General methodology for using Epanet as an optimization element in evolutionary algorithms in a grid computing environment for water distribution network design

Erika Yesenia Avila-Melgar, Marco Antonio Cruz-Chávez and Beatriz Martinez-Bahena

ABSTRACT

In this paper, an evolutionary algorithm, called EA-WDND, is developed to optimize water distribution network design for real instances. The evolutionary algorithm uses the Epanet Solver which, while not an optimizer, helps to evaluate the operational constraints of mass conservation, energy conservation, pressure in nodes (nodal heads) of the network, and velocities of water in network pipes. Epanet is used by the EA-WDND to evaluate whether the looped network is operating properly. Consequently, the EA-WDND obtains feasible configurations of network design. The best configuration, which has the lowest cost and best performance according to defined constraints, is obtained by the EA-WDND. This configuration can be practically implemented in real life. In this paper, a methodology for using Epanet Solver with a parallel evolutionary algorithm is presented.

Key words | Epanet Solver, optimal design, parallel evolutionary algorithm, water distribution network design

Erika Yesenia Avila-Melgar
Marco Antonio Cruz-Chávez
(corresponding author)
Beatriz Martinez-Bahena
CIICAp,
Autonomous University of the State of Morelos,
Av. Universidad 1001. Col. Chamilpa, C.P. 62209,
Cuernavaca,
Morelos,
México
E-mail: *mcruz@uaem.mx*

INTRODUCTION

Currently, water distribution networks are indispensable in society. One of the key issues in order to have a properly working water distribution network is the network design. The network design is a very common problem which requires the correct selection of components. The correct network design helps guarantee proper network operation and, at the same time, the appropriate service for network users in order to satisfy their water needs.

Real-world instances of water distribution networks involve very large networks. Numerous attempts have been made to solve these networks and several different methods have been proposed by researchers, including heuristic methods. The water distribution network design (WDND) problem is a well-known combinatorial optimization problem and is classified by complexity theory as an NP-hard problem. It has been widely studied by the scientific community for decades due to its practical applicability. The problem has been stated using different mathematical formulations. In addition, various solution techniques have been proposed in order to solve it.

The WDND problem has been classified, according to its characteristics and its mathematical model, into two approaches: the classical approach and modern approach (Cruz-Chávez *et al.* 2014).

In this work, the mathematical formulation which represents the WDND problem is defined by two models: the linear programming model and the constraint satisfaction model.

In order to solve the WDND problem, many different techniques and solution methods have been proposed. Recently,

doi: 10.2166/ws.2016.101

evolutionary algorithms have been demonstrated to be a good alternative to working with NP problems. They have been used by many researchers (Savic & Walters 1997; Luque & Alba 2011; Bureerat & Sriworamas 2013; Chang *et al.* 2013; Johns *et al.* 2013; Saldarriaga & Salcedo 2015). In this work, to achieve the goal of solving the WDND problem, an evolutionary algorithm (EA-WDND) is designed and implemented. The EA-WDND can solve different sized instances.

The EA-WDND works with the Epanet Solver to solve both mathematical models presented here and find the best solution to the WDND problem for small, medium, and large real-world instances. These instances are: the two-looped network proposed by Alperovits & Shamir (1977), the Hanoi network proposed by Fujiwara & Khang (1990), and the Balerma network, a large-scale real-world network instance, proposed by Reca & Martínez (2006) and Reca *et al.* (2008). In this work, the linear programming model is solved by the evolutionary algorithm EA-WDND, while the constraint satisfaction model is solved by the Epanet Solver.

The contribution of this work is an evolutionary algorithm called EA-WDND which implements Epanet Solver. The EA-WDND can be used to design new networks or redesign existing networks which do not have the most desirable performance.

Solutions for different sized instances can be found in limited computational time because the EA-WDND works in a grid computing environment, using parallel computing techniques. The grid computing environment is a set of computer clusters which are located in different geographic places but linked using Internet 2 technology.

The sections of the present article are organized as follows. The following section explains the mathematical formulation for the WDND problem. The third section describes the methodology used to link Epanet to the EA-WDND, in order to work in a grid computing environment. The fourth section shows the experimental results. The final section asserts the conclusions of this work.

MATHEMATICAL FORMULATION FOR THE WDND PROBLEM

Basically, the WDND problem consists of finding the most efficient way to deliver water from sources to users, within given constraints.

Let $P = \{p_1, p_2, p_3, \ldots, p_n\}$ be a set of n pipes in a network, and let $D = \{d_1, d_2, d_3, \ldots, d_n\}$ be the set of valid diameters that can be assigned to each $p_i \in P$. For each pipe $p_i \in P$, only one diameter $d_i \in D$ may be assigned in order to have a valid network design. The formulation of WDND used in this work has design and operational constraints. These constraints have been independently classified into two separate models: the linear programming model and constraint satisfaction model. The linear programming model has a main equation: the objective function. The objective function of the linear programming model consists of finding a configuration of diameters for the network design. The most suitable configuration is the one that minimizes the total cost of the network design. The objective function is presented in Equation (1). It consists of minimizing the total cost T_C of the water distribution network configuration, where n is the number of pipes in the network. The total cost T_C is based on the sum of the costs of each network pipe, of length L_{ijd_k}. The cost C_{ijd_k} is taken from a commercial diameters list and depends directly on the diameter and the length of the pipe.

$$Min\ T_C = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{d_k=d_1}^{d_n} C_{ijd_k} L_{ijd_k} X_{ijd_k} \tag{1}$$

$$\sum_i Q_{\text{in}} - \sum_i Q_{\text{out}} = Q_e \ \forall i \in n \tag{2}$$

$$\sum_m h_f - \sum_m E_p = 0 \ \forall m \tag{3}$$

$$H_{\min} \leq H_i \leq H_{\max} \ \forall i \in n \tag{4}$$

$$V_{\min} \leq V_{L_{ij}} \leq V_{\max} \ \forall L_{ij} \tag{5}$$

A feasible solution for the WDND problem is obtained by assigning exactly one diameter $d_i \in D$ for each $p_i \in P$. The feasible solution could just present the same diameter $d_i \in D$ for all the pipes of the network, but hydraulic constraints, defined in the constraint satisfaction model, must also be considered.

