ACCEPTED VERSION

Wu, Zheng Y.; Simpson, Angus Ross Competent genetic-evolutionary optimization of water distribution systems Journal of Computing in Civil Engineering, 2001; 15(2):89-101

© 2001 American Society of Civil Engineers.

PERMISSIONS

http://www.asce.org/Content.aspx?id=29734

Authors may post the ***final draft*** of their work on open, unrestricted Internet sites or deposit it in an institutional repository when the draft contains a link to the bibliographic record of the published version in the ASCE Civil Engineering Database. "Final draft" means the version submitted to ASCE after peer review and prior to copyediting or other ASCE production activities; it does not include the copyedited version, the page proof, or a PDF of the published version

21 March 2014

http://hdl.handle.net/2440/978

COMPETENT GENETIC-EVOLUTIONARY OPTIMIZATION OF WATER DISTRIBUTION SYSTEMS

By

Zheng Y. Wu[1] and Angus R. Simpson[2], Member, ASCE

ABSTRACT: A genetic algorithm (GA) has been applied to the optimal design and rehabilitation of a water distribution system. Many of the previous applications have been limited to small water distribution systems, where the computer time used for solving the problem has been relatively small. In order to apply genetic and evolutionary optimization technique to a large-scale water distribution system, this paper employs one of competent genetic-evolutionary algorithms — a messy genetic algorithm to enhance the efficiency of an optimization procedure. Maximum flexibility is ensured by the formulation of a string and solution representation scheme, a fitness definition and the integration of a well-developed hydraulic network solver that facilitate the application of a genetic algorithm to the optimization of a water distribution system. Two benchmark problems of water pipeline design and a real water distribution system are presented to demonstrate the application of the improved technique. The results obtained show that the number of the design trials required by the messy genetic algorithm is consistently fewer than the other genetic algorithms.

Key Words: water distribution, optimization model, genetic algorithms, messy genetic algorithms, optimal design and rehabilitation.

INTRODUCTION

Provision of adequate water supply service is an essential requirement for communities around the world. Tremendous amounts of capital are being spent on the design of new water distribution systems and the rehabilitation (or improvement) of existing networks in both developing and developed countries. Cost effective expenditure on the design and rehabilitation is essential to achieve a sufficient quality service due to an ever-tightened budget. Even today, despite the availability of many research papers since the 1960s on the optimization of water distribution networks, the design of water distribution systems is still an approach of trial

[1] Supervising Engineer, MW Soft Inc, 300 North Lake Ave. #1200, Pasadena, CA91101.

[2] Senior Lecturer, Department of Civil and Environmental Engineering, University of Adelaide, Australia 5005.

and error improvement. An engineer uses judgement, based on the evaluation of - for example - the pressures at junctions from a computer hydraulic simulation, to determine which element sizes should be adjusted to further reduce the cost (Karmeli 1968). In the mid-1980s, Walski (1985) and Goulter (1987) both predicted that within the next decade (that was before 1997) water distribution optimization models should become everyday tools of practicing water engineers. In 1992, Goulter concluded that optimization still has not penetrated the water industry. As of today, the common use of optimization by the water industry still has not occurred. Although traditional mathematical optimization methods including linear, non-linear and dynamic programming provided efficient computation procedures for achieving a lower cost solution, the methods suffered from some disadvantages such as (1) ineffective at reaching the least cost solution due to the zero-gradient optimality criteria that easily trapped a search process at a local optimal solution; (2) lack of flexibility at handling discrete design variables and optimizing a partial network that is often required for many practical engineering designs and (3) complexity of implementing and using the technique. These methods have often required for a sophisticated system analysis and careful (also time consuming) input data preparation. As a result, practicing engineers have been cautious to accept the traditional approach for the optimization of a water distribution system.

The introduction of genetic and evolutionary algorithms (GA) for the optimization of the design of a water distribution system (Murphy and Simpson 1992; Simpson et al. 1994; Dandy et al. 1996; Savic and Walters 1997; Wu and Simpson 1996, 1997 and many others) takes a different approach. GA optimization incorporates a hydraulic network solver seamlessly within an optimization process. Thus all the sophistication features of the latest simulation modeling techniques, including water quality aspects, can be an integral part of the genetic algorithm technique. A hydraulic network solver evaluates the hydraulic performance of each design trial that is a member of the genetic algorithm population of network designs. The network hydraulic information is then passed back to the genetic algorithm module for computation of the fitness of the design. The use of hydraulic simulation within the genetic algorithm formulation is the real strength of the technique. As an outcome, genetic algorithm optimization offers the promise of easily accessible optimization of water distribution systems. However, many previous applications of a genetic-evolutionary algorithm has been limited to a small network, where the computer time of searching for the optimal or near optimal solutions and solving for the flow and pressure conditions of each design trial has been relatively small. In order to apply a genetic-evolutionary optimization technique to a large-scale water distribution system, the overall computation efficiency for achieving the least cost design solution needs to be improved. Thus a more efficient genetic algorithm approach

is needed for solving the optimization problem. This paper describes a competent approach to the genetic algorithm optimization of a water distribution system. The improvement results from:

- application of the messy genetic algorithm (Goldberg et al. 1989) that significantly improves the optimization efficiency;
- formulation of the genetic algorithm string and solution representation scheme, and the fitness definition that facilitate the implementation of a genetic algorithm optimization model for handling any combination of system elements;
- integration of the genetic algorithm with a hydraulic network solver that enables the solution method to optimize all the system components including pipes, tanks, valves and pumps under steady state or extended period simulations (EPS).

The paper starts with a brief overview on the traditional optimization techniques and a more detailed review on genetic-evolutionary optimization approaches, followed by the formulation of a comprehensive optimization model, genetic algorithm string and solution representation scheme together with a fitness definition. Furthermore, the key features of a messy genetic algorithm are described in comparison with other genetic algorithms, along with three case studies presented to demonstrate the application of the improved methodology. Finally, conclusions are drawn from the theoretical formulation and the solid numerical experiments.

LITERATURE REVIEW

Optimization of a water distribution system has been a subject of considerable research since the 1960s. Various researchers (Walski 1985; Goulter 1987; Walters 1988; Lansey & Mays 1989a and Goulter 1992) have made comprehensive reviews on the traditional methods. Early research on the optimization of water distribution systems used a linear programming technique. The applications were to a branched system where flows were able to be explicitly determined for each pipe (Karmeli et al. 1968). Alperovits and Shamir (1977) proposed a linear programming gradient method (LPG) for optimization of a looped water distribution system. The LPG method has been improved by a number of researchers. Most recently, Eiger et al. (1994) extended the LPG method by applying a non-smooth optimization technique and duality theory. A method was developed for the computation of a tight lower bound to the global optimal solution. The optimality of a solution was measured by the difference between the tight lower bound and the solution. Thus the lower bound served as an optimality (stopping) criterion for the optimization of water distribution systems. The efficacy of the improved LPG method has been

demonstrated, however, it involves a considerable amount of mathematical sophistication. Other traditional optimization techniques including direct search techniques, dynamic programming, integer programming, and enumerative methods were applied to the optimal design of a water distribution system. A detailed review was given by Lansey and Mays (1989a) on these traditional optimization techniques. Su et al. (1987) integrated a generalized reduced gradient (GRG) technique with a hydraulic network solver KYPIPE. Lansey and Mays (1989b) improved the technique by using an augmented Lagrangian method for handling the design constraints. More recently, Kim and Mays (1994) developed a mixed-integer nonlinear programming formulation for optimal rehabilitation of water distribution systems. Integer variables (taking a value of either 0 or 1) were used for representing pipe rehabilitation actions (e.g. replacement, cleaning and relining). The other variables such as pipe diameters and pump horsepower were treated as continuous variables. The problem was solved by a solution methodology integrating an implicit enumeration scheme for the integer variables, the GRG and KYPIPE for optimizing the continuous variables of pipe diameter and pump horsepower.

Genetic algorithms (Goldberg 1989) are a general search method based on the principles of natural evolution and biological reproduction. It randomly initiates a population of solutions or individuals. Each individual is represented by either alphabetic or binary string that encodes one possible solution. The number of bits in one string is defined as the string length. The strings representing all the possible solutions for a specific problem have an identical length or so-called a problem length that remains unchanged during the search process. This type of genetic algorithms processes fixed-length strings during a GA optimization and is referred as to a "simple genetic algorithm". Application of a simple GA to the optimal design of water distribution systems was developed in the early 1990s (Murphy and Simpson 1992). In this early work, a binary string was used to represent the decision variables of (1) pipe diameters for new pipes and duplicated pipes parallel to existing pipes and (2) rehabilitation actions including cleaning a pipe and laying a parallel pipe to an existing pipeline. The simple genetic algorithm using roulette wheel selection, one-point crossover and bit-wise mutation was applied to determine the least cost combination of pipe diameters and rehabilitation actions for optimal expansion and rehabilitation of a small two-reservoir fourteen-pipe looped water distribution system. The optimal solution was subject to just the minimum junction pressure requirement under three demand loading cases including a peak hour demand loading case and two fire flow demand conditions. The GA found the optimal solution for a set of diameters of the new and duplicated pipes and the options of cleaning the existing pipes. The performance of the simple GA was compared with a complete enumeration and other optimization methods (Simpson et al. 1994). The GA based approach was found to outperform other optimization methods at solving this small problem. It

was observed, however, that the simple GA optimization result was sensitive to the GA parameters and operators. Simpson and Goldberg (1994) investigated factors that influence the performance of the simple GA in finding the optimal solution for the two-reservoir looped network problem. They concluded that the use of the tournament selection scheme and an adequate population size were the most critical aspects of applying a simple GA to the optimal design of water distribution systems. Dandy et al. (1996) improved the GA by using (*i*) fitness scaling; (*ii*) creeping mutation and (*iii*) Gray coding (instead of binary coding), and solved the New York City Tunnel water supply network problem. The improved GA found the least cost solution of $38.8 million. The main difficulty associated with the improved simple GA (as with the simple GA) included the considerable effort required to tune the GA parameters (population size, probability of crossover and mutation) to find the range of low cost solutions. It took dozens of runs to find the optimal solution. In fact, the lowest cost solution of $38.80 million was found infrequently by the improved GA. Savic and Walters (1997) integrated a simple GA with the multi-quality EPANET hydraulic network solver (Rossman 1994) and applied to three benchmark problems in literature. They identified that the optimal solution was sensitive to the coefficients in Hazen William formula used in hydraulic simulation. More recently, Lippai et al. (1999) linked EPANET with a number of simple GA-based optimizers for the optimization of a water distribution system.

