Contribution of parallel NSGA-II in optimal design of water distribution networks

Sandro Artina, Cristiana Bragalli, Giovanni Erbacci, Angela Marchi and Marzia Rivi

ABSTRACT

Optimization of water distribution networks is a NP-hard problem that researchers have tried to deal with using different formulations and algorithmic approaches. Among these, multi-objective heuristic algorithms are interesting because of their capacity for dealing with separate objectives that allow us to choose *a posteriori* the best compromise, but one of their main drawbacks is the long time required to obtain good solutions. Parallel processing is the most promising way to reduce the computing time and can make the convergence to adequate solutions faster. This paper intends to investigate the possibility of improving the efficacy and efficiency of an NSGA-II algorithm by parallelization of the optimization process at the same time. Results of different parallel implementations of NSGA-II applied to optimal design of small- and medium-size water distribution networks are presented. Good speed-up can be reached with a global model, hence improving the algorithm efficiency. Unlike the global model, the island model (or the hierarchical parallelization) can also improve its efficacy because it introduces fundamental changes in the algorithm exploration method. Possibilities offered by parallel island models have been investigated showing that some parameter configurations can find better solutions compared with the serial version of the algorithm.

Key words | high performance computing, island parallelization, NSGA-II algorithm, optimal design, speed-up, water distribution networks

Sandro Artina
Cristiana Bragalli (corresponding author)
Angela Marchi
DISTART,
University of Bologna,
Viale Risorgimento 2,
I-40136 Bologna,
Italy
E-mail: *cristiana.bragalli@unibo.it*

Giovanni Erbacci
Marzia Rivi
CINECA,
Interuniversity Consortium,
Via Magnanelli 6/3,
I-40033 Casalecchio di Reno,
Bologna,
Italy

ABBREVIATIONS AND NOTATION

$c(D_p)$	unit cost of pipe p
CW_p	Hazen–Williams roughness coefficient of the pipe p
D	set of diameters
D_p	diameter of pipe $p \in D$
G_t	parent solutions at generation t
L_p	length of pipe p
N	set of nodes
H_i, H_j	hydraulic head of node i ($i \in N$)
$H_{i,\min}$	minimum hydraulic head required at node i ($i \in N \backslash S$)
P	set of pipes
$Q_{e,i}$	demand or inflow at node i
Q_p	flow in pipe $p \in P$
S	set of source nodes
β, γ, ω	hydraulic parameters of energy conservation equation
$\delta_+(i)$	set of pipes with tail at node i ($i \in N$)
$\delta_-(i)$	set of pipes with flow leaving node i ($i \in N$)
c_{d}	crowding distance
f_{ex}	migration frequency
m	number of islands
n	number of processors
NGen	number of allowed generations
OFC	cost objective function
OFP	pressure objective function
p_{c}	crossover probability
P_{ex}	percentage of migrating solutions in an island

doi: 10.2166/hydro.2011.014

P_0 initial population
P_t population at generation t
Pop number of total solutions of population
p_m mutation probability
Q_t offspring solutions at generation t
R_t population and offspring at generation t
t_s time of serial execution

INTRODUCTION

The optimization of water distribution networks (WDN) concerns the design and operation of different components, not just the pipes, but also tanks, pumps and valves subject to multiple working conditions. Given the practical complexity of WDN design, the problem has been simplified, leading to the classical *optimal design* formulation. This consists in assigning a diameter to each pipe, given the layout of the network, with the aim of finding the cheapest solution that satisfies pressure constraints (Alperovits & Shamir 1977). In this simplified version a unique load condition (demand) is assumed so that variations over an extended period are not taken into account. Furthermore aspects related to network reliability and water quality are not considered. This formulation represents a useful starting point both for solving the whole problem and providing some useful indications to designers. It also allows comparison with other investigations because of the vast literature on the subject.

From a mathematical point of view the optimal design is a NP-hard problem (Yates *et al.* 1984). To solve it, different methods have been developed and they can be divided into two main categories: exact algorithms (Schaake & Lai 1969; Kessler & Shamir 1989; Fujiwara & Khang 1990; Sherali *et al.* 2001; Bragalli *et al.* 2008) and heuristic algorithms. This last approach has the advantage of tackling only the combinatorial part of the problem, leaving the resolution of nonlinear constraints to a hydraulic solver. Some example of heuristic algorithms are: genetic algorithms (Goldberg & Kuo 1987; Savic & Walters 1997; Vairavamoorthy & Ali 2000; Wu & Simpson 2001; Reca & Martínez 2006; Kadu *et al.* 2008), simulating annealing (Cunha & Sousa 1999; Tospornsampan *et al.* 2007), ant colony (Maier *et al.* 2003; Zecchin *et al.* 2007a, b), harmony search (Geem 2006), shuffled frog-leaping algorithm (Eusuff & Lansey 2003), shuffled complex evolution algorithm (Liong & Atiquzzaman 2004), tabu search (Sung *et al.* 2007) and memetic algorithm (Baños *et al.* 2010).

The recent development of multi-objective algorithms allows us to consider and treat separately the problem objectives (Prasad & Park 2004; Perelman *et al.* 2009; Montalvo *et al.* 2010). This has the advantage of not requiring a unique objective function representing the combination of more objectives. Therefore the search in the solution space is not affected by the choice of weights used to define this type of function. Moreover this kind of algorithm produces a set of different solutions instead of a single one. In WDN design, this means that it is possible to choose *a posteriori* the best compromise between objectives. Multi-objective algorithms are also effective in introducing criteria not considered in the classic optimal design formulation also because of the complexity of describing them through mathematical expressions (Reca *et al.* 2008).

Besides the focus on the definition of problem objectives (Giustolisi & Berardi 2009), research interest is also devoted to the improvement of the algorithm's performance in order to design large-scale WDN efficiently. This means reducing the computation time and improving the exploration of the space of solutions. In fact, a large network size implies a longer time to compute hydraulic characteristics for each solution and the enlargement of the search space. This last point is particularly true for genetic algorithms (and then for multi-objective genetic algorithms) that stochastically explore this space, evolving a set of solutions (population) in analogy with natural evolution, where the best individuals survive and procreate. There are different possibilities to achieve this purpose, from the union of rules (Keedwell & Khu 2005) and classifiers (di Pierro *et al.* 2009) to the use of multiple processors on which we are running the optimizations (Sinha & Minsker 2007; Lopez-Ibanez *et al.* 2008; Tsai *et al.* 2009).

This paper presents results on a way to improve, at the same time, the efficacy and efficiency of the NSGA-II algorithm by parallelization of the optimization process. NSGA-II (Deb *et al.* 2000), described in the next section, is a multi-objective genetic algorithm (MOGA) that provides a trade-off between the objectives considered and

has a wide range of applications in water distribution network optimization (Atiquzzaman *et al.* 2006; Dridi *et al.* 2008; Prasad & Tanyimboh 2008; di Pierro *et al.* 2009).