The mathematical formulation for the WDND incorporates information from real-world water distribution networks. The linear programming model includes design constraints for the network which reflect the commercial diameters

available that can be assigned to the pipes. The constraint satisfaction model includes network operation restrictions which ensure proper network performance and fulfillment of the users' pressure requirements. Some of the most important restrictions in the constraint satisfaction model are described in constraints (2) through (5). Constraint (2) represents the physical law of conservation of mass in each of n nodes in the network, where Q_{in} refers to the pipe flow into the loop, Q_{out} refers to the pipe flow away from the loop, and Q_e refers to the external demand. Constraint (2) states that the flow entering a node must be equal to the flow leaving the same node. Constraint (3) refers to the law of conservation of energy in a mesh m. A mesh m is a loop in the network. Therefore, constraint (3) indicates that the sum of the frictional energy lost along pipe lengths belonging to the hydraulic mesh should be zero if there are no power pumps in m. Constraint (4) refers to the minimum and maximum pressure requirements to satisfy the users' water needs while guaranteeing appropriate network operation. The diameter configuration represents a feasible solution for the WDND to be implemented in real life. Pressure requirements are verified at each demand node i of the network to avoid deficient pressures. Finally, constraint (5) is related to the limitation of flow velocity V in pipes. The minimum velocity requirement is defined to avoid reducing pipe diameter because of sediments. Maximum velocity is defined to ensure that the network does not present excessive frictional energy losses.

METHODOLOGY TO LINK EPANET TO THE EVOLUTIONARY ALGORITHM IN LINUX ENVIRONMENTS

In order to solve the linear programming model, an evolutionary algorithm finds the minimum cost configuration for the WDND. However, this minimum cost configuration is not the best solution for the network design problem if the hydraulic constraints, represented in the constraint satisfaction model, are not considered. In order to solve the constraint satisfaction model, a program can be codified. Currently, there are some hydraulic solvers widely accepted and commonly used by the scientific community for network simulations. This is the case for the well-known network solver Epanet. It is a program codified in C language which has been developed by researchers at the Environmental Protection Agency (Epanet 2015). It is free software and can be redistributed and/or modified under the terms of General Public License. The Epanet Solver has been tested and used by many researchers in the Windows environment as dynamic link library (DLL). However, it has rarely been used in Linux environments. López-Ibáñez *et al.* (2011) use Epanet in Linux. After analyzing the Epanet version they use, they conclude that it is a modified version which includes routines for working with water distribution networks that use pumps as a technique for water distribution.

The evolutionary algorithm design called EA-WDND is a very practical tool when working in conjunction with the Epanet Solver. The EA-WDND implements Epanet Solver as a static library and works either in a cluster of computers or in a grid environment. The EA-WDND algorithm and Epanet Solver frequently interact to solve the WDND problem, as can be seen in Figure 1. EA-WDND is a novel and useful sequential evolutionary algorithm, which differs from other evolutionary algorithms in many design and implementation details. For example, EA-WDND can solve small, medium and large-scale real-world instances with satisfactory results in a reasonable period of time. In addition, existing networks with improper performance can be redesigned by the EA-WDND algorithm. The Epanet Solver helps EA-WDND evaluate any configuration to determine pressure and the hydraulic behavior of the network.

First, the EA-WDND generates individuals to create an initial population. This initial population is analyzed by Epanet to determine if each individual in the population is feasible in the constraint satisfaction model. In this case, feasible means an individual must present pressures and velocities with values that are within the established limits. Epanet determines that an individual is feasible if it has pressures in nodes and velocities in pipes that are equal to or greater than the minimum required values and pressures lower than the maximum permitted value. Once Epanet qualifies an individual as feasible, it is placed into the set of feasible population. All the individuals in the feasible population are evaluated by an objective function (Equation (1)) to assign each individual its fitness. The best individuals are those whose fitness is the lowest because the objective function of the problem attempts to find the minimum cost configuration for the WDND. The best individuals constitute the feasible population. All the individuals of the feasible population can be randomly selected to