A simple genetic algorithm represents a solution space with the strings of an identical problem length. It is a *tidy* representation of a solution space in string length. Goldberg et al. (1989) proposed a competent genetic-evolutionary algorithm — the messy genetic algorithm (mGA) using a variable-length string representation. The length of mGA strings changes not only over generations, but also varies from one string to another in one population. It forms a type of *messy* representation of a solution space. It was found that the messy representation of a solution space is more effective than the tidy representation for the optimization of a water distribution system (Wu and Simpson 1996, 1997; Simpson & Wu 1997; Wu et al. 2000). Halhal et al. (1997) proposed a similar approach called the structured messy GA and applied to maximizing the benefit of water distribution system rehabilitation subject to a limited available budget. The structured messy GA retained partial features of the messy GA by Goldberg et al. (1989). It started with a population of short strings of the same length. The short strings were concatenated over generations. Thus the string length increased equally over generations until it reached a prescribed length. The same length was attained for all the strings within one population. This allowed the simple genetic algorithm operators to be applied to reproduce next generation rather than the messy genetic algorithm operators. The strength of the messy genetic algorithm, however, is the versatile variation of the string length not only within one population but also during an artificial evolution process. It is the variable-length

representation, together with the messy GA operators that empowers an artificial evolution process to identify the good clusters of string bit patterns that are contained in good solutions. Goldberg et al. (1989) demonstrated that the messy genetic algorithm was able to locate optimal solutions in the search space that proved difficult-to-find using a simple genetic algorithm. In this paper, we explore the application of full features of the messy genetic algorithm to enhance the capability of the genetic-evolutionary computation approach to the optimal design and rehabilitation of a water distribution system. Performance and working mechanics of the messy GA are also compared to the fixed-length genetic algorithm paradigms.

A DESIGN AND REHABILITATION FORMULATION

Design of a water distribution system is a multi-phase procedure. Walski (1995) classified it into four stages such as (1) master planning; (2) preliminary design; (3) subdivision design and (4) rehabilitation. The optimization model presented in this paper deals with the problems of the last two categories. For a given network layout, demand loading conditions and an operation policy, the optimal design and rehabilitation of a water distribution system is to determine the least cost combination of (1) new pipe diameters $(\vec{D})$, (2) pipe rehabilitation actions $(\vec{E})$, (3) pump capacities $(\vec{P})$, (4) tank sizes $(\vec{T})$, (5) valve sizes $(\vec{V})$ and setting $(\overrightarrow{VS})$. A new pipe can be an expansion (subdivision) to, a replacement of or a parallel pipe (duplication) to an existing pipeline. The total cost of a design and rehabilitation solution is minimized while satisfying a set of prescribed system criteria.

Cost Objective Function

Total cost of a network design and rehabilitation is the sum of the cost associated with all the components being designed and rehabilitated. Let the total numbers of design pipes, pumps, tanks, valves and rehabilitation pipes be *DPP*, *DPM*, *DTK*, *DVV*, *RPP* respectively; and let the costs associated with each group be (*i*) $c_k(d_k)$ = cost per unit length of the *k*-th pipe diameter selected from a set of available pipe diameter D_m^0 of *DC* choices; (*ii*) $c_k(p_k)$ = cost of the *k*-th pump capacity selected from a set of available pump capacity P_m^0 of *PC* choices; (*iii*) $c_k(t_k)$ = cost of the *k*-th tank size selected from a set of possible tank size T_m^0 of *TC* choices; (*iv*) $c_k(v_k)$ = cost of the *k*-th pressure regulating valve selected from a set of possible valve size V_m^0 of *VC* choices; and (*v*) $c_k(e_k, d_k)$ = cost per unit length of a pipe for the *k*-th rehabilitation action e_k chosen from a set of possible action E_m^0 of *EC* choices and corresponding existing pipe diameter d_k. Thus the cost objective function is given as:

$$C(\vec{D},\vec{E},\vec{P}) = \sum_{k=1}^{DPP} c_k(d_k)L_k + \sum_{k=1}^{RPP} c_k(d_k,e_k)L_k + \sum_{k=1}^{DPM} c_k(p_k) \quad (3)$$

where L_k = length of the k-th pipe. Each of decision variables $\vec{D},\vec{E},\vec{P}$ is to select its possible values from a variable choice table or a set of available component sizes (or capacities), given as:

$$\forall k, \forall d_k \in \left\{ D_m^0,\ m = 1,..., DC \right\} \quad (4)$$

$$\forall k, \forall e_k \in \left\{ E_m^0,\ m = 1,..., EC \right\} \quad (5)$$

$$\forall k, \forall p_k \in \left\{ P_m^0,\ m = 1,..., PC \right\} \quad (6)$$

$$\forall k, \forall t_k \in \left\{ T_m^0,\ m = 1,..., TC \right\} \quad (7)$$

$$\forall k, \forall v_k \in \left\{ V_m^0,\ m = 1,..., VC \right\} \quad (8)$$

A design trial solution is analyzed by calling a hydraulic network solver in a steady state or extended period simulation (EPS). The hydraulic simulation solves a set of quasi-linear equations and ensures the satisfaction of the implicit system constraints corresponding to the conservation of flow continuity at nodes and the energy conservation around loops. The hydraulic system responses are checked against a number of the constraints that are prescribed for a feasible design and rehabilitation solution.

Junction Pressure Constraints

Junction pressure is often required to maintain greater than a minimum pressure level to insure adequate water service and less than a maximum pressure level to reduce water leakage within a system. Thus junction pressure constraints are given as:

$$H_{i,j}^{\min} \le H_{i,j}(t) \le H_{i,j}^{\max}, \quad \forall t, i = 1,..., NJ;\ \ j = 1,..., NDM \quad (9)$$

where $H_{i,j}(t)$ = hydraulic head at junction *i* for demand loading case *j* at time *t*; *NJ* = number of junctions in system (excluding fixed grade junctions); $H_{i,j}^{\min}$, $H_{i,j}^{\max}$ = minimum required and maximum allowable hydraulic pressures at junction *i* for demand loading case *j*; and *NDM* = number of demand loading cases.

Pipe Flow Constraints

A design and rehabilitation solution is also constrained by a set of pipe flow criteria that are often given as a maximum allowable flow velocity and a maximum allowable hydraulic gradient or slope, given as:

$$V_{i,j}(t) \le V_{i,j}^{\max}, \quad \forall t, i = 1,..., NP; \quad j = 1,..., NDM \tag{10}$$

$$HG_{i,j}(t) \le HG_{i,j}^{\max}, \quad \forall t, i = 1,..., NP; \quad j = 1,..., NDM \tag{11}$$

where $V_{i,j}(t)$ = flow velocity of pipe *i* for demand loading case *j* at time *t*; $V_{i,j}^{\max}$ = maximum allowable flow velocity of pipe *i* for demand loading case *j*; *NP* = number of constraint pipes in system; $HG_{i,j}(t)$ = hydraulic gradient (slope) of pipe *i* for demand loading case *j* at time *t* and $HG_{i,j}^{\max}$ = maximum allowable hydraulic gradient of pipe *i* for demand loading case *j*.

Pump Capacity Constraints

A pump can be designed by its capacity of a useful horse power P_k that is often required not smaller than a minimum horse power $Pmin_k$ or greater than a maximum horse power $Pmax_k$, thus pump constraints are given as:

$$Pmin_k \le P_k \le Pmax_k \qquad k = 1, \ldots, DPM \tag{12}$$

Valve Setting Constraints

During the optimization process, a valve setting can be optimized within the range of a minimum required and a maximum allowable setting. The constraint for a valve setting is:

$$VSmin_k \le VS_k \le VSmax_k \quad k = 1, \ldots, DVV \tag{13}$$

where $VSmin_k$ represents the minimum required valve setting for valve k, VS_k designates the valve setting for valve k and $VSmax_k$ denotes the maximum valve setting for valve k.

Tank Flow Constraints

When the size of a tank is taken into account as a design variable, a flow balance must be maintained for a sufficient supply to a water distribution system. Thus a tank design is constrained by:

$$\left| Vtank_{i,k}^{in} - Vtank_{i,k}^{out} \right| \le \Delta V_k, \quad k = 1,..., DTK; \quad i = 1,..., NDM \tag{14}$$

where $Vtank_{i,k}^{in}$ = amount of the inflow to tank k under demand loading case i; $Vtank_{i,k}^{out}$ = amount of the outflow from tank k; ΔV_k = flow balance tolerance of tank k. The optimization problem formulated above is to be solved for the least cost solution by a genetic algorithm optimization technique.

A GENETIC ALGORITHM FORMULATION

A design and rehabilitation solution is represented as a string during a genetic algorithm optimization while the string is evaluated by its fitness, a surrogate measure of the solution optimality. Determining a string representation and formulating its corresponding fitness to an objective function are two critical steps to apply a genetic algorithm to solving a network optimization problem.

String and Solution Representation

A genetic algorithm string and solution representation is to determine (1) the type of strings; (2) the number of bits to represent each decision variable and (3) the mapping that converts a string to a possible solution. A string can be consisted of binary bits, decimal digits or alphabets. Let b be the number of one-bit possible values for a particular string type, for example, $b = 2$ for binary strings and $b = 10$ for decimal strings. The number of the bits that are needed to encode all the possible solution values for one decision variable can be calculated as:

$$Nbit = \left\lceil \frac{\log Nchs}{\log b} \right\rceil \tag{15}$$

where *Nbit* denotes the number of the bits in a sub-string representing one decision variable, $\lceil\ \rceil$ is the ceiling operator that calculates the nearest integer greater than the operand and *Nchs* designates the number of possible solution values for the variable to choose from. For example, *Nchs* = *DC* given as Eq.(4) for selecting a possible pipe size. A string, representing a possible solution to the design and rehabilitation of a specific network, is the concatenation of the sub-strings that designate all the decision variables to the system.