Two different parallel models, the global and the island model, have been implemented and then combined in the hierarchical model. The third section contains a synthetic description of the these models and the fourth provides speed-up results achieved by each one. While the global model only reduces the computation time, the island model also affects the quality of solutions. Moreover the application of the last one is not simple because it requires the choice of adequate values for several parameters. Hence our investigation is mostly devoted to the island model in order to understand how different configurations of input parameters affect solutions of the optimal design problem. The fifth section reports results for a small- and a medium-size network (Hanoi and Modena networks). The final section describes the classical optimal design problem of WDN considered here to test the NSGA-II parallelization.

OPTIMAL DESIGN PROBLEM STATEMENT

The optimal design problem for a network of a given layout is described by an objective function of cost minimization OFC (1), subject to (2)–(5) constraints: hydraulic equations of mass (2), energy conservation (3) for each node and each pipe, respectively, operating constraints about minimum pressure maintenance (4) and a construction constraint (5), that specifies that diameters are discrete and belong to the commercial set:

$$\min \mathrm{OFC} = \sum_{p \in P} c(D_p) L_p \tag{1}$$

subject to:

$$\sum_{p \in \delta_+(i)} Q_p - \sum_{p \in \delta_-(i)} Q_p = Q_{e,i} \; (\forall \; i \in N/S) \tag{2}$$

$$H_i - H_j = \omega \frac{L_p}{CW_p^{\beta} D_p^{\gamma}} Q_p |Q_p|^{\beta - 1} \; (\forall p = (i, j) \in P) \tag{3}$$

$$H_i \geq H_{i,\min} \; (\forall i \in N \backslash S) \tag{4}$$

$$D_p \in D \tag{5}$$

where P is the set of pipes, N the set of nodes, S the set of source nodes, D the set of diameters, $\delta_+(i)$ the set of pipes with flow entering node i ($i \in N$), $\delta_-(i)$ the set of pipes with flow leaving node i ($i \in N$), Q_p the flow in pipe $p \in P$, H_i the hydraulic head of node i ($i \in N$), D_p the diameter of pipe $p \in D$, $H_{i,\min}$ the minimum hydraulic head required at node i ($i \in N \backslash S$). Moreover $c(D_p)$ is the unit cost of pipe p with length L_p and diameter D_p, $Q_{e,i}$ is the demand or the inflow at the node i; CW_p is the Hazen–Williams roughness coefficient of pipe p; β, γ and ω are parameters of the hydraulic equations.

To deal with NSGA-II, the respect of pressure constraints (4) is introduced as the minimization of the objective function OFP (6):

$$\mathrm{OFP} = \begin{cases} \max (H_{i,\min} - H_i) & \text{if } \exists i \in N : H_i < H_{i,\min} \\ -\min (H_i - H_{i,\min}) & \text{otherwise} \end{cases} \tag{6}$$

OFP can assume negative or positive values: when it is positive it means that some node in the network has a hydraulic head lower than required and the OFP value is the biggest pressure deficit. When OFP is negative, it means that all nodes have a hydraulic head bigger than required, and its value measures the minimum pressure surplus. So if, for example, OFP is -0.5, it means that the most depressed node in the network has a surplus of 0.5 in hydraulic head units with respect to the minimum required.

This OFP definition (6) has the advantage of maintaining several solutions with different degrees of hydraulic head surplus in the final optimal front as shown in Figure 6. This choice is justified by the fact that, from one point of view, it can be useful to know the cost-saving allowing small hydraulic head deficit (that maybe does not affect network functionality), while, from the other, it could be equally useful to know that the increase of the cost of solutions is bit more expensive but with higher hydraulic head.

NSGA-II ALGORITHM

Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb *et al.* 2000) is an elitist multi-objective evolutionary algorithm based on the concept of non-dominated solutions to drive the evolution of an initial population P_0 of Pop solutions randomly created, in analogy with evolution in nature. This optimization process provides a trade-off between the various objectives considered.

The population evolves for a fixed number of generations NGen. At generic generation t, the offspring population Q_t (size Pop) is created by using the parent population G_t (size Pop/2) selected using the tournament selection applied to the whole population P_t. The reproduction phase requires the probability of single-point binary crossover p_c and the probability of uniform binary mutation p_m. The populations are combined together to form the entire population $R_t = P_t + Q_t$ (size 2Pop). R_t is sorted according to two criteria to derivate the new population P_{t+1} (size Pop). Firstly, solutions are sorted in different fronts using non-domination criteria: in a minimization problem of m objectives, the solution a dominates the solution b, if a has at least one objective strictly minor of b, while all other objectives are minor or equal to those of b. Therefore it is natural preferring solutions with the minor rate of domination. When solutions belong to the same front, the surviving ones will be those with a larger crowding distance c_d, with the aim of promoting a homogeneous cover of the front (Figure 1).

Each individual (solution) is regarded as a string of P integer indexes. In the optimal design of WDN, each index represents a diameter value, according to its position in the list of available and allowable diameters of the set D. Hydraulic Equations (2) and (3) are solved by means of EPANET 2 (Rossman 2000) based on the gradient method (Todini & Pilati 1988), where hydraulic simulations are performed replacing each index with its corresponding diameter. The only constraint handled directly by the algorithm is the available set of diameters D (5). At the end of the whole process, the hydraulic solver is called Pop × NGen times. Usually it becomes a large number in order to obtain good results, in particular for large-size WDN to which the current interest is directed.

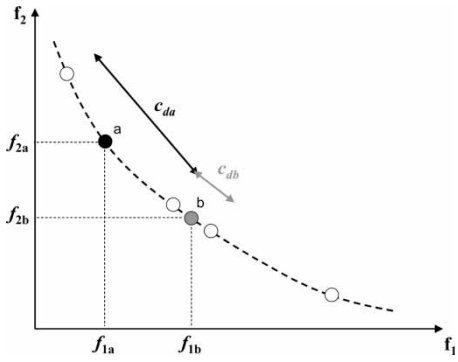

Figure 1 | Non-dominated solutions in a two-objectives minimization problem. Crowding distance of solution 'b' is smaller than that of 'a', so solution 'a' is preferable to 'b'.

PARALLELIZATION OF NSGA-II

MOGAs are good candidates for effective parallelization, given their inspiring principle of evolving in parallel a population of individuals. Parallelization means shorter computing times or, keeping the same whole time, a larger degree of search space exploration. The main sources of time-consuming performance for MOGA algorithms are two: the evaluation of the population in terms of objective functions and the non-dominated sorting of the solutions, both depending on the population size. The former source can be easily reduced by distributing the population evaluation between the different computing elements. This solution is described by the following parallelization model (*global model*): a unique population of solutions evolves as a serial MOGA so that selection and mating is done globally, but at each generation the fitness evaluation of solutions is distributed in a balanced way between processes organized in a master–slave structure (Figure 2(b)).