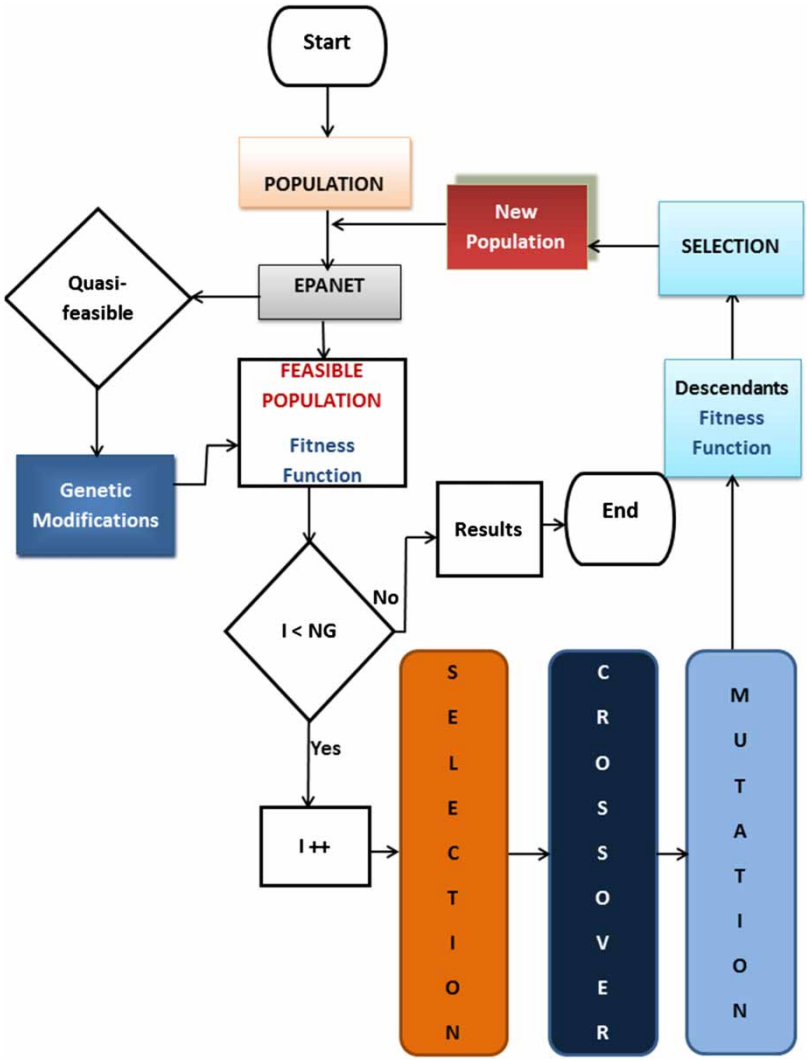

Figure 1 | General scheme of EA-WDND.

reproduce and generate offspring. Some individuals can be randomly affected by the mutation operator as happens in real life. Crossover and mutation operators generate individuals with specific characteristics. In this algorithm, those individuals are placed together in the set of descendants. The fittest descendants, according to their costs, are selected to be part of the new population. The new population is evaluated by Epanet to choose the best individuals according to their pressures. Another important characteristic of the EA-WDND algorithm is the adjustment of solutions. If an individual is quasi-feasible (a maximum of 10% of its genes make it an unfeasible solution), the individual can receive genetic alterations to be converted into a feasible individual. If the process of genetic modification is successful, the individual becomes feasible and is not eliminated as in the case of most evolutionary algorithms. This new feasible individual is placed in the set of feasible individuals. Later, a random decision determines whether the new feasible individual reproduces and continues living in following generations through its descendants.

In order to compare the results of the EA-WDND algorithm with other researchers, Epanet Version 2 was used by the evolutionary algorithm. In this network, the default parameter values used by Epanet 2.0 are the Hazen William equation with $\alpha = 4.277$, $a = 1.852$, $b = 4.5871$. The general methodology for Epanet to work as a part of the EA-WDND algorithm consisted of the following steps:

1. Access the Environmental Protection Agency page
2. Download Epanet for Linux
3. Create a static library in Linux
4. Include library toolkit.h in the evolutionary algorithm
5. Compile the evolutionary algorithm source code with Epanet
6. Execute the evolutionary algorithm.

The steps of the general methodology are described in detail as follows:

1. Access the Environmental Protection Agency Page (Epanet 2015). For this study, the page was last accessed in March 2015. There is Epanet for Linux of 32-bit platforms and Epanet for Linux of 64-bit platforms. Programmers must determine the appropriate version according to their needs.
2. Download Epanet for Linux. Once the programmer has chosen the best version of Epanet for Linux, Epanet must be downloaded and decompressed using the command ***tar -xzvf filename-Epanet.tar.gz***.
3. Create a static library in Linux. It is important to remember that libraries are files which contain functions that can be called on by applications and/or other libraries to perform tasks. Libraries can be used by many programs. In the Windows environment, the shared libraries take the form of dynamic link libraries or DLLs. In the UNIX environment, the fundamental attributes of dynamic link libraries or shared libraries are the same. In C language, the application code is linked with various libraries as static or shared libraries; the result is an executable file. A static library is an archive of object files which can be manipulated in different ways. A static library is created when the object files, created by a compiler from a set of source files, are bundled together into a single archive file. Each application that is linked with a static library essentially includes a copy of the library in the final executable version of the application. Using static libraries is fairly simple because references to function calls in the library code can be resolved in compilation time. Static libraries typically use '.a' extensions. In this work, Epanet is linked as a static library to the evolutionary algorithm developed here. The code within the Epanet library becomes an inseparable part of the executable file of the evolutionary algorithm application. The library function calls are resolved during link time. The evolutionary algorithm and Epanet as a static library form the complete application EAA-WDND (Evolutionary Algorithm Application for WDND) to solve the WDND problem.

In order to create the static library and have it working as part of the evolutionary algorithm, it is necessary to follow these steps:

(1) The source code written in C language (.c extension) must be compiled to obtain the object files (.o extension). For this work, the Epanet version downloaded including the object files into the folder src. This folder contained the Epanet source code and the corresponding object files for that source code. If object files were not available, they were generated by the programmer, or step 2 could not be carried out.
(2) The object files must be packed. The instructions that can be used in Linux are: ***ar –rcs./libstaticname.a filename.o*** and *'libstaticname'* is the name of the resulting static library when the object files are packed. In this work, the resulting static library is called *Epanetsl.a* and *'filename.o'* is the object file that is packed to obtain the static library. In order for Epanet to work properly, all the object files included in the directory 'src' of the downloaded Epanet version (Epanet.o, hash.o, hydraul.o, inpfile.o, input1.o, input2.o, input3.o, mempool.o, output.o, quality.o, report.o, rules.o, smatrix.o) must be packed. At this step, './' indicates the object files are located in the same directory in which the instruction is executed. If this is not the case, the path must be indicated explicitly.
(3) At this point, if steps 1 and 2 were carried out successfully, the static library will have been generated with the indicated name *Epanetsl.a*.
(4) Finally, the programmer must verify that the static library generated, *Epanetsl.a,* is placed in the same directory that contains the evolutionary algorithm to avoid configuring paths.

Including library toolkit.h in the evolutionary algorithm

The evolutionary algorithm, called EA-WDND, is a C language program. As with every program in C language, it includes some common libraries (stdio.h, stdlib.h) and it also includes the library 'toolkit.h.' The most important Epanet functions, included in toolkit.h and used by EA-WDND, can be seen in Pseudocode 1 and Pseudocode 2. Pseudocode 1 obtains information from the network and Pseudocode 2 carries out the network analysis.

