Pump Scheduling Optimization Using Asynchronous Parallel Evolutionary Algorithms

Christian von Lücken   Benjamín Barán   Aldo Sotelo

National Computing Center
National University of Asunción
San Lorenzo, Paraguay

Abstract

Optimizing the pump-scheduling is an interesting proposal to achieve cost reductions in water distribution pumping stations. As systems grow, pump-scheduling becomes a very difficult task. In order to attack harder pump-scheduling problems, this work proposes the use of parallel asynchronous evolutionary algorithms as a tool to aid in solving an optimal pump-scheduling problem. In particular, this work considers a pump-scheduling problem having four objectives to be minimized: electric energy cost, maintenance cost, maximum power peak, and level variation in a reservoir. Parallel and sequential versions of different evolutionary algorithms for multi-objective optimization were implemented and their results compared using a set of experimental metrics. Analysis of metric results shows that our parallel asynchronous implementation of evolutionary algorithms is effective in searching for solutions among a wide range of alternative optimal pump schedules to choose from.

Keywords: Evolutionary Computation, Parallel Evolutionary Algorithms, Multiobjective Optimization, Scheduling.

1 Introduction

In conventional water supply systems, pumping of treated water represents a major expenditure in the total energy budget [1, 2]. Because so much energy is required for pumping, a saving of one or two percent can add up to several thousand dollars over the course of a year. In many pump stations an investment in a few small pump modifications or operational changes may result in significant savings [2]. In the general case, because of economical reasons, it is very difficult to achieve reductions by means of modifications in the current facilities of pump stations. Therefore, pump-scheduling optimization has proven to be a practical and highly effective method to reduce pumping costs without making changes to the actual infrastructure of the whole system.

Typically, a pumping station consists of a set of pumps having different capacities. These pumps are used in combination to drive water to one or more reservoirs. While doing this, hydraulic and technical constraints must be fulfilled. Thus, at a particular point in time, some pumps would be working but others would not. In this context, scheduling the pump's operation means choosing the right combination of pumps that will be working at each time interval of a scheduling period.

Then, a pump schedule is the set of all pump combinations chosen for every time interval of the scheduling horizon. An optimal pump schedule can be defined as a pump schedule that optimizes particular objectives, while fulfilling system constraints. Depending on the number of variables and objectives considered, optimizing a pump-scheduling problem may be very difficult, especially for large systems. Being a practical and challenging problem, it is no surprise that several authors have been studying it introducing different approaches [1–9].

Ormsbee et al. [3] present a detailed review of linear, non-linear, integer, dynamic, mixed, and other kinds of programming used to optimize a single objective: the electric energy cost. Lansey et al. [4] introduce the number of pump switches as an alternate way to evaluate the pumps' maintenance cost, which became the second objective considered until that date. In order to minimize the operating costs associated with water supply pumping systems several researchers have developed optimal control formulations. Mays [1] lists and classifies various algorithms that have been developed to solve the associated control problem. In the past few years, Evolutionary Computation techniques were introduced in the study of the optimal pump-scheduling problem. In fact, Mackle et al. [5] present a single objective optimization (electric energy cost) using Genetic Algorithms (GAs). Also, Savic et al. [6] propose a hybridization of a GA with a local search method, to optimize two objectives: electric energy cost and pump maintenance cost. In addition, Schaetzen [7] presents a single objective optimization using GAs considering system constraints by establishing penalties.

Evolutionary algorithms have proven to be useful tools helping decision makers (DM) to solve a multi-objective pump scheduling problem [8]. Since there is always a need for improvement, in [10], the use of a parallel model for Multiobjective Evolutionary Algorithms (MOEA) is proposed to provide DM with better solutions. Given that parallel implementations in [10] use a centralized migration topology, there is a bottleneck when the number of processes scales up. Our work extends the comparison of parallel models for MOEAs using a different parallel asynchronous approach than the one presented in [10].

This paper has been organized as follows: Section 2 presents a description of the optimal pump-scheduling problem considered. Section 3 presents motivations to choose the MOEA approach and to use parallel concepts to improve the efficiency attained by sequential implementations for the considered problem. Also, in this section the parallel implementation model is presented. Section 4 presents empirical comparisons of six different algorithms. Finally, conclusions of this work are presented.