On the other hand, a way to cope with the second source, the population sorting, is to partition the population between several processors. In this case there is a crucial difficulty concerning the reproduction phase because the choice of new parents would require the availability of the entire population. Several approaches have been proposed to overcome this problem (Cantù-Paz 1997; Nowostawski & Poli 1999; Durillo *et al.* 2008; Risco-Martín *et al.* 2008). A fine-grained population approach (*cellular model*) is suited for massively parallel computers, such as GPUs and clusters of multicore CPUs, because it consists of one spatially structured

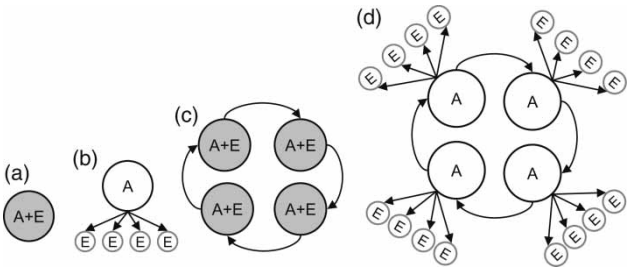

Figure 2 | Scheme of different implementations of parallel NSGA-II: (a) serial version, in which the NSGA-II operations (indicated by A) and the function evaluations (E) are carried out by a unique processor; (b) global parallelization in which only EPANET 2 calls are distributed among several processors, while the operations A as reproduction and sorting are still managed by one process; (c) island parallelization in which each process runs its own NSGA-II and evaluates its own solutions and (d) hierarchical implementation that differs from (c) because each island distributes EPANET 2 calls among different processes.

population partitioned into a large number of very small subpopulations. Hence selection and mating are restricted inside small neighbourhoods of individuals, but their overall interaction can be realized using overlapping neighbourhoods. Therefore good solutions may disseminate across the entire population. A coarse-grained multiple population approach (*island model*) is more appropriate for distributed memory architectures. In this case, the population is divided into a few subpopulations, called *islands*, that evolve serially and independently. Occasionally a migration phase occurs (Figures 2(c, d)), where some solutions are exchanged between islands according to some migration control parameters (Cantù-Paz 2007; Skolicki 2005; Defersha & Chen 2008). It is important to notice that, while the global model does not affect the final optimal front, the last two approaches introduce fundamental changes.

Herein attention will be posed on island parallelization whose choice seems more suitable for optimizing the NSGA-II algorithm. In fact, it maintains the same definition of population, and therefore the same procedures of parent selection and survival criterion as in the serial version: inside each island there are no neighbouring constraints for mating and competing. As multiple populations can evolve in different directions, the space of solutions can be better explored. Moreover, the introduction of some diversity by migration should produce a faster evolution of each island and therefore a convergence to optimal solutions should be reached in a fewer number of generations. From a computational point of view a good scalability of the code is also expected because this approach has a low communication cost as islands almost have no operations in common.

The implementation of this model requires particular care in the generation of random numbers because each island must have independent different streams in order to evolve correctly. This goal has been achieved by using the SPRNG library, an open-source scalable parallel random number generator library which provides several types of generators (Mascagni & Srinivasan 2000). In particular, the 64-bit Linear Congruential Generator has proved to be the fastest and most efficient in our tests. Moreover the island model is particularly difficult to use at best because it requires the tuning of several additional parameters that affect the efficiency and quality of the solutions. The most relevant ones are the size of the island population, the migration frequency f_{ex}, the percentage P_{ex} of migrating solutions and their destination (Cantù-Paz & Goldberg 1999). If the algorithm is multi-objective, it is also very important defining which are the migrating solutions (Khabzaoui *et al.* 2005). The way solutions are moved is a further degree of freedom: each island can send some of its individuals to all the other islands or only to some of them, following a specific geometrical pattern. In our implementation, we choose to connect islands in the migration phase by a ring topology for its simplicity (Figure 3): each island passes information to the next one and receives it from the previous one.

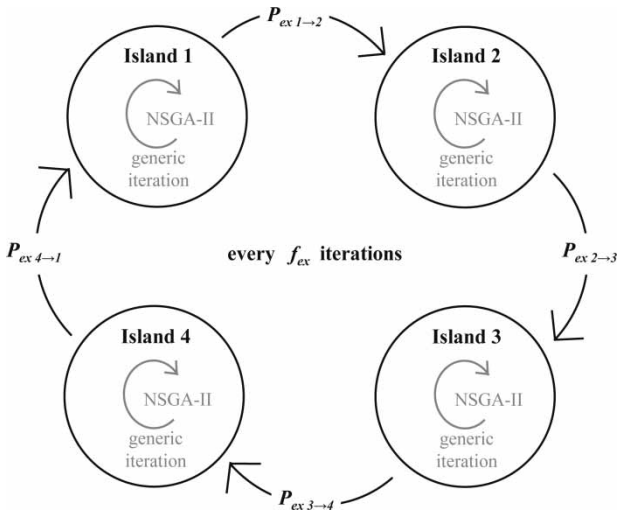

Figure 3 | Scheme of the four island parallelization implemented for the NSGA-II: $P_{ex\ 1\to2}$ means that solutions are moving from island 1 to island 2.

Whenever the evaluation of every single solution involves heavy calculations, in terms of time consumption, the structure of our code allows the speed-up to be improved by a hierarchical parallelization. This approach combines the island model with the global one in order for each island to distribute solution evaluations among several slave processes (Figure 2(d)). The number of slaves is assumed to be the same for each island and is called the *dimension* of each island.

In this case we have implemented two versions of the code that exploit the state-of-the-art supercomputer architecture as available at CINECA: a large amount of nodes (each of them counts several multicore CPUs that share the node's memory) are connected to each other by a fast, machine-wide, communication network. Therefore, we can map our model's structure to the underlying hierarchical hardware infrastructure by realizing each island on a different computing node: we adopt the 'master–slave' configuration for the processes within each island, by means of assigning the role of communicating to other islands to a 'master' process, that also collects the evaluations calculated by the slaves.

The implementation described above is based on the MPI library (http://www.mpi-forum.org), which represents the standard for communication among different processes within a parallel program running on a distributed memory system. In the *pure MPI version* both the master and the slaves are MPI processes that need to communicate with each other, whereas in the *hybrid version* the slaves are OpenMP threads that share access to the node's memory. OpenMP (http://www.openmp.org) is an Application Programming Interface implementing multithreading, a method of parallelization that exploits shared memory: the master process forks a specified number of 'slave' threads (each executed by a different core) in such a way that the 'workload' is divided among them. Each thread executes blocks of code running in parallel on different cores sharing the same global memory.

Note that the simple serial model, island model and global model now become particular cases of the hierarchical version (Figure 2(d)): the first one corresponds to 1 island of dimension 1 (Figure 2(a)), the second one realizes a structure of m islands of dimension 1 (Figure 2(c)) and the last one is an island of dimension greater than 1 (Figure 2(b)).