Pseudocode 1. ReadNetwork

```
1   Input: IFilename
2   ErrorCode ← ENopen(IFilename, RFilename, OFilename)
3   if (ErrorCode >0) then
4     ENclose()
5     return 0
6   else
7     ENopenH()
8     ENgetcount(EN_LINKCOUNT,NumPipes)
9     ENgetcount(EN_NODECOUNT, NumNodes)
10  end if
11  Output: NumPipes, NumNodes
```

Pseudocode 1 shows how Epanet obtains information from the network. Line 1 shows that it is necessary to have an input data file (*IFilename*). This is a text file (typically with an .inp extension) which contains the input data from the network. It describes the pipe network being analyzed. The file is organized in sections, where each section begins with a keyword enclosed in brackets to reference a specific network (junctions, reservoirs, tanks, pipes, pumps, and so on). Table 1 shows the pipes section of the Two-LoopedNetwork.inp (instance proposed by Alperovits & Shamir 1977). As can be seen in Table 1, pipes are labeled by an identifier (ID). This ID shows that pipe 1 connects node 1 to node 2. It shows that the pipe length is 1,000 metres and the pipe diameter is 490 mm. These values are initially defined for every network. Diameters assigned to the pipes are generally different for each individual that represents a network configuration. Individuals are created and optimized by the evolutionary algorithm EA-WDND in order to find the individual with the least cost. EA-WDND sends individuals to Epanet for the evaluation of hydraulic properties. Therefore, EA-WDND interacts constantly with Epanet Solver and can achieve the goal of finding the best possible solution for the WDND problem.

Table 1 shows that the input file has an ID for the pipes. If the programmer wants to know which pipes are connected to a specific node, the information should be obtained based on the pipes' connections. For example, for node 1, pipe 1 is connected while for node 2, pipes 1, 2, and 3 are connected, as seen in Figure 2.

The initial diameters defined in the Two-LoopedNetwork.inp instance (Table 1) are set in the network by Epanet. In addition, Epanet assigns the diameters generated by the evolutionary algorithm. For each individual, which is generally a different design, Epanet sets the diameters for the network and obtains node pressures and pipe velocities when indicated explicitly by the EA-WDND. Lines 2–5 of Pseudocode 1 are used to ensure that Epanet can open the input file without problems. In line 2, three files are used (one input file and two output files). *IFilename* refers to the input file provided by the user of the application (EA-WDND). *RFilename* refers to a text report file created by Epanet, which contains information about the analysis of the network. *OFilename* contains results for all parameters for all the nodes and links of the network. The output files are important because they can be used for post-processing purposes. Line 7 is used to open the hydraulic analysis system. Line 8 is used to obtain the number of pipes of the network. Line 9 provides the number of nodes of the network. Finally, line 11 gives the number of pipes and the

Table 1 | Two-looped network.inp

Pipe ID	Node1	Node2	Length	Diameter
1	1	2	1,000	490
2	2	3	1,000	294
3	2	4	1,000	392
4	4	5	1,000	24
5	4	6	1,000	343
6	6	7	1,000	294
7	3	5	1,000	245
8	5	7	1,000	24

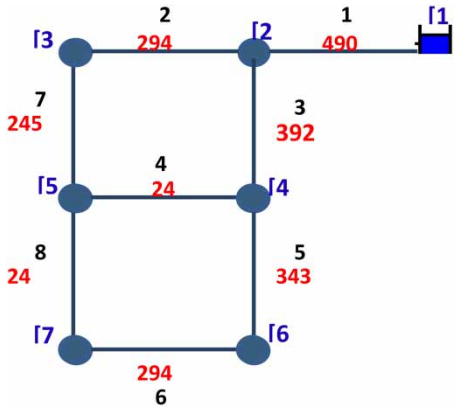

Figure 2 | Network configuration.

number of nodes of the network; these values are later used in the network analysis. Once Epanet acquires information from the network in Pseudocode 1, such as the number of pipes and nodes, the next step Epanet carries out is the network analysis shown in Pseudocode 2. Line 1 shows that the algorithm needs an input parameter network configuration. This configuration is provided by EA-WDND, and represented as an individual. The individual is feasible in the linear programming model because it considers all the defined constraints of the model. Some additional parameters required by Epanet as input parameters are *NumPipes* and *NumNodes*, which refer to the number of the pipes and the number of the nodes in the network, respectively. These parameters are obtained and provided by the algorithm *ReadNetwork*, presented in Pseudocode 1.

Pseudocode 2. AnalyzeNetwork

```
1   Input: Configuration,NumPipes, NumNodes
2   PNetwork ← 1
3   I, NumBelowMin, J,K← 0
4   while (I< NumPipes) do
5     ENsetlinkvalue(I + 1, EN_DIAMETER,Configuration[I])
6   end while
7   ENinitH(0)
8   ENrunH(t)
9   while (J< NumNodes) do
10    pressure,K ← 0
11    ENgetnodetype(J, NodeType)
12    if (NodeType = =0) then
13      ENgetnodevalue(J,EN_PRESSURE, pressure)
14      if (pressure < pressure MinReq) then
15        NumBelowMin ← NumBelowMin +1
16        PNetwork ← 0
17        pressures [K] ← pressure
18        K ← K + 1
19      end if
20    end if
21    J ← J + 1
22  end while
23  ENclose()
24  ENcloseH()
25  Output: PNetwork
```

Line 2 shows the variable *PNetwork,* which is used to represent the configuration's feasibility. The configuration for the WDND is feasible if the value of *PNetwork* is 1. If the value is not 1, it means the configuration is not feasible, frequently because some nodes of the network have pressures below the minimum required pressure. The number of nodes with pressure below the minimum required pressure is obtained by the counter *NumBelowMin* defined in line 3. Lines 4–6 are used to assign diameters in the network, creating a network design with each configuration, as shown in Figure 2. Lines 7 and 8 are used to indicate the initialization of hydraulic analysis, carried out by Epanet. Line 7 initializes storage tank levels, links status and settings, and simulates the clock time prior to running a hydraulic analysis. Line 8 runs a single-period hydraulic analysis, retrieving the current simulation clock time *t*.