To evaluate the fitness of a string, the string must be converted into a design solution by mapping a string value onto a variable value. For each type of design variables, a choice table, designated by Eq.(4)-(8) for a specific problem, is often given for a variable to look up a corresponding solution value according to its string value. Assuming one sub-string for one variable is represented as $a_1\ a_2\ ,\ldots\ldots a_n$, the value of the sub-string can be calculated as:

$$S_i = \sum_{n=1}^{Nbit} a_n b^{n-1} \tag{16}$$

Then, the sub-string is converted to a solution value by mapping the string value as above to index *m* of the variable choice table given as Eq.4 – 8, namely:

$$m = \begin{cases} S_i & if \quad S_i \le Nchs \\ Nchs & if \quad S_i > Nchs \end{cases} \tag{17}$$

By using this string and solution representation, a genetic algorithm can be applied to solving the network optimization problem formulated earlier. It provides a unified computation framework for genetic-evolutionary optimization of a water distribution system and is also applicable to many other discrete optimization problems. An example is given below to illustrate the string and solution representation scheme applied to the design variables of pipe sizes and rehabilitation actions (cleaning and duplicate pipes). Application to the other types of design variables such as tank sizes, pump capacities and valve settings is straightforward.

A water distribution system, studied by Simpson et al. (1994) as shown in Figure 1, consists of two reservoirs and fourteen pipes. There are five new pipes (*DPP* = 5) to be added to the system, three existing pipes (*RPP* = 3) to be rehabilitated by taking one of the three actions as given in Table 1. The actions include cleaning,

duplicating (laying a parallel pipe) and leaving a pipe as it is. Table 2 gives eight commercially available pipe sizes that can be selected from for a new and duplicated pipe. The task is to determine the least cost solution or combination of rehabilitation actions and pipe sizes while the junction pressures meet the minimum required pressure under three demand loading conditions. Binary strings (i.e. $b = 2$) are used for representing a design solution. To encode eight possible pipe sizes ($Nchs = DC = 8$), the number of binary bits required for one design variable of a new pipe diameter is $Nbit = 3$ as calculated by Eq.(15). Similarly, two binary bits are needed to represent three rehabilitation actions ($Nchs = EC = 3$), for each of three existing pipes to be rehabilitated. Thus the total length of a string for this example problem is 30 bits. Figure 2 shows one string representation of a solution for this example network. The string can be converted to a design and rehabilitation solution. For instance, sub-string 101 for a new pipe gives a string value of 5 by Eq. (16), by mapping the string value of 5 to the index of the diameter choice Table 2, a corresponding pipe diameter of 407 mm is assigned to the sub-string 101. For this example, a 2-bit binary sub-string provides four choices, one string value (i.e. 3) is redundant and is set to the last rehabilitation action of cleaning a pipe. The sample string in Figure 2 is converted into a design solution, as given in Table 3, by mapping a sub-string value to the index of a pipe diameter or a rehabilitation action. For a pipe taking the rehabilitation action of cleaning a pipe, the pipe size remains the same but the roughness coefficient needs to be updated to reflect a cleaned pipe condition. In this way a string is converted into a design and rehabilitation solution. A fitness value is to be assigned to the string as a surrogate optimality measure of the corresponding design solution.

Fitness Evaluation

In a genetic algorithm, fitness is introduced as the performance measure of a string or an individual adapting to an objective landscape. A genetic-evolutionary algorithm searches for the best string by mimicking Darwin's natural selection principle of *survival of the fittest*. Thus string fitness is maximized during a search process and accordingly the best string is the string that gains the maximum fitness value. However, the cost associated with a network design and rehabilitation is to be minimized to search for the least cost solution. Therefore a fitness function needs to be defined such that a genetic-evolutionary algorithm equivalently minimizes the cost objective function while the fitness is maximized. The fitness of a string corresponding to a solution can be formulated in many ways, the fitness definition by Wu and Larsen (1996) has been used as:

$$\Phi_{nn} = 1 - \frac{C_{nn}(\vec{D}, \vec{E}, \vec{P})}{\underset{nn=1,\ldots,NN}{\text{Max}}\, C_{nn}(\vec{D}, \vec{E}, \vec{P})} \tag{18}$$

where *NN* = the population size; $C_{nn}(\vec{D}, \vec{E}, \vec{P})$ = the cost of a design and rehabilitation solution *nn* at current generation. This has a desirable property that the fitness is in the range $0 \le \Phi_{nn} \le 1.0$ and that the cost $C(\vec{D}, \vec{E}, \vec{P})$ will be minimized while the fitness is maximized over generations. The optimization procedure is undertaken by using a messy genetic algorithm.

MESSY GENETIC ALGORITHMS

The working mechanics of a genetic algorithm is derived from a simple assumption (Holland 1975) that the best solution will be found in the solution region that contains a relatively high proportion of good solutions. A set of strings that represent the good solutions attains certain similarities in bit values. For example, 3-bit binary strings 001, 111, 101 and 011 have a common *similarity template* of **1, where wild star * denotes *don't-care* symbol taking a value of either 1 or 0. The four strings represent four good solutions and contribute to the fitness values of 10, 12, 11 and 11 to a fitness function of $f(x_1, x_2, x_3) = x_1 + x_2 + 10^{x_3}$, where x_1, x_2 and x_3 directly takes a bit value as an integer from left to right. In general, a short similarity template that contributes an above-average fitness is so-called a *building block*. Building blocks are often contained in short strings that represent partial solutions to a specific problem. Thus searching for good solutions is to uncover and juxtapose the good short strings that essentially designate a good solution region and finally lead a search to the best solution.

Goldberg et al. (1989) developed the messy genetic algorithm as one of competent genetic algorithm paradigms by focusing on improving GA's capability of identifying and exchanging building blocks. The first-generation of the messy GA explicitly initializes all the short strings of a desired length *k*, where *k* is referred as to the order of a building block defined by a short string. For a binary string representation, all the combinations of order-*k* building blocks requires a number of $n = 2^k \binom{l}{k}$ initial short strings of length *k* for an *l*-bit problem. For example, as shown in Figure 3, the initial population size of short strings by completely enumerating the building blocks of order 4 for a 40-bit problem is more than one million. This made the application of the first-generation messy GA to a large-scale optimization problem impossible. This bottleneck has been overcome by

introducing a building block filtering procedure (Goldberg et al. 1993) into the messy GA. It speeds up the search process and is called a fast messy GA.

The fast messy GA emulates the powerful genetic-evolutionary process in two nested loops, an outer loop and an inner loop. Each cycle of the outer loop, denoted as an era, invokes an initialization phase and an inner loop that consists of a building block filtering phase and a juxtapositional phase. Like a simple genetic algorithm, the messy GA initialization creates a population of random individuals. The population size has to be large enough to ensure the presence of all possible building blocks. Then a building block filtering procedure is applied to select better-fit short strings and reduce the string length. It works like a filter that "bad" genes not belonging to building blocks are deleted so that the population contains a high proportion of short strings of "good" genes. The filtering procedure continues until the overall string length is reduced to a desired length *k*. Finally, a juxtapositional phase follows to produce new strings. During this phase, the processed building blocks are combined and exchanged to form offspring by applying the selection and reproduction operators. The juxtapositional phase terminates when the maximum number of generations is reached. Thus the cycle of one era iteration completes. A summary of the steps in a messy GA is given in Figure 4. The length of short strings that contains desired building blocks is often specified as the same as an era, starting with one to a maximum number of eras. Thus preferred short strings increase in length over outer iterations. In another words, a messy GA evolves solutions from short strings starting from length one to a maximum desired length. This enables the messy GA to mimic the natural and biological evolution process that a simple or one cell organism evolves into a more sophisticated and intelligent organism. Goldberg et al. (1989, 1993) has given the detail analysis and computation procedure of the messy GA. The key components and features of the messy GA are outlined as follows.

Variable-length string

Unlike a simple GA, a messy GA represents a gene by a pair of gene locus and gene value, noted as (*gene locus*, *gene value*), in a string of variable length. A gene locus is the location or sequential order of a gene bit in a full-length string. For binary string representation, each gene bit takes a value of either 0 or 1. For instance, the sample solution string given in Figure 2 is represented as a messy GA string given as:

(1,1), (2,1), (3,1), (4,1), (5,0), (6,1), (7,1), (8,1), (9,0), (10,0), (11,0), (12,1), (13,0),
(14,1), (15,0), (16,0), (17,0), (18,1), (19,1), (20,1), (21,0), (22,1), (23,1), (24,0), (25,1),
(26,1), (27,1), (28,0), (29,1), (30,0)

where the first number within a bracket is the gene locus the sequential order of a bit in the string and the second number refers to the bit value (i.e. 1 or 0). It is the locus that enables the messy GA to locate a bit value in a variable-length string that can be under or over specified. A under specified string is the string with some missing bits while an over specified string is the string with multiple bit values. For example, a 3-bit string can be represented either by (1,1), (3,0) or by (1,1), (2,1), (3,0), (3,1). The former coding set, containing only two pairs of gene representation for bit 1 and 3, is called an under-specification because bit 2 is missing. An under specified string is evaluated by filling the missing bit with a corresponding bit value from a full-length string — a competitive template. An initial competitive template can be randomly generated and replaced by the best string found in later generations. The latter coding set, consisting of four pairs of gene representation, is called an over-specification because more than one value is given for bit 3. A redundant bit value is removed by following a *first-come-first-served* rule scanning from left to right. The scanning rule together with a full-length competitive template enables the messy GA to evaluate both under and over specified strings. It provides the messy GA a maximum flexibility at varying the string length to uncover better-fit short strings — building blocks.

Building blocks filtering

The power of a genetic algorithm is its capability of searching for and grouping together building blocks — short strings (or partial solutions) with greater (or above-average) fitness. The messy GA emphasizes on uncovering building blocks before grouping them together for a better solution. After generating an initial population of strings with a problem length of l, a messy GA identifies building blocks of a certain length (or order) by randomly deleting gene bits in a string. The length of the string is subsequently reduced to a desired length. The process of detecting good building blocks is called *building block filtering*.