SPEED-UP RESULTS

Some benchmarks have been made in order to evaluate, for different configurations of the code, the computation time as the number of processors increases. Tests are run on the IBM Linux Cluster BCX/5120 of CINECA. It is equipped with 2560 AMD Opteron Dual Core 2.6 GHz processors and Infiniband interconnections, four cores per node and a memory of 8 GB/node.

We have chosen a medium-size network (Modena, described in the final section) and fixed the following parameters of the NSGA-II algorithm: crossover and mutation probability p_c and p_m are 0.6 and 0.0033. respectively, number of iterations NGen = 8,000 and population of size Pop = 1,600. All tests are executed using the pure MPI version of the hierarchical code. In our tests pure MPI and the hybrid version have equivalent performance as will be explained at the end of this section.

The first benchmark (Figure 4) refers to the global model (1 island of dimension n). In theory, if the evaluation of Pop solutions takes t_s seconds on one processor, we should expect that it takes t_s/n seconds on n processors/slaves. In this case the scalability of the code is said to be *linear*. However, the communication between processors (part of which is independent of the amount of data transferred) and the distribution of work must be taken into account. Therefore real times usually are slower than this theoretical limit. Moreover, in this case,

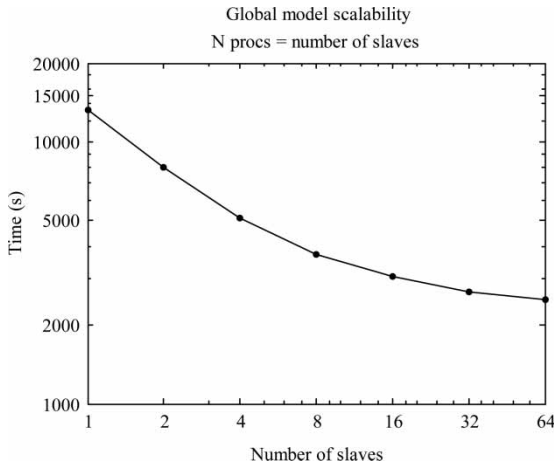

Figure 4 | Scaling of the CPU time (total wallclock time) with the number of slaves, plotted in logarithmic scale for the Modena network. A total of 8,000 iterations are executed.

only the fitness evaluation of individuals is distributed among processes while the rest of the code is still serial. Figure 4 shows that, with a small number of slaves, we get a good improvement of performance, but the master–slave structure of the algorithm generates a bottleneck problem of communication as the dimension of the island increases, reducing the scalability of the code. For the size of our problem, we see that the number of slaves must not be greater than 8 in order to get a speed-up, exploiting at best the computer resources without decreasing the rate of speed improvement.

The second benchmark (Figure 5) refers to the island model. In this case the population Pop is split among several islands that will evolve independently, with the same parameters (p_c, p_m, NGen). Every fixed number of generations, $f_{ex} = 200$, there is a communication phase in which 50 solutions are exchanged (P_{ex} varies accordingly to the island population size, given that Pop is fixed and the number of islands is modified). The choice of these migration parameters involves a relatively small number of communications, therefore they slightly affect the execution time of the code which is mostly determined by the size of the water distribution network instead. Figure 5 shows that this model has a linear scalability, clearly visible using the logarithmic scale. In fact, the load is well balanced among processes during all the execution and there is little communication. However, the number of islands to create depends on the population size because each island cannot have a subpopulation too small (a necessary condition for the algorithm to work correctly).

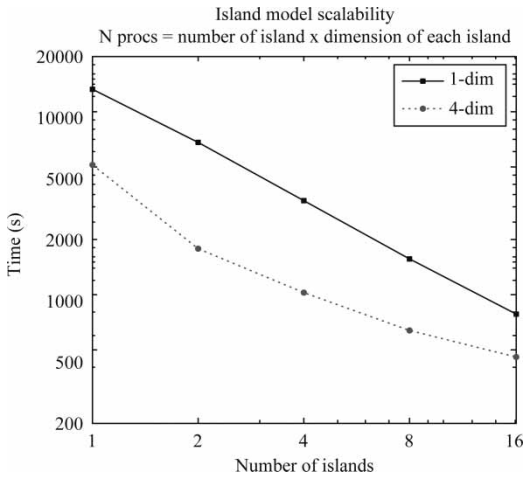

Figure 5 | Scaling of the CPU time with the number of islands plotted in logarithmic scale for the Modena network. A total of 8,000 iterations are executed; 50 solutions of each island are migrated every 200 iterations.

Figure 5 also shows how the computational time reduces by increasing the dimension of the island, for example fixing the dimension equal to 4 which corresponds to the number of cores for the computing node.

Other tests carried out corroborate the idea that the performances of the hybrid and pure MPI versions of the hierarchical model are equivalent (i.e. about the same time of execution). This is what we expected because in this code shared memory is used only for distributing individuals of the island population among threads (instead of making an MPI communication). Each thread does not take advantage of shared memory to compute the solution fitness because each evaluation corresponds to a simulation of a different WDN, so they behave as single processes. Moreover MPI communication is fictitious inside a computing node. Therefore it is not surprising that the two paradigms perform in the same time. However, the hybrid version significantly simplifies the source code!

COMPUTATIONAL RESULTS AND DISCUSSION

Different configurations of island parallel NSGA-II have been tested on Hanoi and Modena WDN in relation to the optimal design problem (1)–(6). The experiments are run with the same set of NSGA-II parameters for both networks: crossover and mutation probability parameters were assumed equal to $p_c = 0.6$ and $p_m = 0.0033$, respectively. Even if the choice of these parameters often depends on the problem dimension, here it has been preferred to set the same values and vary only the parameters related to the parallelization process in order to better compare the performance of serial and parallel NSGA-II versions, which is the actual purpose of this work. The optimization was carried out using four islands and different values of P_{ex} and f_{ex}. The parameter P_{ex} is defined with respect to the island population size: $P_{ex} = 2$ means that only 2% of island solutions are exchanged. Exchange frequency $f_{ex} =$ 10 means that every 10 iterations there is communication between islands.

In terms of the quality of solutions, the particular determinant is the criterion by which the migrant solutions are selected. In this work three modes have been tested:

- *Mode V1*: the migrating solutions are randomly selected inside the island population and moved towards the empty positions of the next island;
- *Mode V2*: the migrating solutions are chosen among the 'best' solutions in the islands. These solutions are moved into the next island replacing its 'worst' solutions;
- *Mode V3*: the choice of migrating solutions and the ones to be replaced is the same as in the previous version, but in this case there is a check to avoid importing already existing solutions.