Lines 9–22 are used to analyze all the nodes in the network to obtain the pressure at each node.

Line 11 retrieves the node-type code for a specific node. It can be a network user or a reservoir.

Line 12 obtains the existing pressure for the users of the network.

In general, the algorithm *AnalyzeNetwork* analyzes the network to find out important information about it. The most important information obtained is the current pressure at each node (network users). With this information, the EA-WDND computes the number of nodes in the network with pressures below the minimum required. The information related to the number of nodes in the network with pressures below the minimum required is used by EA-WDND to qualify an individual as quasi-feasible or unfeasible. Once the individual is qualified by EA-WDND as quasi-feasible, it can be adjusted via genetic alterations to make it feasible. It is important to mention that Pseudocode 1 and Pseudocode 2 could be unique functions. This would work perfectly in centralized environments. However, in distributed systems and grid environments it is necessary to split these into the two functions presented in Pseudocode 1 and Pseudocode 2. *ReadNetwork* and *AnalyzeNetwork* are independently implemented in order to guarantee that the processes running in the parallel algorithm can have correct access to network information without having to implement a synchronization method. The implementation of these two functions also helps the evolutionary algorithm in a grid

environment to minimize the required time for the algorithm's execution.

Compiling the evolutionary algorithm with Epanet

At this point, the evolutionary algorithm has to be compiled to generate the EAA-WDND, with the next instruction: ***gcc–o3 EAA-WDND EA-WDND.c –lm Epanetsl.a***. The command shows that the sequential evolutionary algorithm is compiled by using gcc. It is important to mention that the resulting application is ready to be executed in centralized environments. In this work, the parallel evolutionary algorithm is compiled by using an mpic or mpich compiler to run in grid environments. The EA-WDND is ready to be executed in grid environments when it has been previously compiled by using ***mpich –o3 EAA-WDND EA-WDND.c -lm Epanetsl.a***.

Executing the evolutionary algorithm

The last step of the methodology proposed here for using Epanet with an evolutionary algorithm is the execution of the algorithm in a grid environment.

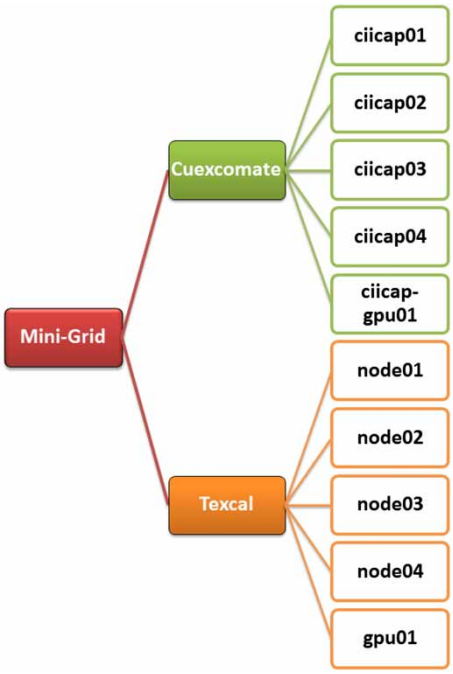

Figure 3 | Grid environment scheme.

The EA-WDND runs in a distributed environment by using a cluster of computers located in different geographic places. The general schema of the grid environment used in this work is presented in Figure 3. This grid environment includes homogeneous computational resources. For the grid environment used, clusters Cuexcomate and Texcal are the elements of the platform. Each cluster has its own nodes. The grid environment details are presented in (Cruz-Chávez *et al.* 2012).

This work focuses on the nodal heads for the networks obtained with the resultant EAA-WDND, working in a grid environment. The components of the application are illustrated in Figure 4. The EAA-WDND obtains the minimum-cost solution with the required minimum pressures for acceptable service for users. The Evolutionary Algorithm Application is the result of interaction between an evolutionary algorithm and an Epanet solver. In order for EA-WDND to interact with Epanet correctly, two elements must be included: (1) the toolkit.h and (2) the previously created static library Epanetsl.a. At the same time, a network must be sent to Epanet as the input parameter.

The necessary files to run the EAA-WDND are EA-WDND.c, Alperovits.inp, toolkit.h, and Epanetsl.a. These files are placed in each cluster of computers: Cuexcomate and Texcal. The Network File System service included in the clusters automatically replicates the necessary files for EAA-WDND execution in all the nodes of grid. After EA-WDND.c compilation in each cluster, the EAA-WDND is generated and automatically placed in the grid to be executed by all the nodes. At this point, the evolutionary algorithm can be observed to be correctly linked to the Epanet Solver as a static library.

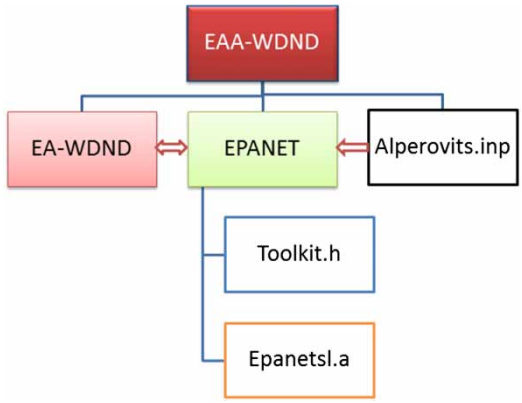

Figure 4 | EAA-WDND components.