2 Multi-objective Optimal Pump Scheduling Problem

In modeling pump operation, a simple framework can be used because only the large mains between the pump station and tanks are important in calculations [2]. Therefore, this work considers a simplified hydraulic model based on a real pumping station in Paraguay similar to the one presented in [7]. This model is composed of:

- an inexhaustible water source: the potable water reservoir;
- an elevated reservoir, which supplies water on demand to a community;

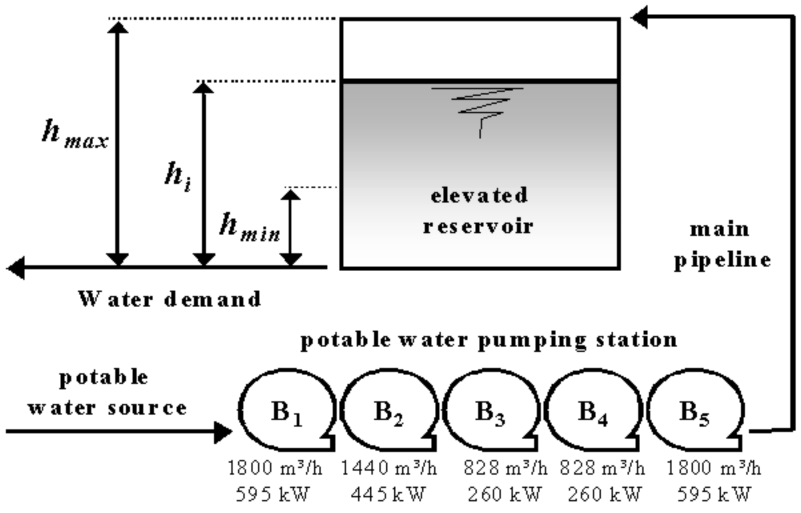

Figure 1: Model using five pumps.

- a potable water pumping station with n_p pumps used to pump water from a water source to the elevated reservoir; and
- a main pipeline used to drive water from a pumping station to an elevated reservoir.

The proposed model, using five pumps, is drawn in Figure 1. The pumping station is comprised of a set of n different constant velocity centrifugal pumps working in parallel association [2]. Pumping capacities are assumed constant during every time interval. Therefore, for a time interval of one hour, each pump combination has an assigned fixed discharge, electric energy consumption and power. The discharge rate of a pump combination may not be a linear combination of the pumping capacities of the individual pumps. Therefore, non-linearities in the combination of pumps are handled through a table of pump combination characteristics as presented in Table 1. The only data considered outside the model are the community's water demand.

The elevated reservoir stores water coming from the pumping station, it satisfies the community's water demand by gravity. An important aspect is the initial level, which has to be recovered by the end of the optimization period because:

- a final level above the initial one means extra water in the reservoir, and there is no need to store extra water in the reservoir if it is not going to be consumed by the community. This also implies a useless extra cost;
- a final level below the initial one means lack of water for the next day. This lack of water has to be recovered the next day, affecting its schedule through a variation of the initial parameters and an extra cost; and

Pump Combination	Code	Discharge [m^3/h]	Power [kW]
0	00000	0	0
1	00001	1800	595
2	00010	828	260
3	00011	2600	855
4	00100	828	260
5	00101	2600	855
6	00110	1650	520
7	00111	3450	1115
8	01000	1440	445
9	01001	3235	1040
10	01010	2260	705
11	01011	4060	1300
12	01100	2260	705
13	01101	4060	1300
14	01110	3090	965
15	01111	4890	1560
16	10000	1800	595
17	10001	3600	1190
18	10010	2620	855
19	10011	4420	1450
20	10100	2620	855
21	10101	4420	1450
22	10110	3450	1115
23	10111	5250	1710
24	11000	3235	1040
25	11001	5035	1635
26	11010	4060	1300
27	11011	5860	1895
28	11100	4060	1300
29	11101	5860	1895
30	11110	4890	1560
31	11111	6690	2155

Table 1: Technical characteristics of pump combinations.

- the goal is to keep a periodical schedule if conditions in consecutive days do not change substantially.