The definitions 'best' and 'worst' refer to the purpose of the hydraulic problem: finding the cheapest solution that obeys pressure constraints. In this work, 'best' solutions are those that straddled the feasibility limit, i.e. those having OFP nearest to zero with values both positive and negative. If the number of solutions to be moved is, for example, 10, it means that the 5 cheapest feasible solutions and the 5 less infeasible are selected for migration. In this case, 10 solutions will be deleted in the arriving island: 5 are the ones with the biggest cost objective function OFC and 5 are the ones with the biggest pressure-deficit objective function OFP. The classification 'worst' means the most expensive solutions and the more infeasible solutions, i.e. solutions that violate the pressure constraints more.

It could be useful to stress that, when all solutions are non-dominated, NSGA-II prefers those with larger crowding distance. This implies, for example, that solutions with extreme values of the objective functions, i.e. having the biggest c_d, will be considered better than all the others even if these solutions are the most expensive and the most infeasible. So the preference criterion of the algorithm can be completely opposite with respect to the one used for migration.

To evaluate the efficacy of island parallelization, two other modalities have been considered: *No Exchange* version, where solutions are never exchanged between islands, and the '*equivalent serial*' version. The first one is useful to establish whether frequent migrations improve results, while the second one can be used to see whether communicating islands perform better than a unique process with a larger population and the same number of EPANET calls. Having a larger population means more space for non-dominated solutions and therefore a lower probability to delete some good solutions because of its small crowding distance. The idea is to see if the island model can equal or improve the results with respect to the *equivalent serial* version. With the aim of having the same random number generator of the island model, this *equivalent serial* version is realized as an island version that does not exchange solutions and where each island has a population whose size is the same of the global one used in V1, V2 and V3.

Hanoi network

This network (Fujiwara & Khang 1990) has 34 pipes to design using six possible diameters. The OFP constraint (6) requires us to maintain 30 m for all 31 demand nodes served by a reservoir of 100 m head. The best solution obtained for this network is $\$6.081 \times 10^6$ (Reca & Martinez 2006).

Each island has a population of 100 solutions (i.e. Pop = 400) and the evolution time is fixed to NGen = 1,000. Tables 1 and 2 report results for V1, V2 and V3 modes in terms of mean cost (Table 1) and best cost (Table 2) obtained using 10 different seeds of the random number generator.

In order to compare the results of Tables 1 and 2 with those of the *equivalent serial* NSGA-II with the same EPANET calls, we used four islands with population Pop = 1,600 solutions (400 for each islands), NGen = 1,000 generations and without migration. The mean and the best cost of the cheapest feasible solutions of this '*equivalent serial*' version are respectively $\$6.310 \times 10^6$ and $\$6.170 \times 10^6$. The fact that neither the parallel versions nor the serial one reach the best solution of $\$6.081 \times 10^6$ is probably due to the combination of parameters (p_c, p_m, Pop, NGen). They usually have to be expressly tuned for the specific problem, but this is beyond the aim of this paper. However, other configurations find results with costs very close to the best solution (see the discussion subsection) (Figure 6).

Figure 7 summarizes the best results obtained, with the aim to simplify the comparison. We see that the worst

Table 1 | Mean of the cheapest feasible solutions found (10^6) in 10 runs for Hanoi network

P_{ex}	f_{ex} 10	50	100	200	500
V1					
2	6.436	6.423	6.463	6.433	6.493
10	6.440	6.428	6.406	6.416	6.487
20	6.409	6.398	6.383	6.438	6.444
50	6.398	6.460	6.397	6.403	6.433
V2					
2	6.375	6.346	6.381	6.402	6.456
10	6.341	6.335	6.379	6.407	6.486
20	**6.282**	**6.230**	6.325	6.392	6.469
50	**6.266**	**6.232**	**6.247**	6.348	6.469
V3					
2	6.413	6.373	6.386	6.402	6.456
10	6.390	6.367	6.412	6.407	6.456
20	**6.266**	**6.240**	6.328	6.392	6.469
50	**6.262**	**6.248**	**6.260**	6.351	6.469
No exchange 6.469					

Configurations with better results compared with the equivalent serial version ($6.310 × 10^6) are highlighted in bold.

Table 2 | Best feasible solutions found (10^6) in 10 runs for Hanoi problem

P_{ex}	f_{ex} 10	50	100	200	500
V1					
2	6.313	6.271	6.334	6.281	6.283
10	6.193	6.233	6.275	6.277	6.290
20	6.220	6.256	6.278	6.263	6.252
50	6.295	6.218	6.281	6.250	6.317
V2					
2	6.207	**6.140**	6.201	6.205	6.307
10	**6.129**	**6.148**	6.182	6.226	6.361
20	**6.096**	**6.122**	**6.138**	6.215	6.300
50	**6.101**	**6.097**	**6.106**	6.211	6.308
V3					
2	6.269	6.204	6.201	6.205	6.307
10	6.199	6.191	6.251	6.226	6.355
20	**6.101**	**6.139**	**6.131**	6.215	6.300
50	**6.105**	**6.105**	**6.131**	6.208	6.308
No exchange 6.310					

Configurations with better results compared with the equivalent serial version ($6.170 × 10^6) are highlighted in bold.

results are obtained for configurations that rarely exchange information (i.e. f_{ex} equal to 500) or that exchange very few solutions (i.e. P_{ex} equal to 2). This is also confirmed by other researchers (Khabzaoui *et al*. 2005).

Mode V1, in which migrating solutions are randomly chosen, has the worst results, as observed by Defersha & Chen (2008). Even if communication improves the final cost with respect to the *No Exchange* case, it is not clear which configurations are better. Moreover, having a good *Mean Cost* value does not mean that *Best Cost* is also good.

Modes V2 and V3 show a better behaviour and find better results than the equivalent serial optimization. However, in this case it is still not clear which is the best parameter configuration. Nevertheless we can state that it must be located in a region corresponding to frequent migrations and a large number of solutions passed.

Modes V2 and V3 *try* to concentrate the population (and the search) in regions of interest for the designer: cheapest feasible solutions and nearly feasible solutions. In this way it could be possible to explore the more interesting part of solution space without constraint, much like the research of a standard NSGA-II.

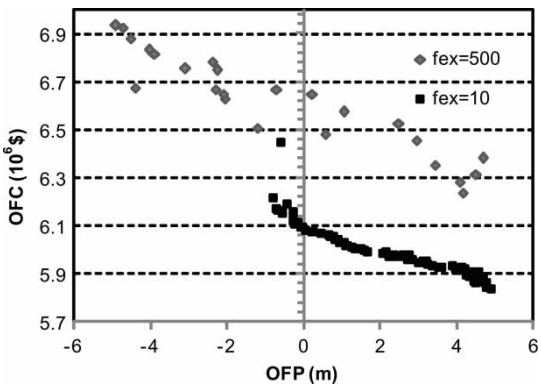

Figure 6 | Results for mode V3 with $P_{ex} = 20\%$, equal seed, and $f_{ex} = 10$ and $f_{ex} = 500$, respectively.