EXPERIMENTAL RESULTS

The experiments with EAA-WDND consisted of obtaining pressures for network test problems Alperovits, Hanoi and Balerma (Reca & Martínez 2006; Reca *et al.* 2008; Baños *et al.* 2010; Artina *et al.* 2012), on the Mini-Grid platform. In all the experiments executed, the nodes Texcal and Cuexcomate were not used because they are the front end of each cluster. They have specific functions, such as communication. Assigning them more activities can affect algorithm efficiency. Each node of the cluster is called a processor to avoid confusion with the nodes of the water distribution network. Let ciicap01 be processor01, ciicap02 be processor02, and so on. In this experiment, every processor0*i* has access to the complete network layout and can obtain all the nodal heads for the network. The Alperovits network has seven nodes. One of the nodes is a reservoir to feed the network. The network has eight pipes arranged in two loops. The minimum required pressure is 30 metres above ground level for each node. The pipes are all 1,000 m long. Detailed data for the network is widely reported in many previous works (Reca *et al.* 2008; Baños *et al.* 2010; Cruz-Chávez *et al.* 2014).

Table 2 shows the configurations used by other researchers. A configuration is a set of diameters. A diameter is assigned to a specific pipe in the network; usually a different diameter is assigned for each pipe. A configuration is a solution for the WDND problem. The configurations in Table 2 are presented in inches (Abebe & Solomatine 1998). In Table 3, the same configurations are used in order to compare results. Table 3 shows the equivalent configurations of Table 2 in millimetres, which is the specific format Epanet needs for the input parameters. The EAA-WDND includes the Epanet Solver, so it also requires diameters expressed in millimetres as input parameters. Configuration number five, represented as 'Best Run' in Table 2, is also used and presented by other researchers (Abebe & Solomatine 1998; Tospornsampan *et al.* 2007). As a result, similar pressures are obtained.

In the experiments carried out in this work, each processor0*i* of the grid environment obtains only the pressure for one node of the water network. The first nodal head shown in Tables 4 and 5 corresponds to the reservoir, so there is no pressure, which is represented with values 0.00. Tables 4 and 5 present a comparison using the results obtained in different works. These results are compared to the ones obtained by the evolutionary algorithm in a grid environment working with Epanet as a static library, EAA-WDND. The results shown in Table 5 indicate that the nodal heads obtained by the

Table 2 | Alperovits network, pipe diameters in inches (Abebe & Solomatine 1998)

Pipe	CRS2	GA	ACCOL	CRS4	Best Run
1	18	18	22	18	18
2	10	14	18	16	10
3	16	14	20	14	16
4	4	1	3	2	4
5	16	14	16	14	16
6	10	1	4	1	10
7	10	14	18	14	10
8	2	12	16	10	1

Table 3 | Alperovits network, pipe diameters in millimetres, used for EAA-WDND

Pipe	CRS2	GA	ACCOL	CRS4	Best Run
1	457.2	457.2	558.8	457.2	457.2
2	254.0	355.6	457.2	406.4	254.0
3	406.4	355.6	508.8	355.6	406.4
4	101.6	25.4	76.2	50.8	101.6
5	406.4	355.6	406.4	355.6	406.4
6	254.0	25.4	101.6	25.4	254.0
7	254.0	355.6	457.2	355.6	254.0
8	50.8	304.8	406.4	254.0	25.4

Table 4 | Alperovits network, nodal heads (m) corresponding to the optimal diameters node algorithm (Abebe & Solomatine 1998)

Pipe	CRS2	GA	ACCOL	CRS4	Best Run
1	0.00	0.00	0.00	0.00	0.00
2	53.21	53.21	57.45	53.21	53.21
3	30.50	36.62	45.59	39.79	30.34
4	43.36	43.92	51.65	43.89	43.39
5	33.92	42.01	54.31	45.22	33.63
6	30.30	31.51	40.32	31.47	30.36
7	30.25	30.01	42.86	30.34	30.43

Table 5 | Alperovits network, relative error of nodal heads with EAA-WDND vs the literature (%RE)

Pipe	CRS2	GA	ACCOL	CRS4	Best Run
1	0.00	0.00	0.00	0.00	0.00
2	0.075	0.075	0.017	0.075	0.075
3	0.098	0.219	0.066	0.151	0.429
4	0.208	0.137	0.039	0.001	0.138
5	0.295	0.262	0.074	0.199	0.565
6	0.792	4.380	8.854	3.209	0.593
7	0.628	5.265	5.973	3.988	0.033

EAA-WDND are slightly superior in relative error, which means nodal heads can be obtained by different processors in a grid environment with satisfactory results. It can be observed that a relative error of less than 0.6% is obtained in all comparisons with the Best Run. At the same time, the results obtained show the appropriate performance of the application EAA-WDND, with an excellent coupling module EPANET.

The nodal heads results in the compared works was the same, because the configuration used as the input parameter is the same. The equations and the coefficients defined in the Epanet version impact the results.

The relative error is calculated by Equation (6), where *Nhl* is the nodal head of the algorithm consulted in Abebe & Solomatine (1998) (CRS2, GA, ACCOL, CRS4, Best Run) and *NhEAA* is the nodal head of the algorithm EAA-WDND:

$$\%RE = \frac{Nhl - NhEAA}{Nhl} 100 \tag{6}$$

The best solution known in the literature was found by EAA-WDND for the Alperovits instance. It was 419,000 for all 30 executions of the algorithm. Figure 5 shows the performance of the algorithm EAA-WDND and the results obtained.

For this small instance, Alperovits, the process of obtaining the nodal heads in a grid environment does not have great benefits in terms of the evaluation time. However, in real-life large-scale networks, the time required to evaluate configurations when many processors cooperate would be significantly reduced. In EAA-WDND execution, every processor has access to the complete network layout and can

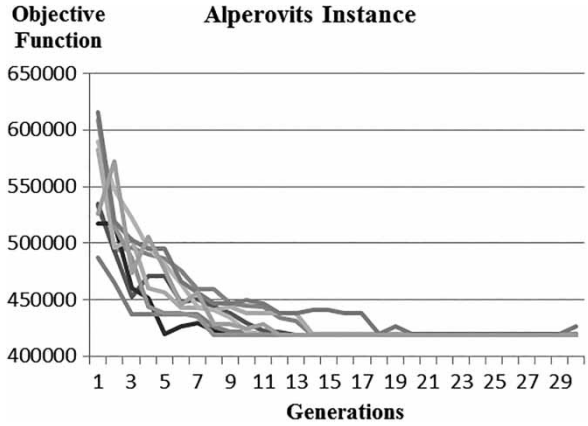

Figure 5 | Fitness values for the Alperovits instance.

obtain all the nodal heads from the network. These experiments have been conducted in this manner to show the feasibility of using grid environments to execute parallel applications. In this case, the evolutionary algorithm with Epanet is a static library, represented by the EAA-WDND application.