Building block filtering offers a way of gradually detecting building blocks of order-k from the strings of l'-length ($l' \leq l$). During this phase, a string is first selected by a thresholding selection (explained in a later section), then the genes are randomly deleted to reduce the string length and the new string with the remaining genes is evaluated. As given in Figure 4, an iteration of selecting strings and deleting genes continues until the string length is reduced to a desired order of building blocks. The gene deletion rate, the number of genes being deleted in each iteration of a building block filtering loop, has to be chosen such that it is on average less than the rate at which better strings get more copies by selection. Good results have been obtained for the numerical experimental testing of the fast messy GA by using a deletion rate of 0.5 (Goldberg et al. 1993). It means that 50

percent of the current genes are randomly deleted from the selected strings, which reduces the string length to just half of the previous string length. These shortened strings are then evaluated and the same procedure of the selection and gene deletion are applied until the string length is near the order *k* of the required building blocks.

Thresholding selection

Since the messy GA allows variable length strings to be processed, comparing two strings without a gene bit from any common gene locus or bit tag is meaningless. For example, for a 5-bit problem, the strings ((1,1) (2,0)) and ((3,1) (5,0) (4,1)) can be selected to participate in a tournament competition, but comparing both strings does not make a sense because there are no bits specified for the same locus. Thresholding selection was introduced to ensure that strings compete with each other only when they contain some genes from the same gene locus or with the same tags. A similarity measure θ is used to denote the number of common genes in two strings. In practice, a tournament selection is held, where two strings are allowed to compete with each other if the number of θ genes from the matching tags is greater than a prescribed threshold value given as (Goldberg et al. 1989):

$$\theta = \left\lceil \frac{l_1 l_2}{l} \right\rceil \tag{19}$$

where l_1 = the length of the first string, l_2 = the length of the second string and l = the problem length. For example, for a 10-bit problem (l = 10), string ((1,0) (5,0) (3,1)) (l_1 = 3) and string (((1,1) (3,0) (5,0) (6,1)) (l_2 =4) can be selected, a threshold value $\theta = \lceil 12/10 \rceil = 2$ is required for the two strings to participate in tournament selection. The number of the common genes in the two strings is 3, greater than the required threshold number of genes, thus they are allowed to compete each other by the thresholding selection.

Cut and splice operators

The crossover operator used in a simple GA cannot be applied to variable length strings in a messy GA. Two operators, *cut* and *splice*, have been designed and are used for a messy genetic reproduction. *Cut* acts to cut a chromosome into two, while *splice* links or concatenates two chromosomes to form one individual. If *cut* and *splice* are called in turn and applied to two strings, both operators work in a similar way to one point crossover operator in a simple genetic algorithm. The cut operator is activated by the *cut* probability given as:

$$P_C = P_k(\lambda - 1) \quad (20)$$

where P_k is the specified bit-wise cut probability and λ is the length of the string. The *splice* is initiated by a prescribed probability P_S that is taken as a constant value for the messy GA optimization. The *cut* probability for a string is defined as a linear function of a string length as above. It increases as a string length increases. During the early stages of a juxtapositional phase, strings are short, and the *cut* probability is low, consequently a *cut* operation is unlikely to be invoked. A *splice* operation is more likely to be applied at this stage. Thus strings grow in length. However, the longer a string grows, the higher the *cut* probability becomes and the more likely the string is cut. The length of strings remains within a certain range when a *cut* probability is about the same as a *splice* probability.

INTEGRATED SOLUTION METHODOLOGY

The optimal design and rehabilitation problem is solved by seamlessly integrating the messy GA with a hydraulic network solver EPANET (Rossman 1994). The messy genetic algorithm is employed as a solution seeker while EPANET is used as a hydraulic network simulator solving the system hydraulic equations for each trial. First of all, a string is converted to a design and rehabilitation solution by following the string and solution representation scheme, namely mapping the sub-string values to the index of possible decision choices by Eq.(12) and (13). The network solver may be called to perform hydraulic simulations for single or multiple demand loading conditions. Hydraulic results such as junction pressures, flow velocities and hydraulic gradients (slopes) are then passed back to the genetic algorithm module and checked against the design constraints given by Eq.(7)-(10). Subsequently, the maximum design constraint violation can be found for all demand loading cases. The actual cost of a design and rehabilitation trial is calculated by Eq.(3). In addition, a penalty cost is computed when a design constraint is violated. The total cost for the solution is the sum of an actual design and rehabilitation cost and a penalty cost. Finally the fitness for the string is given by Eq.(14) using the information of the total cost. The messy genetic algorithm, employing the fitness as surrogate measure of solution optimality, searches for the optimal design and rehabilitation solution.

CASE STUDIES

The messy genetic algorithm optimization methodology is applied to three case studies, a two-reservoir system, the New York City tunnels problem in literature and one real water distribution system in Morocco. The results obtained are presented below.

Two-reservoir network

As described earlier, the design task for a two-reservoir network is to determine the least cost combination of rehabilitation actions for three existing pipes and pipe diameters for five new pipes while the junction pressures are required to satisfy the minimum pressure. Previous studies (Simpson et al. 1994; Simpson & Goldberg 1994) applied the simple genetic algorithm and identified the global optimal solution of $1.7503 million for the two-reservoir network problem. The same optimal solution was found by using the messy GAs. The performance of the messy GA and the simple GAs is summarized and compared by the statistical results over ten computer runs. Table 4 shows that the messy GAs found the lowest cost solution (global optimum) in each of the 10 runs with different random seeds. The original messy GA using building block enumeration required only one third to half of the evaluation numbers of the simple GA using roulette wheel selection (Simpson et al. 1994), and also less than the simple GA with tournament selection (selection pressure s = 2). Simpson and Goldberg (1994) observed that increasing tournament pressure (s = 5) for the simple GA could reduce the number of evaluations, and thus improve the search efficiency, but too much pressure (s =20) might lead the search to a local optimum. Overall, the fast messy GA, using building block filtering, has further reduced the number of the evaluations and has been shown the most efficient at solving this small problem.

The New York city tunnels problem

The New York city water tunnels problem was posed by Schaake and Lai (1969). Figure 5 shows the layout of the system as in 1969. It consists of one water supply source at Hillview reservoir, and two main city tunnels named City Tunnel No. 1 and City Tunnel No. 2. The objective is to determine if a new pipe is to be laid parallel to an existing pipe and the diameter of a parallel pipe while the system is required to provide minimum hydraulic grades. This problem has been previously studied by a number of researchers in literature (Gessler 1982; Bhave 1985; Morgan and Goulter 1985; Quindry et al. 1981; Fujiwara and Khang 1990; Savic and Walters 1995; Dandy et al. 1996 and Lippai et al. 1999). The messy GA approach was applied to demonstrate its performance to the optimization of the New York city water tunnels system. A binary coding scheme has been used for the messy GA optimization. Four bits providing sixteen choices were used to code the possible sizes for each pipe.

There are fifteen choices of new pipe sizes in Table 9. The sixteenth choice was encoded as 0000 for a parallel pipe of zero-diameter namely leaving an existing pipe as it was. A total of eighty-four binary bits were used to represent the New York water tunnels optimization problem.

In order to compare the performance of the messy GA with the improved GA results the same Hazen-Williams equation as used by Dandy et al. (1996) was adopted as:

$$h_f = 4.7291L\left(\frac{Q}{C}\right)^{1.852} D^{-4.8704} \qquad (21)$$

The messy GA was run several times with different penalty factors applied for the constraint violation of the minimum required hydraulic grades. A set of low cost solutions obtained by the messy GA is compared with the results by the improved GA (Dandy et al. 1996) in Table 5. The corresponding diameters for each solution are given in Table 6. It shows that the cost of the optimal or near optimal solutions found by the messy GA are very similar to the improved GA, however, the messy GA is more efficient than the improved GA at searching for the lower cost design solutions. The improved GA required an average of 143,790 evaluations over five GA runs to reach the optimal or near-optimal solution. In contrast, the messy GA evaluated an average of 48,427 solutions over five messy GA runs to achieve similar solutions. The number of evaluations required by the messy GA is about one third of the evaluations required by the improved GA for this case study. Figure 6 compares a typical convergence rate of the messy GA solution with the improved GA for the optimization of New York city tunnels problem. It is demonstrated that the messy GA approach has significantly improved the computation efficiency for this particular case study.

A Moroccan network

A real water distribution system, as shown in Figure 7, is for a town of 50,000 inhabitants in Morocco. This network consists of one hundred and fifteen nodes, one hundred and fifty-eight existing pipes to be rehabilitated and nine new pipelines to be designed (or sized) for the system. Four possible rehabilitation actions, including replacing a pipe, relining a pipe, duplicating a pipe and leaving a pipe as it is, can be applied to the rehabilitation of the existing pipes. The problem has been studied by Hahal et al. (1997) using a multi-objective genetic algorithm approach. The fast messy genetic algorithm has been applied to solving this problem for a set of lower cost solutions.

The optimization of the Moroccan network has been specified as determining the least cost solution of the rehabilitation action for each of one hundred and fifty-eight existing pipes and the diameter for each of nine new pipes while satisfying the minimum required junction pressure of 20 meters. Apart from the cost associated with the rehabilitation actions, a repair cost is assumed to the pipe without taking an actual rehabilitation action or being assigned the action of leaving the pipe as it is. It is also assumed that no annual repair cost occurs to a new pipe during its first 10 years, as a new pipe is usually under warranty for this period of time. The repair cost is calculated as follows (Hahal et al. 1997).

$$C_{rep}(j) = \sum_{t=tp}^{t=tr} \frac{J(t)c_{rep}(j)}{(1+r)^{t-tp}} \tag{22}$$

where $c_{rep}(j)$ = repair cost of a breakage for pipe *j;* r = interest rate; tp = present year; tr = year tp + 10; and $J(t)$ = breakage rate in year t, which is given as:

$$J(t) = J_o(1+b_r)^t \tag{23}$$

where J_o = break rate in year 0 (break/km/yr); b_r = break rate growth coefficient and t = time in years.

