Savic DA & Walters GA, 1997, *Genetic algorithms for least-cost design of water distribution networks*, Journal of Water Resources Planning and Management, **123(2)**, pp. 67–77.

[39] Schaake JC & Lai D, 1969, *Linear programming and dynamic programming applications to water distribution network design*, (Unpublished) Technical Report 116, Hydrodynamic Laboratory, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge (MA).

[40] Simpson AR, Dandy GC & Murphy LJ, 1994, *Genetic algorithms compared to other techniques for pipe optimisation*, Journal of Water Resources Planning and Management, **120(4)**, pp. 423–443.

[41] Storn R & Price K, 1997, *Differential evolution: A simple and efficient heuristic for global optimisation over continuous spaces*, Journal of Global Optimisation, **11**, pp. 341–359.

[42] Todini E, 2000, *Looped water distribution networks design using a resilience index based heuristic approach*, Urban Water, **2**, pp. 115–122.

[43] Tolson BA, Maier HR, Simpson AR & Lence BJ, 2004, *Genetic algorithms for reliability based optimisation of water distribution systems*, Journal of Water Resources Planning and Management, **130(1)**, pp. 63–72.

[44] Van Zyl JE, 2007, *OOTEN — Object-oriented Toolkit for EPANET* [Online], [Cited October 7th 2009], Available from `http://epanet.de/en/ooten/index.html`

[45] Vrugt JA & Robinson BA, 2007, *Improved evolutionary optimisation from genetically adaptive multimethod search*, Proceedings of the National Academy of Sciences, **104(3)**, pp. 708–711.

[46] Walski TM, Brill ED, Gessler J, Goulter IC, Jeppson RM, Lansey K, Lee H, Liebman JC, Mays LW, Morgan DR & Ormsbee LE, 1987, *Battle of the network models: Epilogue*, Journal of Water Resources Planning and Mangement, **113(2)**, pp. 191–203.

[47] Walski TM, 2001, *The wrong paradigm — Why water distribution optimisation doesn't work*, Journal of Water Resources Planning and Management, **127(2)**, pp. 203–205.

[48] Walski TM (Ed), 2003, *Advanced water distribution modeling and management*, Haestad Press, Waterbury (CT).

[49] Walters GA, Halhal D, Savic DA & Ouzar D, 1999, *Improved design of Anytown distribution network using structured messy genetic algorithms*, Urban Water, **1(1)**, pp. 23–38.

[50] Wu ZY & Walski T, 2004, *Self-adaptive penalty cost for optimal design of water distribution systems*, Critical Transitions in Water and Environmental Resources Management, American Society of Civil Engineers.

[51] Wu ZY & Simpson AR, 2001, *Competent genetic-evolutionary optimisation of water distribution systems*, Journal of Computing in Civil Engineering, **15(2)**, pp. 90–101.

[52] Xu C & Goulter IC, 1998, *Probabilistic model for water distribution reliability*, Journal of Water Resources Planning and Management, **124(4)**, pp. 218–228.

[53] Xu C & Goulter IC, 1999, *Reliability-based optimal design of water distribution networks*, Journal of Water Resources Planning and Management, **125(6)**, pp. 352–362.

[54] Yen GG & Lu H, 2003, *Dynamic multiobjective evolutionary algorithm: Adaptive cell-based rank and density estimation*, IEEE Transactions on Evolutionary Computation, **7(3)**, pp. 253–274.

[55] Zitzler E, Deb K & Thiele L, 2000, *Comparison of multiobjective evolutionary algorithms: Empirical results*, Evolutionary Computation **8(2)**, pp. 173–195.

[56] Zitzler E, Laumans M & Thiele L, 2002, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm for multiobjective optimisation*, in Giannakoglou K, Tsahalis D, Periaux J, Papailiou K, Fogarty T (Eds), *Evolutionary methods for design, optimisation and control*, CIMNE, Barcelona.

[57] Zitzler E, Laumanns M & Bleuler S, 2003, *A tutorial on evolutionary multiobjective optimisation*, (Unpublished) Technical Report, Computer Engineering and Network Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich.

[58] Zitzler E & Thiele L, 1999, *Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto Evolutionary Algorithm*, IEEE Transactions on Evolutionary Computation, **3(4)**, pp. 257–271.

Appendix: List of acronyms

ADMOEA	Another Dynamic Multi-objective Evolutionary Algorithm
AMALGAM (A)	AMALGAM hyperheuristic
AMALGAMI (AI)	AMALGAM with squared inverse dominance rankings
AMALGAMJ (AJ)	AMALGAMI with reduced order of number generated
AMALGAMS (AS)	AMALGAM with SPEA-II environmental selection
ANIMA	ANIMA Self-adaptive Evolutionary Algorithm
BLACK	Blacksburg Network
CANDA	Cellular Automaton Network Design Algorithm
DE	Differential Evolution
DMOEA	Dynamic Multi-objective Algorithm
EDA	Estimation of Distribution Algorithm
EXNET	Exeter Network
FMGA	Fast Messy GA
FOSS	Fossolo Network
FORM	First Order Reliability Method
GA	Genetic Algorithm
GREEDY	Greedy heuristic MOO algorithm
HANOI	Hanoi Network
MCS	Monte Carlo simulation
MOD	Modena Network
MOEA	Multi-objective Evolutionary Algorithm
MOO	Multi-objective Optimisation
MOPSO	Multi-objective PSO
NHV	Normalised Hypervolume
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NT	Normalised Time
PESCA	Pescara Network
PSO	Particle Swarm Optimisation
PUMDA	Partitioned UMDA
RAM	Random Access Memory
SBX	Simulated Binary Crossover
SMGA	Structured Messy GA
SPEA-II	Strength Pareto Evolutionary Algorithm II
TAMALGAM (TA)	Time penalised AMALGAM
TAMALGAMI (TAI)	Time penalised AMALGAMI
TAMALGAMJ (TAJ)	Time penalised AMALGAMJ
TD	Triangular Distribution
TLN	Two Loop Network
TRP	Two Reservoir Problem
UMDA	Univariate Marginal Distribution Algorithm
WDS	Water Distribution System
WDSDO	WDS Design Optimisation