EAA-WDND performance in a grid environment was tested using the Hanoi network, another well-known theoretical instance commonly used by researchers. In this experiment, every processor has access to the complete network layout and can obtain all the nodal heads for the Hanoi network. However, in the experiments, each processor obtained approximately three or four nodal heads.

Tables 6 and 7 have the same structure. They show the nodal heads obtained from the Hanoi network with EAA-WDND execution using different configurations. They show the nodes of the Hanoi network and the names of the processes in a grid environment which obtain the nodal heads from the network. Finally, Tables 6 and 7 show the comparison of the nodal heads obtained in this work with results obtained by other researchers (Abebe & Solomatine 1998). The first nodal heads, shown in the tables in column EAA-WDND, correspond to the results of pressures obtained by the EAA-WDND application developed in this work. These results are compared to the ones obtained in previous work (Abebe & Solomatine 1998). Table 7 shows a comparison using another configuration. In this comparative table, satisfactory results are presented. They show the appropriate performance of the EAA-WDND in a grid environment. All the nodal heads for the

Table 6 | Hanoi network nodal heads comparison

Processor ID – Name		Pipes mm	EAA-WDND	(Abebe & Solomatine 1998)	
			Node	Nodal heads	
1	ciicap02	1,016	1	97.14	97.14
2	ciicap03	1,016	2	61.67	61.67
3	ciicap04	1,016	3	58.60	58.59
4	node01	1,016	4	54.84	54.82
5	node02	762	5	39.51	39.45
6	node03	1,016	6	38.71	38.65
7	node04	1,016	7	37.93	37.87
8	ciicap01	762	8	35.72	35.65
9	ciicap02	762	9	34.36	34.28
10	ciicap03	762	10	32.80	32.72
11	ciicap04	762	11	31.65	31.56
12	node01	762	12	30.23	30.13
13	node02	406.4	13	36.43	36.36
14	node03	609.6	14	37.23	37.17
15	node04	762	15	37.69	37.63
16	ciicap01	762	16	48.14	48.11
17	ciicap02	762	17	58.63	58.62
18	ciicap03	1,016	18	60.64	60.64
19	ciicap04	1,016	19	53.89	53.87
20	node01	1,016	20	44.62	44.48
21	node02	508.8	21	44.19	44.05
22	node03	508.8	22	39.90	39.83
23	node04	762	23	30.64	30.51
24	ciicap01	406.4	24	30.62	30.50
25	ciicap02	508.8	25	32.26	32.14
26	ciicap03	304.8	26	32.74	32.62
27	ciicap04	609.6	27	33.62	33.52
28	node01	508.8	28	31.57	31.46
29	node02	609.6	29	30.56	30.44
30	node03	762	30	30.51	30.39
31	node04	762	31	30.29	30.17
		762			
		762			
		304.8			

Table 7 | Hanoi network nodal heads comparison

Processor ID – Name		Pipes mm	EAA-WDND	(Abebe & Solomatine 1998)	
			Node	Nodal heads	
1	ciicap02	1,016	1	97.14	97.14
2	ciicap03	1,016	2	61.67	61.67
3	ciicap04	1,016	3	57.69	57.68
4	node01	1,016	4	52.78	52.75
5	node02	1,016	5	47.69	47.65
6	node03	762	6	43.02	42.97
7	node04	1,016	7	41.74	41.68
8	ciicap01	1,016	8	40.76	40.70
9	ciicap02	609.6	9	32.55	32.46
10	ciicap03	1,016	10	32.17	32.08
11	ciicap04	762	11	31.01	30.92
12	node01	1,016	12	30.66	30.56
13	node02	406.4	13	30.66	30.55
14	node03	406.4	14	30.80	30.69
15	node04	762	15	30.86	30.74
16	ciicap01	304.8	16	46.27	46.16
17	ciicap02	508.8	17	54.44	54.41
18	ciicap03	609.6	18	60.59	60.58
19	ciicap04	762	19	49.25	49.23
20	node01	1,016	20	47.95	47.92
21	node02	762	21	47.90	47.86
22	node03	762	22	42.01	41.96
23	node04	1,016	23	40.23	40.18
24	ciicap01	1,016	24	39.00	38.95
25	ciicap02	1,016	25	36.08	36.01
26	ciicap03	609.6	26	36.00	35.93
27	ciicap04	762	27	36.54	36.47
28	node01	304.8	28	36.52	36.45
29	node02	406.4	29	36.61	36.54
30	node03	1,016	30	36.71	36.64
31	node04	406.4	31	36.83	36.76
		508.8			
		762			
		609.6			

configurations used for the Hanoi network obtained with the EAA-WDND present better pressures for the network, as can be seen in Tables 6 and 7.

Finally, the performance of EAA-WDND was tested with the Balerma Network. The best solution found in the literature for the Balerma instance is €2,302,423 (Reca &

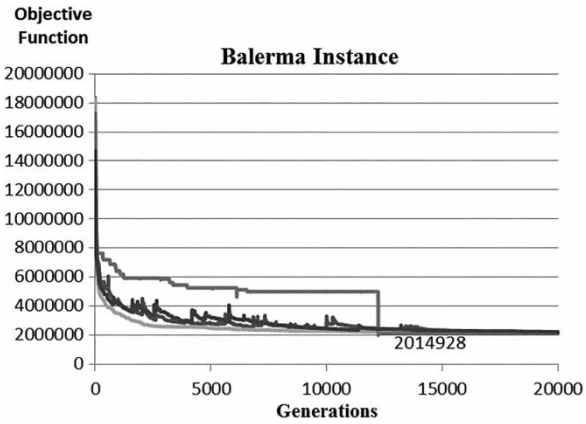

Figure 6 | Fitness values for the Balerma instance.