Therefore, a mass balance mathematical model was chosen. According to this, the amount of water that goes into the reservoir must be equal to the amount of water that comes out of it [2]. As an additional advantage, this model allows the same schedule to be used several times if water demand does not change substantially.

As water demand is an input data in this problem, it has to be obtained from reliable sources. The quality and applicability of an algorithm's solution depend on how good the predictions of the water demand are. Data are obtained through a statistical study of the community demand during many years. Through these studies, an estimated water demand can be established, according to certain parameters. Several models to predict this demand are presented in [2, 11].

In order to code the pumping schedule, a binary alphabet is used. At every time interval, a bit represents each pump. A 0 represents a pump that is not working, while a 1 represents a pump that is working. An optimization period of one day divided into twenty-four intervals of one hour each is considered. Thus, pumps can be turned on or off only at the beginning of each time interval in this model.

2.1 Mathematical definition of the problem: Objectives

In order to define the multi-objective optimal pump-scheduling problem to be solved in this work, the next subsections introduce the four different objectives considered in the optimization.

2.1.1 Electric energy cost (f_1)

Electric energy cost is the cost of all electric energy consumed by all pumps of the pumping station during the optimization period. An important issue to be considered when analyzing electric energy cost is the charge structure used by the electric company. In most electricity supply systems, electric energy cost is not the same throughout the whole day. This work considers the following charge structure:

- Low cost (C_l): from 0:00 to 17:00 hours and from 22:00 to 24:00 hours.
- High cost (C_h): from 17:00 to 22:00 hours.

The influence of this variable in the pump scheduling is remarkable. Electric energy costs can be substantially reduced if the optimal pump schedule uses the smallest possible number of pumps working during the high cost period [5,6]. Water already stored in the reservoir can be used during this period in order to satisfy the community's water demand. A different charge structure can also be considered if needed. The mathematical expression to calculate the electric energy cost E_c is given by Equation (1) [5]:

$$E_c = C_l \sum_{i=1}^{17} c(p_i) + C_h \sum_{i=18}^{22} c(p_i) + C_l \sum_{i=23}^{24} c(p_i) \tag{1}$$

where

i :	time interval index
p_i :	pump combination at interval i using n_p to denote the number of pumps in the station, p_i can be coded by a binary string in $\{0,1\}^{n_p}$, see *code* in Table 1 for $n_p = 5$
$c(p_i)$:	electric energy consumed by pump combination p_i at time interval i, see *power* in Table 1.

2.1.2 Pump maintenance cost (f_2)

Pump maintenance cost can be as important as electric energy cost or even more relevant. In Lansey et al. [4], the concept of the number of pump switches is introduced as an option to measure maintenance cost, i.e. a pump's wear can be measured indirectly through the number of times it has been switched on.

A pump switch is considered only if the pump was not working in the preceding time interval and it has been turned on. A pump that was already working in the preceding time interval and continues to be in the same state or is switched off, does not count as a pump switch for the present work. This way, pump maintenance cost can be reduced indirectly by reducing the number of pump switches.

The total number of pump switches N_s is simply calculated by adding the number of pump switches at every time interval. The number of pump switches between the last time interval of the preceding optimization period (day before) and the first time interval of the day being analyzed, is also computed. However, just half of that quantity is added to the total number of pump switches, in order to consider possible switches between two consecutive optimization periods, supposing there is a certain periodicity between consecutive pump schedules, as shown in Equation (2):

$$N_s = \sum_{i=1}^{24} |max\{0; (p_i - p_{i-1})\}| + \frac{|max\{0; p_1 - p_{24}\}|}{2} \tag{2}$$

where $|\cdot|$ represents the 1-norm of a vector.

2.1.3 Reservoir level variation (f_3)

There are three levels to be considered in the reservoir:

- a minimum level that guarantees enough pressure in the pipeline. This level must also be maintained for security reasons, since unexpected events, as a fire, may demand a large amount of water in a short time;
- a maximum level, compatible with the reservoir's capacity; and
- an initial level that has to be attained by the end of the optimization period.