The messy GA used a binary representation for solving the optimization of the Moroccan network. Two bits have been used for coding the four rehabilitation actions and three bits have been used for coding the eight pipe sizes for each of 158 existing pipes. Three bits have been used for coding the eight pipe sizes for each of the nine new pipes. Thus 817 binary bits are used for representing one solution of the Moroccan network. A number of different penalty factors were used for the optimization of the Moroccan network. Table 7 summarizes the least cost rehabilitation solutions for eight different penalty factors. The results show that there are slight differences in cost among the lower cost solutions obtained by using penalty factors from \$550,000 to \$750,000 per meter of the excess of a junction pressure head. The greater the penalty factor that was used in a messy GA run, the greater the cost of the best solution was found. This was due to the large penalty factor that forced the genetic algorithm search towards the feasible solution region. The genetic algorithm operations tended to reproduce more solutions within the feasible solution region than the infeasible region. It helps to ensure the feasibility of the optimal solution, but requires more evaluations to reach the optimal solution as shown in Table

8. However, the different optimal solutions provide engineers and/or decision-makers with more options to choose the optimal rehabilitation strategy by using other non-quantifiable engineering criteria.

A simple GA using binary strings, tournament selection (S = 5), uniform crossover and mutation was also applied to solving this problem. It was noticed that the simple GA was hardly able to find a good solution on such a large-scale optimization problem. The simple GA was run with a population size of 1500 and a maximum generation of 5000. The best solution of about $6.5 million was found at the first of 300 generations and hardly improved to the end of 5000 generation. Figure 8 gives a comparison of the convergence rates of the simple GA with the messy GA. It shows that both the messy GA and the simple GA start with a similar cost of initial design solutions, but the messy GA rapidly improved the design and rehabilitation solutions from about $9.0 million to near-optimal solution of approximately $1.1 million over 600,000 evaluations. The total number of possible solutions for the design and rehabilitation of the Moroccan network is about 2^{817}, approximately 8.74×10^{245} solutions. A complete enumeration of this solution space would consume an astronomical number of centuries of CPU time even if hundreds of trillions of objective evaluations can be done every second (the fastest computer up to date performs 3.9 trillion operations per second). The messy GA identified lower cost or near-optimal solutions by evaluating about 600,000 trials. The success of applying the messy GA to the design and rehabilitation of the Moroccan network represents one of the largest-scale optimization problems of this type.

CONCLUSION

Optimization of the design and rehabilitation for a water distribution system is improved by a comprehensive formulation of optimizing all system components, a unified genetic algorithm formulation and application of the full features of the messy genetic algorithm. The optimization model is extended to take into account all system elements including pipes, tanks, pumps and valves. The string and solution representation scheme, and the fitness formulation define two key steps for applying a genetic algorithm to solving the network optimization problem. The application of the messy genetic algorithm provides the most efficient search method for locating the least cost solution. In this way the computation efficacy is enhanced for optimizing almost any components of a water distribution system.

Optimization of a water distribution system is a non-linear optimization problem. This type of problem has been studied previously by applying many different optimization techniques including genetic algorithms. One of the main benefits by the genetic algorithm optimization approach is attributed to the integration of a genetic algorithm with a hydraulic network solver. The hydraulic solver is called for each design trial and solves

for pipe flows and junction pressures. This approach is able to cope with steady state or EPS simulations for either a single or multiple demand loading conditions. It makes the best use of well-developed network simulation techniques and optimizes a partial or entire system with any combination of system elements including pipes, tanks, pumps and valves. It provides the maximum flexibility to the cost-effective design and rehabilitation of a water distribution system.

A simple genetic algorithm is effective at solving a water pipeline optimization problem, but the difficulty at searching for optimal or near-optimal solutions increases as the dimension of the problem increases. Thus previously developed GA techniques are limited to the optimization of a relatively small water distribution system. The messy GA has significantly improved the efficacy of genetic-evolutionary computation. It uses an adaptive string representation of the solutions to a specific problem and focuses on searching for the short strings with above-average fitness — building blocks. The original messy GA suffered from the bottleneck of explicitly enumerating building blocks. It was overcome by introducing a building block filtering procedure that adaptively identified better-fit short strings. The messy genetic algorithm approach has been tested on two benchmark problems of water pipeline design and rehabilitation. The results obtained demonstrate that the messy GA consistently outperforms other GA paradigms. The application of the integrated messy GA technique to the optimal design and rehabilitation of the Moroccan water distribution system particularly shows its capability of optimizing a large-scale water distribution system. It is therefore concluded that the messy genetic algorithm provides a competent approach for the optimization of a water distribution system. The approach allows the least cost solution to be located more efficiently. It enables the optimal design and rehabilitation solution to be achieved for a large-scale water distribution system in a rapid manner.

ACKNOWLEDGMENTS

This paper is based on the first author's Ph.D thesis having been carried out under the supervision of Dr. Angus Simpson at the University of Adelaide. The financial support from the University is very much acknowledged for his research.

APPENDIX I. REFERENCES

Alperovits, E. and Shamir, U. (1977). "Design of water distribution systems." *Water Resources Research.* Vol. 13, No. 6, 885-900.

Bhave, P. R. (1985). “Optimal expansion of water distribution systems,” *J. Environmental Engineering,* ASCE, 111(2), 177-197.

Dandy G. C., Simpson A. R., Murphy L. J. (1996). “An improved genetic algorithm for pipe network optimization,” *Water Resources Research,* Vol. 32, No. 2, 449-458.

Eiger, G., Shamir, U. and Ben-Tal, A. (1994). “Optimal design of water distribution networks.” *Water Resour Res.,* 30(9), 2637-2646.

Fujiwara O. and Khang D. B. (1990). “A Two-Phase Decomposition Method for Optimal Design of Looped Water Distribution Networks,” *Water Resources Research.* Vol. 26, No. 4, 539-549.

Gessler, J. (1982). “Optimization of pipe networks.” *Proc., International Symposium on Urban Hydrology, Hydraulics and Sediment Control,* University of Kentucky, Lexington, Kentucky, 165-171.

Goldberg, D E. (1989). *Genetic algorithms in search, optimization and machine learning,* Addison-Wesley Publishing Company, Inc., 412pp.

Goldberg, D. E., Korb, B., & Deb, K. (1989). “Messy genetic algorithms: Motivation, analysis, and first results,” *Complex Systems, 3*, 493-530.

Goldberg, D. E., Deb, K., Kargupta, H., & Harik G. (1993). “Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms,” *IlliGAL Report No. 93004*, Illinois Genetic Algorithms Laboratory, University of Illinios at Urbana-Champaign, Urbana, IL 61801, USA.

Goulter I. C. (1987). “Current and future use of systems analysis in water distribution network design.” *Civ. Engng Syst.,* Vol. 4, 174-184.

Goulter I. C. (1992) “Systems Analysis in Water-Distribution Network Design: From Theory to Practice,” *Journal of Water Resources Planning and Management*, ASCE , Vol. 118, No. 3, 238-348.

Halhal, D, Walters, G. A., Ouazar, D. and Savic, D. (1997) “Water network rehabilitation with structured messy genetic algorithm.” *Journal of Water Resources Planning and Management*, ASCE, Vol. 123, No. 3, 137-146.

Holland, J.H. (1975). *Adaptation in natural and artificial systems,* University of Michigan Press, Ann Arbor, Michigan.

Karmeli D., Gadish Y. and Meyers S. (1968). “Design of optimal water distribution networks,” *J. of Pipeline Division,* ASCE, Vol. 94, No. PL1, pp. 1-10.

Kim, J. H. and Mays, L. W. (1994) “Optimal rehabilitation model for water-distribution systems.” *Journal of Water Resource Planning and Management,* ASCE, Vol.120, No.5, 982-1000.

Lansey, K. E. and Mays, L.W. (1989a). "Optimization model for design of water distribution systems." *Reliability analysis of water distribution systems*, L. R. Mays, ed., ASCE, New York, N. Y.

Lansey, K. E. and Mays, L.W. (1989b). "Optimization Model for Water Distribution System Design". *J. Hydr. Engrg.*, ASCE, Vol. 115, No. 10, 1401-1418.

Lippai, I, Heaney, J. P. & Laguna M. (1999). "Robust Water System Design with Commercial Intelligent Search Optimizers". *J. of Computing in Civil Engrg*, ASCE, Vol. 13, No. 3, 135-143.

Morgan D. R. and Goulter I. C. (1985). "Water Distribution Design with Multiple Demands," *Proc., Computer Application for Water Resources.,* ASCE, Buffalo, N. Y., 582-590.

Murphy, L.J. and Simpson, A.R. (1992). "Pipe optimisation using genetic algorithms," *Research Report No. R93*, Department of Civil Engineering, University of Adelaide, May, 53pp.

Quindry, G., Brill, E. D., and Liebman, J. C. (1981). "Optimization of looped water distribution systems," *J. Environmental Engineering Division,* ASCE, 107(EE4), 665-679.

Rossman, L. A. (1994) "EPANET user's manual," Risk Reduction Engineering Laboratory, Office of Research and Development, U.S. Environmental Protection Agency, Cincinnati, OH 45268.

Savic D. A. and Walters G. A.(1997). "Genetic Algorithms for the Robust, Least-cost Design of Water Distribution Networks," *J. Water Resour. Plng. and Mgnmt.,* ASCE, 123(2), 67-77.

Schaake, J. C. and Lai, D.(1969). "Linear programming and dynamic programming applications to water distribution network design." *Report 116,* Hydrodynamics Laboratory, Department of Civil Engineering, MIT, Cambridge, Massachusetts.

Simpson A. R., Dandy G. C., Murphy L. J. (1994) "Genetic algorithms compared to other techniques for pipe optimization." *Journal of Water Resource Planning and Management*, ASCE Vol. 120 No. 4, 423-443

Simpson A. R. and Goldberg D. E. (1994). "Pipeline optimization via genetic algorithms: from theory to practice." *2nd Int'I. Conf. on Pipeline Systems,* Edinburgh, Scotland.

Simpson A.R. & Wu Z.Y. (1997). "Optimal rehabilitation of water distribution systems using a messy genetic algorithm." *AWWA 17th Federal Convention Water in the Balance,* 16-21 March, Melbourne, Australia.

Su, Y. C., Mays, L.W., Duan, N., and Lansey, K.E. (1987). "Reliability-based optimization model for water distribution system." *J. Hydr. Engrg.,* ASCE, 114(12), 1539-1556.