It is also interesting to note that an island parallel NSGA-II can perform better than a serial NSGA-II with a large population: besides a cost improvement, this also means shorter times. In fact, non-dominated sorting solutions are a time-consuming phase in which solutions are

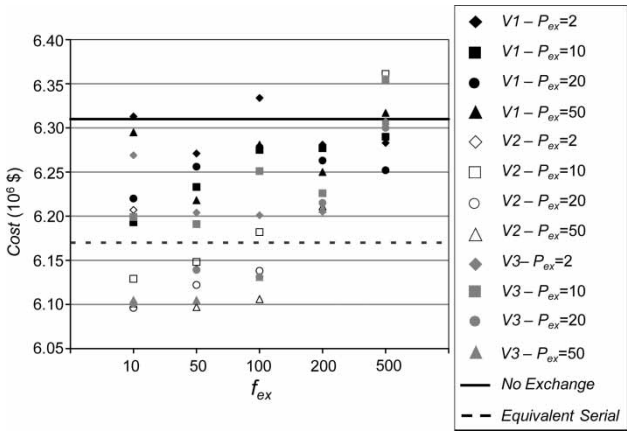

Figure 7 | Best results for Hanoi network obtained from modes V1, V2 and V3 with different configurations: $P_{ex} = 2$, 10, 20 and 50% and $f_{ex} = 10$, 50, 100, 200 and 500, respectively, compared with the results obtained from the *no exchange* version and from the *equivalent serial* version.

repeatedly compared for all the objectives and it is not parallelized. Island parallel NSGA-II versions with suitable parameters can have the double advantage of finding better results with shorter times. Obviously there is an inferior limit to the island's population size, but it is related to the fact that each island runs an independent NSGA-II for most of the time and tries to build its own Pareto front.

Another interesting point is related to the effect of massive communication. Figure 6 shows the final front in proximity to the feasibility limit. We can see that, while a small number of communications produces sparse solutions, all the solutions are concentrated in a narrow region when migration is frequent. This could seem a disadvantage because the aim of NSGA-II is to discover the front in a wide and homogeneous way, but, in WDN optimal design, being the objective OFP a constraint translation, only a part of the final front is interesting to the designer. Moreover, forcing the exploration of this part of the search space by communication brings us to the cheapest solution more quickly and that is what the user looks for. Obviously, if the problem specifies the OFP range in which solutions are desired, more efforts could be made to obtain a homogenous cover of this range.

Modena network

The Modena network consists in 317 pipes to design using 13 possible diameters. Required pressure is 20 m for all 268 demand nodes, served by four reservoirs. The currently best solution that satisfies pressure constraints is $\$2.56 \times 10^6$ (Bragalli *et al.* 2008).

Each island works with 400 solutions (i.e. Pop = 1,600) and the fixed number of allowed iterations is 8,000. The more promising configurations were tested and Tables 3 and 4 report the mean and best solutions of five runs with different seeds for the random number generator.

To evaluate these results, an *equivalent serial* NSGA-II has been tested with population equal to 1,600 solutions for 8,000 iterations (all other genetic parameters are the same as in previous tests). In this case the mean and best cost of the cheapest feasible solutions are $\$2.875 \times 10^6$ and $\$2.865 \times 10^6$. For the configurations that we have considered, the communication between islands improves results with respect to the *No Exchange* case in which every island is completely independent. Moreover, the

Table 3 | Mean of the cheapest feasible solutions found in 5 runs for Modena problem ($\$10^6$)

	f_{ex}	
P_{ex}	10	50
V2		
20	**2.659**	**2.751**
50	**2.629**	**2.639**
V3		
20	**2.679**	**2.790**
50	**2.628**	**2.699**
No exchange 3.027		

Configurations with better results compared with the equivalent serial version ($\$2.875 \times 10^6$) are highlighted in bold.

Table 4 | Best solutions found in 5 runs for Modena network ($\$10^6$)

	f_{ex}	
P_{ex}	10	50
V2		
20	**2.621**	**2.728**
50	**2.607**	**2.629**
V3		
20	**2.654**	**2.767**
50	**2.601**	**2.689**
No Exchange 2.994		

Configurations with better results compared with the equivalent serial version ($\$2.865 \times 10^6$) are highlighted in bold.

equivalent serial version is also outperformed, reaching a lower mean and a better cost. Figure 8 highlights the cost difference of the best solutions.

For this network it is clear the difference between modes V2 and V3: not checking the existence of solutions, V2 in general is faster at the beginning of the search, because in the reproduction phase it is easy that feasible or nearly feasible solutions are chosen, leading presumably to new working solutions. As the optimization proceeds, more 'best' solutions are passed, but if in f_{ex} iterations there are no improvements, this means that more copies of already existing solutions are inserted in the population. At the end of the evolution, i.e. after 8,000 generations, only a few dozen of solutions out of 400 are different, thus limiting the exploration capacity of NSGA-II.

The mode V3 has the advantage that solutions are always different in an island population. Therefore this version has more chances to improve results when a large number of generations is allowed. Repeating five runs with a double number of iterations (Niter = 16,000, Pop = 1,600, $f_{ex} = 10$, $P_{ex} = 50$), mode V3 finds the optimal solution of $\$2.56 \times 10^6$ whereas mode V2 reaches only $\$2.60 \times 10^6$. Moreover, for all these tests, mode V3 always outperforms V2 for each single seeds (Table 5).

DISCUSSION

This paper is focused on the contribution of parallel processing applied to a multi-objective algorithm from the perspective of using fast and effective optimization techniques for real size networks. Our results compare the performance of different NSGA-II parallel implementations with the basic version of the algorithm, taking into account the new parameters introduced by the parallelization (f_{ex}, P_{ex}). The experiments are run varying (f_{ex}, P_{ex}) and considering the same set of NSGA-II parameters for both networks, as done in other cases in the literature (Jourdan *et al.* 2005; Keedwell & Khu 2005; Reca & Martínez 2006; di Pierro *et al.* 2009).

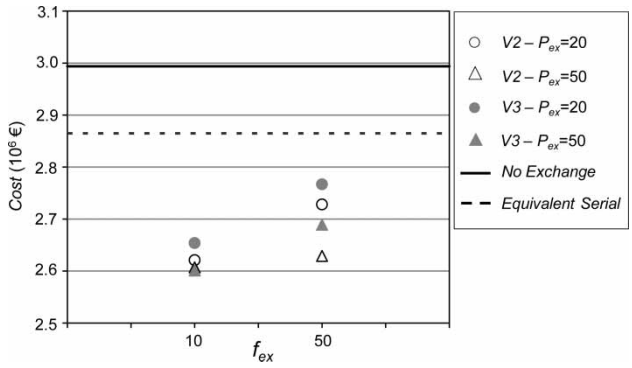

Figure 8 | Results for Modena network obtained from modes V2 and V3 with different configurations: $P_{ex} = 20$ and 50% and $f_{ex} = 10$ and 50, respectively, compared with the results obtained from the *no exchange* version and from the *equivalent serial* version.