Maximum and minimum levels are considered as constraints. Hence, at the end of each time interval, water level must end up in some position between the maximum level (h_{max}) and the minimum level (h_{min}). However, level variation between the beginning and the end of the optimization period (Δh), is stated as another objective to be minimized, since small variations do not necessarily make a solution not unacceptable, as shown in Equation (3):

$$\Delta h = \frac{\sum_{i=1}^{24}[D(p_i) - d_i]}{S} \tag{3}$$

$$h_i = \frac{h_{i-1} + [D(p_i) - d_i]}{S} \tag{4}$$

subject to

$$h_i \leq h_{max}$$
$$h_i \geq h_{min}$$

where

S : reservoir's surface, assumed constant

$D(p_i)$: discharge pumped at time interval i using pump combination p_i

d_i : water demand at time interval i.

Other constraints are considered as follows:

- The water source is supposed to supply enough water at any time and without additional costs.
- Maximum and minimum pressure constraints in the pipeline are always fulfilled, no matter at what level the reservoir is kept.
- Valves in the system are not considered.

2.1.4 Maximum power peak (f_4)

Some electricity companies charge their big clients according to a reserved power peak. This reserved power has a fixed charge, but an expensive additional charge, or penalty, is added when this reserved power is exceeded. The penalty is computed using the maximum peak power reached during the time considered for billing purpose. Therefore, reducing such penalties becomes very important. This work approaches this issue proposing the daily power peak P_{max} as another objective to be minimized. This is easily computed using Equation (5):

$$P_{max} = max[P(p_i)] \tag{5}$$

where:

$P(p_i)$: power at interval i using pump combination p_i, see Table 1.

2.2 Multi-objective pump scheduling problem

With each of the four objectives defined, the multi-objective pump scheduling problem can be stated as follows:

$$\text{Minimize } \mathbf{y} = \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), f_4(\mathbf{x})) \tag{6}$$

subject to

$h(x_i) \leq h_{max}$

$h(x_i) \geq h_{min}$, for each time interval i

$h(x_i)$: reservoir level by the end of time interval i; see Equation (4)

where:

f_1 : electric energy cost; see Equation (1)

f_2 : number of pump switches; see Equation (2)

f_3 : reservoir level variation; see Equation (3)

f_4 : maximum peak power; see Equation (5)

$\mathbf{x} \in \mathcal{X} \subseteq B^{24 \cdot n_p}$ is the decision vector, $B = \{0, 1\}$

$\mathbf{y} = (y_1, y_2, y_3, y_4) \in \mathcal{Y} \subset \mathbb{R}^4$ is the objective vector

In summary, the defined multi-objective pump scheduling problem considers the pumps' characteristics (pumping capacities) in order to satisfy water demand, while fulfilling other constraints such as the maximum and minimum levels in the reservoir. At the same time, electric energy cost, pump maintenance cost, maximum power peak, and level variation in the reservoir between the beginning and the end of the optimization period, are minimized. Clearly, in a 24 hour period, these objectives may be conflictive, e.g., for minimizing power peak a good schedule uses a small amount of energy during the whole day, while for minimizing cost it is better to consume energy while the cost is lower, trying to turn off the pumps during the high cost period.

3 Pump-Scheduling Optimization Using Parallel MOEAs

3.1 Discussion

In multi-objective optimization problems with several conflicting objectives there is no single optimal solution optimizing all objectives simultaneously, but rather a set of alternative solutions representing optimal trade-off between the various objectives. These solutions are known as Pareto-optimal or non-dominated solutions. A solution is said to be Pareto-optimal regarding a given subset of solutions if no other solution in the subset can be considered as better when all objectives are taking in account and no other preference information is provided. A solution is called as a true Pareto-optimal solution if it is non-dominated with respect to the whole search space. Pareto-optimal solutions form the so-called Pareto-optimal set and its image in the objective space is known as the Pareto-optimal front. A true Pareto-optimal set is composed of all the true Pareto-optimal solutions of the considered problem. The true Pareto-optimal set and its corresponding true Pareto-optimal front are denoted as P_{true} and PF_{true} respectively.

In many multi-objective optimization problems, knowledge about the true Pareto-optimal front helps the decision maker to choose the best compromise solution according to her preferences. Classical search methods handle multi-objective problems by means of scalarization techniques. Therefore, they really work with only one objective, formed by a composition of the other objectives. In this way, these methods aren't able to deal adequately with the simultaneous optimization of various conflicting objectives. Since traditional methods were developed with one objective in mind, they are not well suited to obtaining multiple solutions for a multi-objective problem in a single run.