Walski T. M. (1985). "State-of-the-Art Pipe Network Optimization." *Proc., Computer Application for Water Resources.,* ASCE, Buffalo, N. Y., 559-568.

Walski T. M. (1995). "Optimization and Pipe-Sizing Decisions". *Journal of Water Resource Planning and Management*, ASCE Vol. 121 No. 4, 340-343

Walters, G. A. (1988). "Optimal design of pipe networks: A Review." *Proc. 1st International conference on computer methods and water resources*, Vol. 2 Computational Hydraulics, Morocco, Edited by Ouazar D., Brebbia C.A. & Barthet H, Computational Mechanics Publications, Southampton Boston.

Wu Z. Y. and Simpson A. R. (1996). "Messy genetic algorithms for optimization of water distribution systems." *Research Report*, R140, Dept. of Civil and Envir. Eng., The University of Adelaide, Australia.

Wu Z.Y. & Larsen C.L. (1996). "Verification of hydrological and hydrodynamic models calibrated by genetic algorithms." *Proc. of the 2nd International Conference on Water Resources & Environmental Research*, Vol. 2, Kyoto, Japan, pp175-182.

Wu Z.Y. & Simpson A.R. (1997). "An efficient genetic algorithm paradigm for discrete optimization of pipeline networks." MODSIM 97, *Proc. of International Congress on Modeling and Simulation*, Vol. 2, Hobart, Tasmania, Australia, pp983-988.

Wu Z. Y., Boulos P.F., Orr C.H., and Ro J.J. (2000). "An efficient genetic algorithm approach to an intelligent decision support system for water distribution networks." To be appeared in *Proc. of Hydro Informatics 2000*, 23-27 July, Iowa, IA, U.S.A.

APPENDIX II. NOTATION

The following symbols are used in this paper:

$C(\vec{D}, \vec{E}, \vec{P})$ = cost of a design and rehabilitation solution;

$c_k(d_k)$ = cost per unit length of the *k*-th pipe diameter;

$c_k(p_k)$ = cost of the *k*-th pump capacity;

$c_k(t_k)$ = cost of the *k*-th tank size;

$c_k(v_k)$ = cost of the *k*-th pressure regulating valve size;

$c_k(e_k, d_k)$ = cost per unit length of the *k*-th rehabilitation action e_k for a pipe diameter of d_k.;

$c_{rep}(j)$ = cost of repairing a break for pipe *j*;

$\vec{D}$ = decision variables of new pipe diameter;

$\vec{E}$ = decision variables of pipe rehabilitation actions;

$H_{i,j}(t)$ = hydraulic head at junction *i* for demand loading case *j* at time *t*;

$H_{i,j}^{\min}$ = minimum required hydraulic pressures at junction *i* for demand loading case *j*;

$H_{i,j}^{\max}$ = maximum allowable hydraulic pressures at junction *i* for demand loading case *j*;

$HG_{i,j}(t)$ = hydraulic gradient (slope) of pipe *i* for demand loading case *j* at time *t;*

$HG_{i,j}^{\max}$ = maximum allowable hydraulic gradient of pipe *i* for demand loading case *j*;

$J(t)$ = pipe break rate in year *t;*

J_0 = pipe break rate in year 0;

$\vec{P}$ = decision variables of pump capacities;

$Pmin_k$ = minimum pump horse power;

$Pmax_k$ = maximum pump horse power ;

P_c = probability of c*ut* operator;

P_k = bit-wise cut probability;

P_s = probability of *splice* operator;

$\vec{T}$ = decision variables of tank sizes;

$\vec{V}$ = decision variables of valve sizes;

$V_{i,j}(t)$ = flow velocity of pipe *i* for demand loading case *j* at time *t*;

$V_{i,j}^{\max}$ = maximum allowable flow velocity of pipe *i* for demand loading case *j*;

$\overrightarrow{VS}$ = decision variables of valve settings;

$VSmin_k$ = minimum required valve setting for valve *k*;

$VSmax_k$ = maximum valve setting for valve *k*.

$Vtank_{i,k}^{in}$ = inflow to tank *k* under demand loading case *i*;

$Vtank_{i,k}^{out}$ = outflow from tank *k* under demand loading case *i*;

ΔV_k = flow balance tolerance of tank *k*;

λ = length of the string;

θ = number of common genes in the two strings;

Φ_{nn} = fitness of string *nn*;

Table 1 Binary string representation of possible rehabilitation actions for the two-reservoir network

Binary string	Rehabilitation action index	Possible rehabilitation actions
00	0	Leaving a pipe
01	1	Duplicating a pipe
10 or 11	2 or 3	Cleaning a pipe

Table 2 Binary string representation of available pipe sizes for the two-reservoir network

Binary sub-string coding the pipe size	Corresponding diameter index	Available pipe diameters (mm)
000	0	152
001	1	203
010	2	254
011	3	305
100	4	356
101	5	407
110	6	458
111	7	509

Table 3 A mapping from the sample string to its corresponding design and rehabilitation solution

Pipes	Variables	Sub-string	String values	Solution values
New pipe 1	Diameter	111	7	509
New pipe 2	Diameter	101	5	407
New pipe 3	Diameter	110	6	458
New pipe 4	Diameter	001	1	203
New pipe 5	Diameter	010	2	254
old pipe 1	Action	00	0	No change
	Diameter	111	7	No change
old pipe 2	Action	01	1	Parallel
	Diameter	101	5	407
old pipe 3	Action	11	2	Clean
	Diameter	010	2	No change

Table 4 Statistic results over ten runs of different genetic algorithm paradigms for optimization of the two-reservoir network

Genetic algorithm paradigms	Simple GA (Simpson et al. 1994; Simpson & Goldberg 1994)				Messy genetic algorithm	
	Roulette selection	Tournament selection			Building block enumeration	Building block filtering
		S = 2	S = 5	S = 20		
Success runs (out of 10 runs)	6	10	10	9	10	10
Avg. evaluations	14,697	8,800	4,300	2,900	6,181	2,400
Avg. least cost ($million)	1.7773	1.7503	1.7503	1.7600	1.7503	1.7503

Table 5 Results of the messy GA runs compared with the improved GA

Improved GA (Dandy et al.1996)			Messy GA			
GA runs	Lowest cost ($million)	Achieved at Evaluations	mGA runs	Penalty factor ($/ft)	Lowest cost ($million)	Achieved at evaluations
1	38.80	96,750	1	9,000,000	38.80	49,587
2	39.06	137,400	2	13,000,000	39.06	42,787
3	38.80	151,400	3	11,000,000	38.80	48,387
4	39.06	145,700	4	15,000,000	40.17	53,187
5	39.17	187,700	5	7,000,000	38.64*	48,187
	Average =	143,790			Average =	48,427

* The hydraulic pressure constraint at the junction 15 is violated by 0.02 (ft).

Table 6 Comparison of messy GA designs with previous GA solutions

Duplicated Pipe No.	Optimal diameters (in.)					
	Improved GA (Dandy et al. 1996)			Messy GA		
	GA 1	GA 2	GA 3	mGA 1	mGA 2	mGA 3
[1]	0	0	0	0	0	0
[2]	0	0	0	0	0	0
[3]	0	0	0	0	0	0
[4]	0	0	0	0	0	0
[5]	0	0	0	0	0	0
[6]	0	0	0	0	0	0
[7]	0	144	156	0	144	144
[8]	0	0	0	0	0	0
[9]	0	0	0	0	0	0
[10]	0	0	0	0	0	0
[11]	0	0	0	0	0	0
[12]	0	0	0	0	0	0
[13]	0	0	0	0	0	0
[14]	0	0	0	0	0	0
[15]	120	0	0	120	0	0
[16]	84	96	96	84	96	96
[17]	96	108	96	96	108	96
[18]	84	72	84	84	72	84
[19]	72	72	72	72	72	72
[20]	0	0	0	0	0	0
[21]	72	72	72	72	72	72
Cost ($million)	38.80	39.06	39.17	38.80	39.06	38.64*

* The hydraulic pressure constraint at the junction 15 is violated by 0.02 (ft).

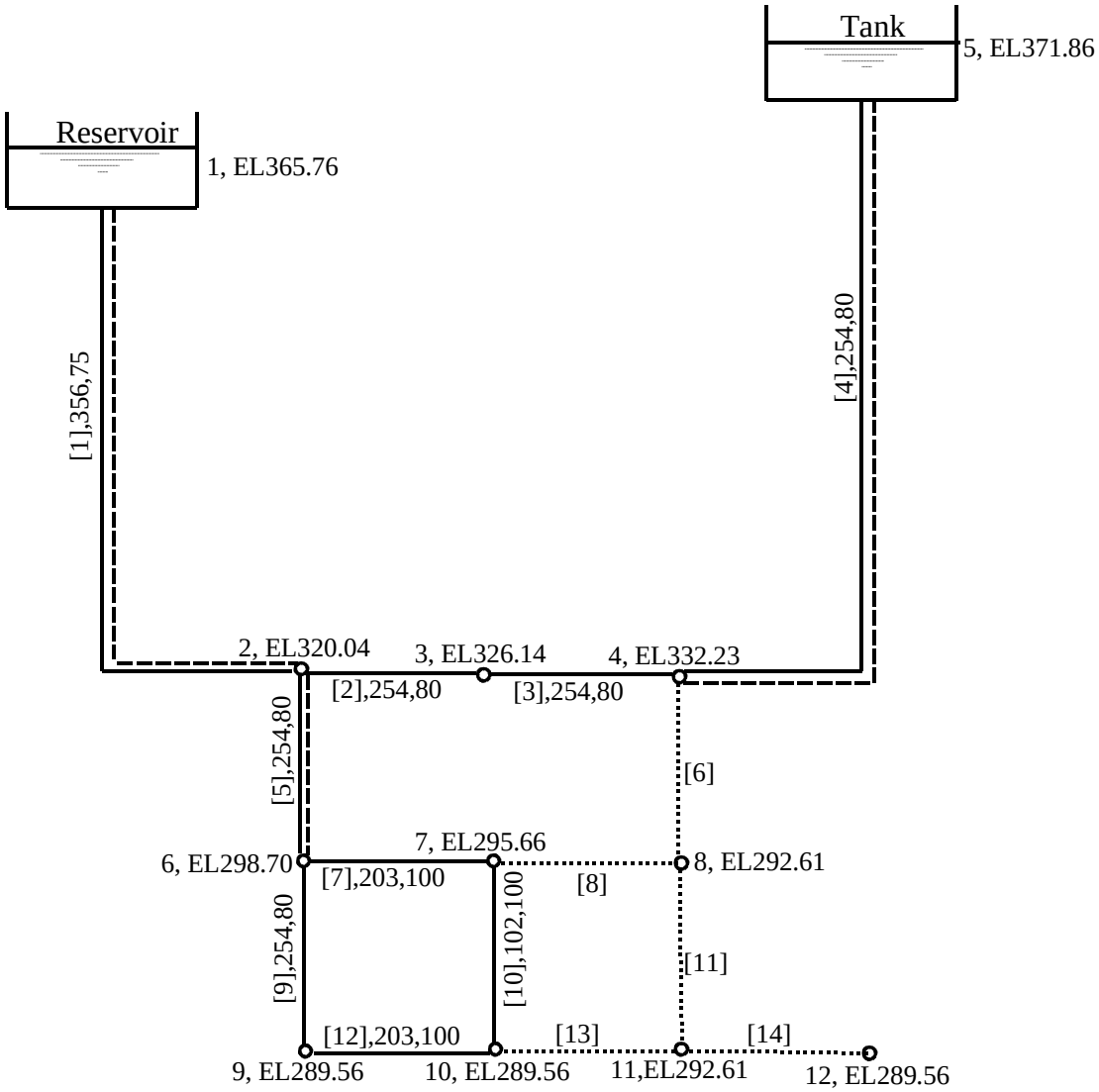
Table 7 Cost of optimal rehabilitation strategies of the Moroccan network