Table 5 | Cost of the cheapest feasible solution found with modes V2 and V3 with Pop = 1,600 (400 for each island) and NGen = 16,000

	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
V2	2.598	2.6379	2.634	2.635	2.609
V3	2.560	2.591	2.578	2.583	2.597

To improve the algorithm effectiveness, it could be interesting to investigate the interaction between (f_{ex}, P_{ex}) and the NSGA-II parameters, also in relationship to the size of the network. So far, we tried to run tests of island version V2 for the Hanoi network considering a few other values for the mutation probability: $p_m = 0.0098 = 1/(34 \times 3)$ and $p_m = 0.033$, for a configuration with $f_{ex} = 10$ and $P_{ex} = 20$. The best results obtained are solutions with cost $\$6.110 \times 10^6$ and $\$6.093 \times 10^6$, respectively. They do not change in a significant way with respect to the result of $\$6.096 \times 10^6$ obtained with $p_m = 0.0033$. These preliminary results suggest that a particular fine-tuning of the whole set of parameters of NSGA-II parallel implementations could be necessary. This study will be the topic of future investigations.

At the time results show that the different algorithm exploitation method introduced by parallelization has a positive impact on the NSGA-II algorithm. In fact, the optimal solution obtained in this work is not so far from the best solution of $\$6.081 \times 10^6$ known in the literature for the Hanoi network (without considering split pipe design) and obtained by means of different algorithms, in particular genetic algorithm (Reca & Martínez 2006), tabu search (Sung *et al.* 2007) and particle-swarm harmony search (Geem 2009).

CONCLUSIONS

This work showed a different way of improving speed-up and results of an NSGA-II algorithm applied to optimal design of WDN. Both global and island parallelization produce a good speed-up of the algorithm up to a limited number of processors: good scalability in the global model is reduced by a bottleneck problem of communication in the master–slave structure, while in the island model the population size of each island must be sufficiently large to produce significant solutions. Therefore the initial population cannot be partitioned among too many processors. However, the island model has a linear scalability: doubling the population size requires twice the number of islands to reach good solutions in the same time. In fact, this model is almost embarrassingly parallel because it consists in different NSGA-IIs that evolve independently for most of the evolution time, except for a communication phase where solutions are moved from one population to another.

In island parallelization further parameters are necessary: frequency and number of migrating solutions and the criterion for selecting the migrants. We observed that migration improves the final cost of feasible solutions compared with the configuration where there is no communication.

The different criteria for selecting solutions showed that frequent and massive exchange of interesting (from an engineering point of view) solutions bring about better results with respect to the *equivalent serial* version. Moreover, allowing migration of identical solutions has the advantage of fast convergence towards good solutions, but the exploration capacity of the algorithm reduces as the number of allowed generations increases. The selection criterion of choosing more interesting solutions gives interesting results but obviously fails to take in account the extreme part of the front. As this extreme can be useful for maintaining a large degree of solution diversity, and therefore a good potential for exploration, future works are needed to balance these two aspects.

Island parallelization is still a vast territory to explore. Future research can concern further tests on parameter configurations or the possibility of varying their values during the algorithm generations. Selection criteria or even the link of other algorithms as it happens for serial NSGA-II can be considered. Finally it could be interesting testing different approaches in the definition of objectives, in particular the possibility of considering different objective functions for each island.

ACKNOWLEDGEMENTS

We would like to thank the anonymous referees for their contribution to the improvement of this paper.

REFERENCES

Alperovits, E. & Shamir, U. 1977 Design of optimal water distribution systems. *Wat. Res. Res.* **13** (6), 885–900.

Atiquzzaman, M., Liong, S. Y. & Yu, X. 2006 Alternative decision making in water distribution network with NSGA-II. *J. Wat. Res. Plann. Mngmnt.* **132** (2), 122–126.

Baños, R., Gil, C., Reca, J. & Montoya, F. G. 2010 A memetic algorithm applied to the design of water distribution networks. *Appl. Soft Comput.* **10**, 261–266.

Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A. & Toth, P. 2008 *Water Network Design by MINLP*. IBM Research Report, RC24495 (W0802-056).

Cantù-Paz, E. 1997 *Designing Efficient Master–Slave Parallel Genetic Algorithms*. IlliGAL report no. 97004, Department of Computer Science, University of Illinois at Urbana-Champaign.

Cantù-Paz, E. 2007 Parameter setting in parallel Genetic Algorithms. *Stud. Comput. Intell. (SCI)* **54**, 259–276.

Cantù-Paz, E. & Goldberg, D. E. 1999 On the scalability of parallel genetic algorithms. *Evolut. Comput.* **7** (4), 429–449.

Cunha, M. & Sousa, J. 1999 Water distribution network design optimization: simulated annealing approach. *J. Wat. Res. Plann. Mngmnt.* **125** (4), 215–221.

Deb, K., Agrawal, S., Pratap, A. & Meyarivan, T. 2000 *A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II*. Technical Rep. 200001, Indian Institute of Technology, Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur, India.

Defersha, F. M. & Chen, M. 2008 A parallel genetic algorithm for dynamic cell formation in cellular manufacturing systems. *Int. J. Product. Res.* **46** (22), 6389–6413.

Dridi, L., Parizeau, M., Mailhot, A. & Villenueve, J. P. 2008 Using evolutionary optimization techniques for scheduling water pipe renewal considering a short planning horizon. *Comput. Aided Civil Infrastruct. Engng.* **23**, 625–635.

Durillo, J. J., Nebro, A. J., Luna, F. & Alba, E. 2008 A study of master-slave approaches to parallelize NSGA-II. 978-1-4244-1694-3/08IEEE.

Eusuff, M. M. & Lansey, K. E. 2003 Optimization of water distribution network design using the shuffled frog leaping algorithm. *J. Wat. Res. Plann. Mngmnt.* **129** (3), 210–225.

Fujiwara, O. & Khang, D. B. 1990 A two phase decomposition method for optimal design of looped water distribution networks. *Wat. Res. Res.* **26** (4), 539–549.

Geem, Z. W. 2006 Optimal cost design of water distribution networks using harmony search. *Engng. Optimiz.* **38** (3), 259–277.

Geem, Z. W. 2009 Particle-swarm harmony search for water networks design. *Engng. Optimiz.* **41** (4), 297–311.

Giustolisi, O. & Berardi, L. 2009 Prioritizing pipe replacement: from multiobjective genetic algorithms to operational decision support. *J. Wat. Res. Plann. Mngmnt.* **135** (6), 484–492.

Goldberg, D. E. & Kuo, C. H. 1987 Genetic algorithms in pipeline optimization. *J. Comput. Civil Engng.* **1** (2), 128–141.

Jourdan, L., Corne, D., Savic, D. & Walters, G. 2005 Preliminary investigation of the 'learnable evolution model' for faster/better multiobjective water systems design. (C. A. Coello Coello, *et al*. eds). EMO 2005, LNCS 3410, Springer-Verlag, Berlin, Heidelberg, pp. 841–855.