In addition, exact methods are not adequate for searching solutions in huge search spaces because of the impossibility to explore the entire domain. For example, having a pump station comprised of five pumps and an optimization scope of 24 one-hour intervals, there are $2^{120} > 10^{36}$ solutions to explore for this optimal pump-scheduling problem. Problem constrains reduce the search space to a subset of feasible solutions, but the cardinality of such subset is still too large to be exhaustively analyzed by classical methods.

When computing the true Pareto-optimal front is computationally expensive or infeasible and exacts methods can't be applied, a good approximation to the real Pareto set is desirable. MOEAs do not guarantee identification of the true Pareto-optimal front, but they have demonstrated their ability to explore effectively and efficiently huge and complex search spaces, finding good approximations of the entire true Pareto-optimal set for many difficult multi-objective problems in a single run. Therefore, MOEAs become a promising alternative to solving the pump scheduling problem.

At each generation of a MOEA's execution, a certain set of trade-off solutions is identified. These solutions can be considered as Pareto-optimal regarding the current genetic population. This set is represented by $P_{current}(t)$, where t stands for the generation number. The Pareto front associated with $P_{current}(t)$ is denoted as $PF_{current}(t)$. It is expected that, while evolutionary process goes on, $P_{current}(t)$ approximates to P_{true}. Then, when a stop criterion is reached, the final solution set obtained by a MOEA has the potential to be a good enough approximation for P_{true}. This final solution set is represented by P_{known}, while PF_{known} denotes its associated Pareto front.

MOEAs are stochastic search algorithms. Hence, they don't guarantee to find the global optimum in a given execution. Therefore, to obtain a set of good solutions it is usual to perform various executions of a given MOEA and combine their reported results. Since multi-objective functions may be computationally expensive, the size of a MOEA population and the number of generations have to be limited in order to obtain solutions in a reasonable time.

Both population size and number of generations affect the quality of final solutions. Hence, it is desirable to provide a method that can explore a huge search space and/or carry out more generations in a given wall-clock time period, improving the quality of obtained solutions.

Parallelization of MOEAs appears to be a very good option to expand the search space an algorithm can examine [10]. Also, by interchanging individuals between several populations, it is possible to speed up the convergence of these algorithms to the true Pareto-optimal front.

To demonstrate these two statements for the pump scheduling problem, this work proposes a parallel asynchronous model for MOEAs. Using this model, parallel implementations of various outstanding MOEAs are tested. Then, results obtained by sequential and parallel MOEA executions were compared using a set of metrics. Thus, this work extends the comparison of the parallel MOEA implementation presented in [10] by using a new parallel model as well as a slightly different comparison method.

3.2 Parallel Asynchronous MOEAs

The parallel model for MOEA presented in this work is based on a multi-deme or island genetic algorithms approach [12, 13]. In multi-deme genetic algorithm, one population is divided into subpopulations called islands, regions or demes. Each subpopulation runs a separate genetic algorithm. The fitness value of an individual is calculated only relative to other individuals from the same region. Additionally to the basic operators of a genetic algorithm, a migration operator is introduced. This operator controls the exchange of individuals between islands. By dividing the population into regions and by specifying a migration policy, the multi-deme model can be adapted to various parallel architectures, especially for MIMD machines [12].

The proposed parallel framework consists of two kinds of processes, a collector and several pMOEAs (parallel Multi-objective Evolutionary Algorithms). The collector structure is presented in Algorithm 1. This procedure spawns all pMOEA processes and receives calculated solutions from them. In addition, the collector maintains an archive of the non-dominated solutions interchanged between demes and provides the final approximation set. This process does not utilize any evolutionary operator and does not interfere with the evolutionary process that is done by each pMOEA process. If the number of solutions in the collector process exceeds a desired number, an SPEA clustering procedure [14] is used to prune the set of solutions.