Martínez 2006). The best solution found by EAA-WDND was €2,014,928, an improvement of 12.5% over the current best-known solution. Figure 6 shows the convergence of the EAA-WDND in 30 tests for the Balerma instance.

CONCLUSIONS

The EAA-WDND presented in this work has excellent performance in a grid environment. A methodology for using Epanet with an evolutionary algorithm in a grid environment is presented. Experiments were carried out in this work and the results were compared to previous work. The results obtained in a grid environment are correct. Most work related to WDND uses algorithms to solve the problem in centralized environments or clusters of computers. Algorithms to solve this problem in grid environments have not been explored. This paper shows how the evolutionary algorithm interacts with Epanet as a static library, working in a grid environment. This work is important because it explores the possibility of applying parallel computing techniques to solve real problems, like the WDND problem, in a grid environment.

REFERENCES

Abebe, A. J. & Solomatine, D. P. 1998 Application of global optimization to the design of pipe networks. In: *Proc. 3rd International Conference on Hydroinformatics*, V. Babovic & L. C. Larsen (eds), Balkema, Rotterdam, The Netherlands, pp. 989–996.

Alperovits, E. & Shamir, U. 1977 Design of optimal water distribution systems. *Water Resources Research* **13** (6), 885–900. http://doi.org/10.1029/WR013i006p00885.

Artina, S., Bragalli, C., Erbacci, G., Marchi, A. & Rivi, M. 2012 Contribution of parallel NSGA-II in optimal design of water distribution networks. *Journal of Hydroinformatics* **14** (2), 310–323. http://doi.org/10.2166/hydro.2011.014.

Baños, R., Gil, C., Reca, J. & Montoya, F. G. 2010 A memetic algorithm applied to the design of water distribution networks. *Applied Soft Computing* **10** (1), 261–266. http://doi.org/10.1016/j.asoc.2009.07.010.

Bureerat, S. & Sriworamas, K. 2013 Simultaneous topology and sizing optimization of a water distribution network using a hybrid multiobjective evolutionary algorithm. *Applied Soft Computing* **13** (8), 3693–3702. http://doi.org/10.1016/j.asoc.2013.04.005.

Chang, J., Bai, T., Huang, Q. & Yang, D. 2013 Optimization of water resources utilization by PSO-GA. *Water Resources Management* **27** (10), 3525–3540. http://doi.org/10.1007/s11269-013-0362-8.

Cruz-Chávez, M. A., Juárez-Pérez, F. & Moreno Bernal, P. 2012 Minigrid Morelos. *Hypatia, Revista de Divulgación CientíficoTecnológica del Consejo de Ciencia y Tecnología del Estado de Morelos* **40**, 32–33.

Cruz-Chávez, M. A., Ávila-Melgar, E. Y., Cruz-Rosales, M. H., Martínez-Bahena, B. & Flores-Sánchez, G. 2014 Search space analysis for the combined mathematical model (linear and nonlinear) of the water distribution network design problem. In: *Artificial Intelligence and Soft Computing* , Lecture Notes in Computer Science Vol. 8467, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh & J. M. Zurada (eds), Springer International Publishing, Switzerland, pp. 347–359.

Epanet (2015) United States Environmental Protection Agency. http://www2.epa.gov/water-research/Epanet (accessed 31 March 2015).

Fujiwara, O. & Khang, D. B. 1990 A two-phase decomposition method for optimal design of looped water distribution networks. *Water Resources Research* **26** (4), 539–549. http://doi.org/10.1029/WR026i004p00539.

Johns, M. B., Keedwell, E. & Savic, D. 2013 Pipe smoothing genetic algorithm for least cost water distribution network design. In: *Proceedings of the Fifteenth Annual Conference on Genetic and Evolutionary Computation – GECCO '13*, C. Blum (ed.), ACM Press, New York, New York, USA, pp. 1309–1316. http://doi.org/10.1145/2463372.2463533.

López-Ibáñez, M., Prasad, T. D. & Paechter, B. 2011 Representations and evolutionary operators for the scheduling of pump operations in water distribution networks. *Evolutionary Computation* **19** (3), 429–467. http://doi.org/10.1162/EVCO_a_00035.

Luque, G. & Alba, E. 2011 *Parallel Genetic Algorithms: Theory and Real World Applications*. Springer-Verlag, Berlin and Heidelberg.

Reca, J. & Martínez, J. 2006 Genetic algorithms for the design of looped irrigation water distribution networks. *Water Resources Research* **42** (5), 1–9. http://doi.org/10.1029/2005WR004383.

Reca, J., Martínez, J., Gil, C. & Baños, R. 2008 Application of several meta-heuristic techniques to the optimization of real looped water distribution networks. *Water Resources Management* **22** (10), 1367–1379. http://doi.org/10.1007/s11269-007-9230-8.

Saldarriaga, J. & Salcedo, C. A. 2015 Determination of optimal location and settings of pressure reducing valves in water distribution networks for minimizing water losses. *Procedia Engineering* **119**, 973–983. http://doi.org/10.1016/j.proeng.2015.08.986.

Savic, D. A. & Walters, G. A. 1997 Genetic algorithms for least-cost design of water. *Journal of Water Resources Planning and Management* **123** (2), 67–77.

Tospornsampan, J., Kita, I., Ishii, M. & Kitamura, Y. 2007 Split-pipe design of water distribution network using simulated annealing. *International Journal of Computer and Information Engineering* **1** (3), 154–164.

First received 23 February 2016; accepted in revised form 31 May 2016. Available online 28 June 2016