Kadu, M. S., Gupta, R. & Bhave, P. R. 2008 Optimal design of water networks using a modified genetic algorithm with reduction in search space. *J. Wat. Res. Plann. Mngmnt.* **134** (2), 147–160.

Keedwell, E. & Khu, S. T. 2005 A hybrid genetic algorithm for the design of water distribution networks. *Engng. Appl. Artif. Intell.* **18**, 461–472.

Kessler, A. & Shamir, U. 1989 Analysis of the linear programming gradient method for optimal design of water supply networks. *Wat. Res. Res.* **25** (7), 1469–1480.

Khabzaoui, M., Dhaenens, C. & Talbi, E. 2005 Parallel genetic algorithms for multi-objective rule mining. In *MIC2005: the 6th Metaheuristics International Conference*, August 22–26, Vienna, Austria.

Liong, S. & Atiquzzaman, M. 2004 Optimal design of water distribution network using shuffled complex evolution. *J. Inst. Engrs., Singapore* **44** (1), 93–107.

Lopez-Ibanez, M., Prasad, D. T. & Paechter, B. 2008 Parallel optimisation of pump schedules with a thread-safe variant of Epanet toolkit. In *Proc. 10th Annual Water Distribution Systems Analysis Conference WDSA2008*, August 17–20, Kruger National Park, South Africa.

Maier, H. R., Simpson, A. R., Zecchin, A. C., Foong, W. K., Phang, K. Y., Seah, H. Y. & Tan, C. L. 2003 Ant colony optimization for design of water distribution systems. *J. Wat. Res. Plann. Mngmnt.* **129** (3), 200–209.

Mascagni, M. & Srinivasan, A. 2000 Algorithm 806: SPRNG: a scalable library for pseudorandom number generation, *ACM Trans. Math. Software (TOMS)* **26** (3), 436–461.

Montalvo, I., Izquierdo, J., Schwarze, S. & Pérez-García, R. S. 2010 Multi-objective particle swarm optimization applied to water distribution systems design: an approach with human interaction. *Math. Comput. Modell.* **52** (7–8), 1219–1227.

Nowostawski, M. & Poli, R. 1999 Dynamic Demes Parallel Genetic Algorithm, *Proceedings of International Conference, IEEE*, 93–98.

Perelman, L., Krapivka, A. & Ostfeld, A. 2009 Single and multi-objective optimal design of water distribution systems: application to the case study of the Hanoi system. *Wat. Sci. Technol: Wat. Supply* **9** (4), 395–404.

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. *Environ. Modell. Software* **24**, 202–213.

Prasad, T. D. & Park, N. 2004 Multiobjective genetic algorithms for design of water distribution networks. *J. Wat. Res. Plann. Mngmnt.* **130** (1), 73–82.

Prasad, T. D. & Tanyimboh, T. T. 2008 Entropy based design of 'Anytown' water distribution network. In *Proc 10th Annual Water Distribution Systems Analysis Conference WDSA2008*, August 17–20, Kruger National Park, South Africa.

Reca, J. & Martinez, J. 2006 Genetic algorithms for the design of looped irrigation water distribution networks. *Wat. Res. Res.* **42**, W05416.

Reca, J., Martínez, J., Gil, C. & Baños, R. 2008 Application of several meta-heuristic techniques to the optimization of real looped water distribution networks. *Wat. Res. Mngmnt.* **22**, 1367–1379.

Risco-Martín, J. L., Atienza, D., Hidalgo, J. I. & Lanchares, J. 2008 A parallel evolutionary algorithm to optimize dynamic data types in embedded systems. *Software Comput.* **12**, 1157–1167.

Rossman, L. A. 2000 *EPANET2 User's Manual*. US Environmental Protection Agency, Cincinnati, OH.

Savic, D. A. & Walters, G. A. 1997 Genetic algorithms for least cost design of water distribution networks. *J. Wat. Res. Plann. Mngmnt.* **123** (2), 67–77.

Schaake Jr., J. C. & Lai, D. 1969 *Linear Programming and Dynamic Programming Application to Water Distribution Network Design*. Report 116, Hydrodynamics Laboratory, Department of Civil Engineering, School of Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Sherali, H. D., Subramanian, S. & Loganathan, G. V. 2001 Effective relaxation and partitioning schemes for solving water distribution network design problems to global optimality. *J. Global Optimiz.* **19**, 1–26.

Sinha, E. & Minsker, B. S. 2007 Multiscale island injection genetic algorithms for groundwater remediation. *Adv. Wat. Res.* **30**, 1933–1942.

Skolicki, Z. 2005 An analysis of island model in evolutionary computation. In *GECCO'05*, June 25–29, Washington, DC.

Sung, Y., Lin, M., Lin, Y. & Liu, Y. 2007 Tabu search solution of water distribution network optimization. *J. Environ. Engng. Mngmnt.* **17** (3), 177–187.

Todini, E. & Pilati, S. 1988 A gradient algorithm for the analysis of pipe networks. *Comput. Appl. Wat. Supply* **1**, 1–20.

Tospornsampan, J., Kita, I., Ishii, M. & Kitamura, Y. 2007 Split-pipe design of water distribution network using simulated annealing. *Int. J. Comput., Inf. Syst. Sci. Engng.* **1** (3), 153–163.

Tsai, F. T.-C., Katiyar, V., Toy, D. & Goff, R. A. 2009 Conjunctive management of large scale pressurized water distribution and groundwater systems in semi-arid area with parallel genetic algorithm. *Wat. Res. Mngmnt.* **23**, 1497–1517.

Vairavamoorthy, K. & Ali, M. 2000 Optimal design of water distribution systems using genetic algorithms. *Comput. Aided Civil Infrastruct. Engng.* **15**, 374–382.

Wu, Z. Y. & Simpson, A. R. 2001 Competent genetic-evolutionary optimization water distribution systems. *J. Comput. Civil Engng.* **15** (2), 89–101.

Yates, D. F., Templeman, A. B. & Boffey, T. B. 1984 The computational complexity of the problem of determining least capital cost designs for water supply networks. *Engng. Optimiz.* **7**, 143–155.

Zecchin, A. C., Maier, H. R. & Simpson, A. R. 2007a Case study based convergence behaviour. Analysis of ACO applied to optimal design of water distribution systems. In: *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization* (T. Felix, S. Chan & Manoj Kumar Tiwari, eds). Itech Education and Publishing, Vienna, p. 532.

Zecchin, A. C., Maier, H. R., Simpson, A. R., Leonard, M. & Nixon, J. B. 2007b Ant colony optimization applied to water distribution system design: comparative study of five algorithms. *J. Wat. Res. Plann. Mngmnt.* **133** (1), 87–92.

First received 14 January 2010; accepted in revised form 24 January 2011. Available online 30 June 2011