Messy GA solutions	Penalty factor	Repair cost	Relining cost	Replace-ment cost	Duplication cost	New pipe cost	Total cost
fmGA1	550,000	235,352	41,600	29,020	497,680	309,200	1,112,852
fmGA2	750,000	221,087	105,900	0	519,150	301,100	1,147,237
fmGA3	500,000	215,266	82,500	14,740	526,100	309,200	1,147,806
fmGA4	700,000	230,839	79,500	21,600	518,180	309,200	1,159,319
fmGA5	600,000	224,083	54,600	0	606,480	309,200	1,194,363
fmGA6	650,000	195,863	173,300	12,100	515,880	309,200	1,206,343
fmGA7	800,000	158,192	277,200	79,120	543,570	326,000	1,384,082
fmGA8	1,000,000	213,990	67,510	22,400	614,380	477,500	1,395,780

Table 8 Pressure heads and *excess* at critical nodes of Moroccan network (EPANET)

Optimal solutions	Node 59		Node 69		Node 87		Node 111		Evaluations achieved
	head (m)	excess (m)	head (m)	excess (m)	head (m)	excess (m)	head (m)	excess (m)	
fmGA1	20.09	0.09	20.01	0.01	20.13	0.13	20.17	0.17	630,290
fmGA2	20.40	0.40	20.24	0.24	20.02	0.02	20.45	0.45	571,290
fmGA3	20.92	0.92	20.18	0.18	20.03	0.03	20.08	0.08	566,290
fmGA4	20.54	0.54	19.99	**-0.01***	20.02	0.02	20.00	0.00	586,290
fmGA5	20.13	0.13	20.017	0.017	20.01	0.01	20.87	0.87	430,290
fmGA6	20.07	0.07	20.00	0.00	20.20	0.20	19.99	**-0.01**	575,290
fmGA7	20.54	0.54	20.12	0.12	20.02	0.02	20.25	0.25	599,290
fmGA8	20.24	0.24	20.04	0.04	20.48	0.48	21.42	1.42	906,970

* A negative value implies the pressure deficit.



Pipes: [1],356,75 [pipe number], diameter(mm), Hazen-Williams roughness C

Note. 1. All pipe lengths are 1609m, except pipe[1]=4828m and pipe[4]=6437m.

2. C=120 for new pipes and cleaned pipes.

Nodes: 2, EL320.04 node number, node elevation(m)

Figure 1 Two-reservoir Network

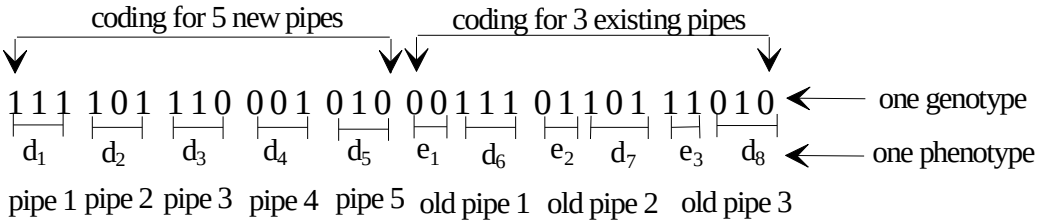


Figure 2 A sample string representation of one possible solution for the design of five new pipes and rehabilitation of three existing pipes

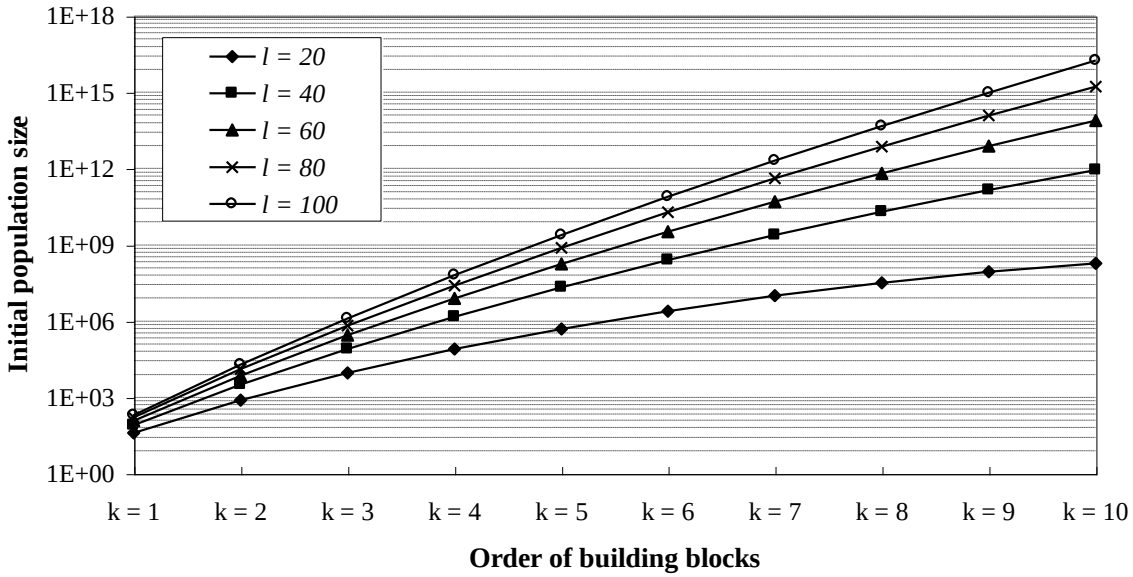


Figure 3 Initial population sizes required by original messy GA using complete enumeration

```
era = 0;
while (era < max era ){
        initializing population;
        evaluating population;
        while ( current string length > desired building block length){
                //building block filtering phase;
                selecting competitive strings;
                deleting genes;
                evaluating new strings
        }
        generation = 0;
        while (generation < max generation ){
        //juxtapositional phase;
                selecting competitive strings;
                cut and splice operations;
                mutation operations;
                evaluating new population;
                generation = generation + 1;
        }
        era = era + 1;
 }
```

Figure 4 A summary of the steps in a messy genetic algorithm

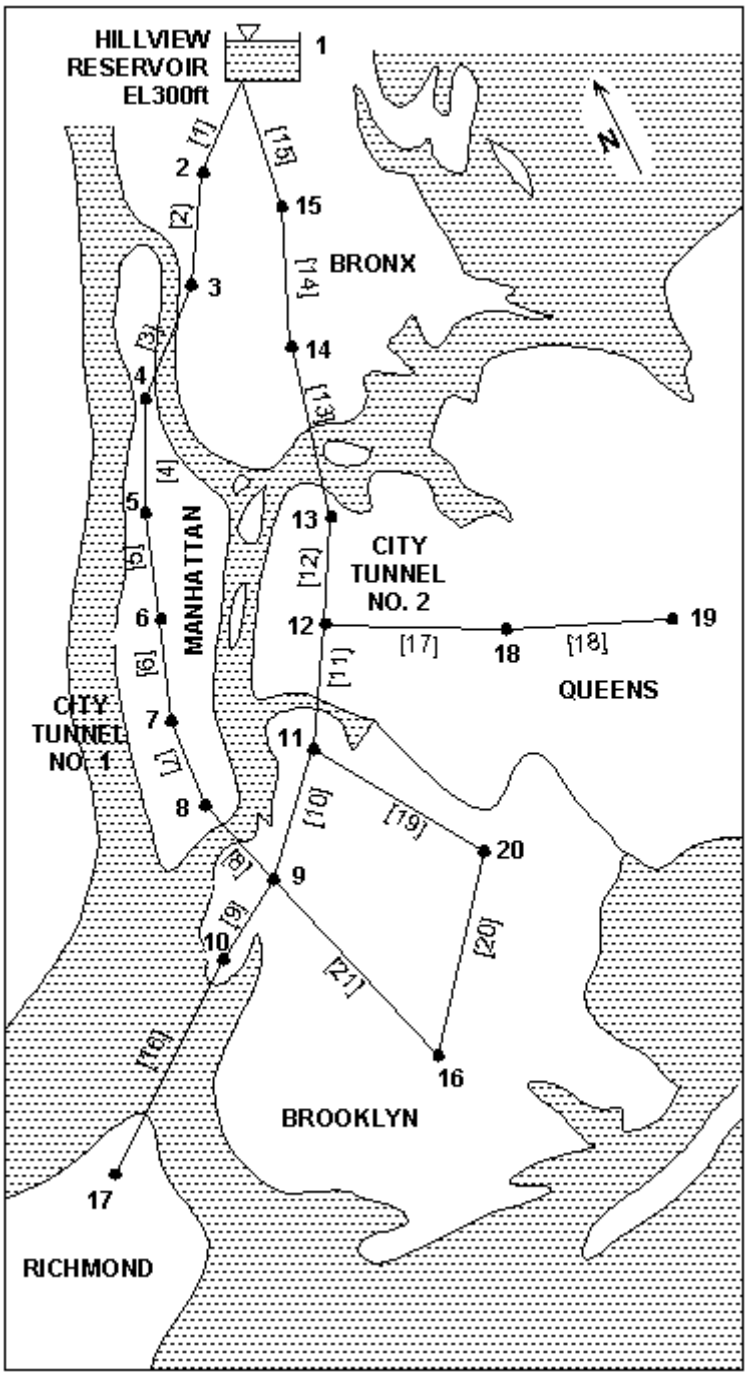


Figure 5 New York City water supply tunnels (Dandy et al. 1996)

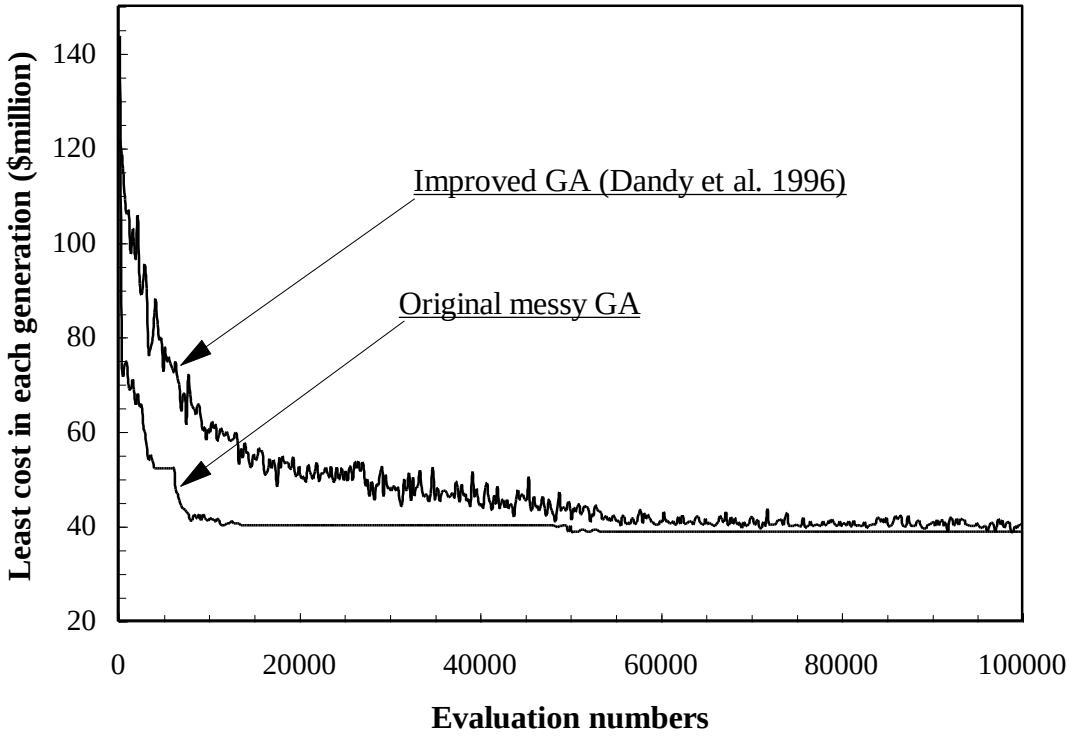


Figure 6 Comparison of convergence rates of the improved GA (Dandy et al. 1996) and the messy GA for the New York water tunnels problem

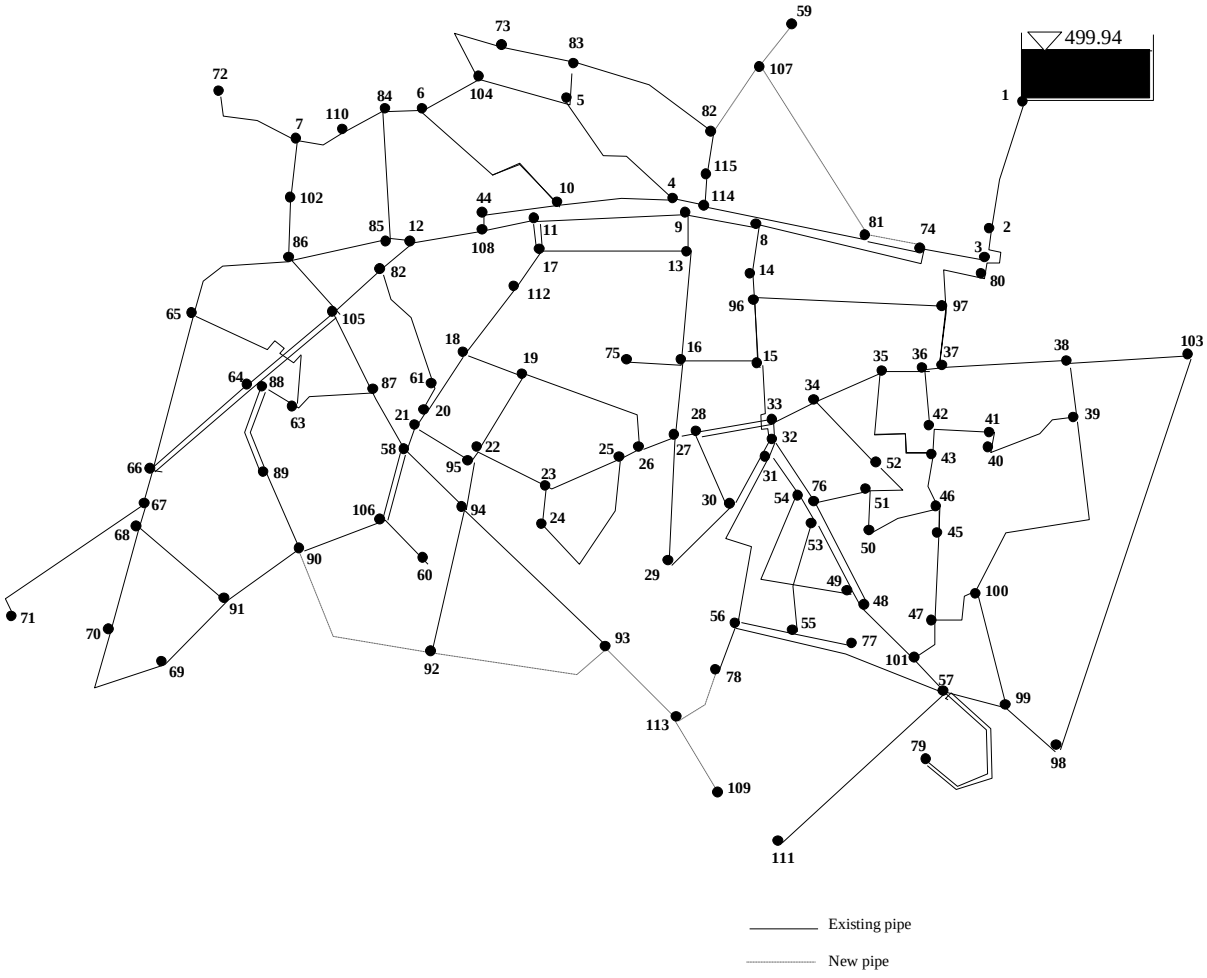


Figure 7 Layout of a Moroccan network

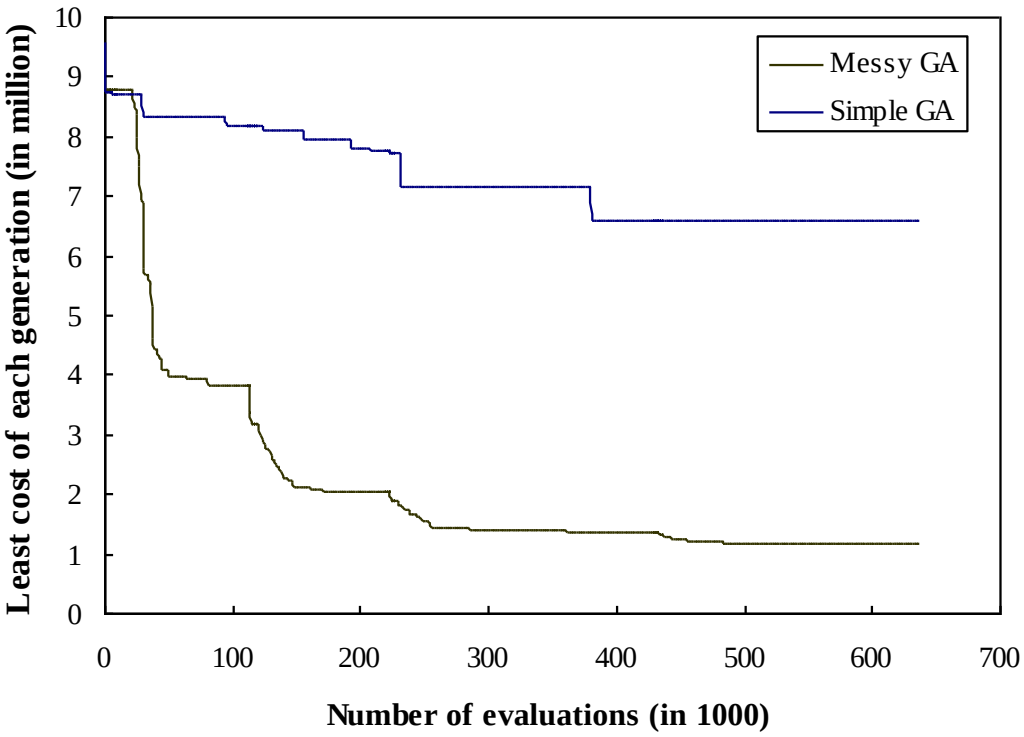


Figure 8 Comparison of convergence rates of the simple GA and the messy GA for the optimization of the Moroccan water distribution system