Meanwhile, pMOEAs are responsible for performing the real computational work. These pMOEAs basically differ from their sequential counterparts as follows:

1. pMOEAs use parameters received from the collector;
2. pMOEAs select individuals to migrate and send them to the other pMOEAs;
3. pMOEAs receive individuals from other islands and replace local solutions by received ones; and finally
4. pMOEAs send their final solution set to the collector responsible for the user interface work.

Algorithm 1 General structure of the collector procedure

```
Algorithm Collector ()
Initialize parameters
Spawn $H$ pMOEAs with their corresponding parameters
while $H > 0$ do
  Wait for pMOEAs solutions
  Collect received solutions in collector population $P$
  Eliminate covered solutions in $P$
  if the number of solutions in $P$ exceeds a given number then
    Prune $S$ applying the SPEA clustering procedure
  end if
  if results are marked as finals then
    $H = H - 1$
  end if
end while
Write final result out
```

Algorithm 2 presents the general framework for pMOEA processes. In each island, parameters are received from the collector and an initial population is generated. Then, as long as a stop criterion is not reached, the evolutionary process proceeds. At each generation, the migration condition is tested. In this work, the migration condition is based on a probability test. If the migration condition is true, migrants are selected.

Since there is no unique best solution to migrate, some criterion must be applied. In this work, elements to migrate are considered only among non-dominated solutions in the current generation. In some cases, the number of non-dominated solutions in a population may be very large. Hence, a parameter controlling the maximum number of migrants is provided. Therefore, migration of individuals is controlled by two parameters, one for the frequency of communications, and another for the number of migrants. In this case, migrating elements may represent a fraction of the non-dominated set of individuals that currently are in a MOEA's population. Thus, a number of individuals are randomly selected from the set of non-dominated solutions. In this way, all non-dominated solutions are treated equal and no other consideration is needed. After choosing individuals to migrate, these are broadcasted to all other processes.

Once the migration condition is tested and the corresponding procedure has been executed, it is checked if there are received solutions. If they are not, the procedure just goes on. Otherwise, replacement actions are taken. There are many alternatives to receive and replace individuals, among them:

1. apply selection to the union set of received migrants and current population;
2. randomly, replace elements in the genetic population;
3. randomly, replace elements dominated by the received ones.

From the above alternatives, the first requires major modifications of the original algorithm. The other approaches store received solutions in an auxiliary buffer and copy solutions to the genetic population if a condition is satisfied. The second approach permits the loss of good quality solutions

Algorithm 2 General structure of pMOEA procedures

```
Algorithm pMOEA()
Receive parameters of the MOEA execution plus the migration probability $p_{mig}$ and the maximum number
of non-dominated solution to migrate $n_{mig}$
Generates an initial population $P(0)$ at random, and set $t = 0$
while the stop criterion is not reached do
  $t = t + 1$
  Generates a new population $P(t)$ using a given MOEA procedure
  if condition to migrate is reached then
    Select migrants from $P(t)$ according to a specified policy
    Send migrants to all other processes
  end if
  if there are received solutions from other demes then
    Replace individuals in $P(t)$ by received ones according to an specified policy
  end if
end while
Send $P_{known}$ to collector with termination signal
```

when they are replaced by bad migrants. On the other hand, the third option ensures that non-dominated solutions will not be replaced at random. At the same time, this last method has a non-zero probability of not losing the worst solution, preserving genetic information of several fronts, while guaranteeing the maintenance of good solutions. When a pMOEA reaches its stop criterion, final non-dominated solutions are sent to the collector before finishing.

4 Experimental Parameters and Results

4.1 Parameters

In order to test the performance of pMOEA implementations in solving the multi-objective pump-scheduling problem, a test problem was chosen. The results of different executions of these implementations were compared under a set of selected metrics [15]. The multi-objective pump scheduling test problem parameters used in this work are based on technical characteristics of the main pumping station of a water supply system in Asuncion, Paraguay's capital, as described below:

- A set of $n_p = 5$ pumps is used.
- An elevated reservoir with the following dimensions is considered:
  - $area = 2,600m^2$,
  - $h_{max} = 7m$,
  - $h_{min} = 1m$,
  - $h_{initial} = 3m$,
  - $usable\ volume = 15,600m^3$.
- A demand curve based on statistical data of water consumption in a typical day, as presented in Figure 2.

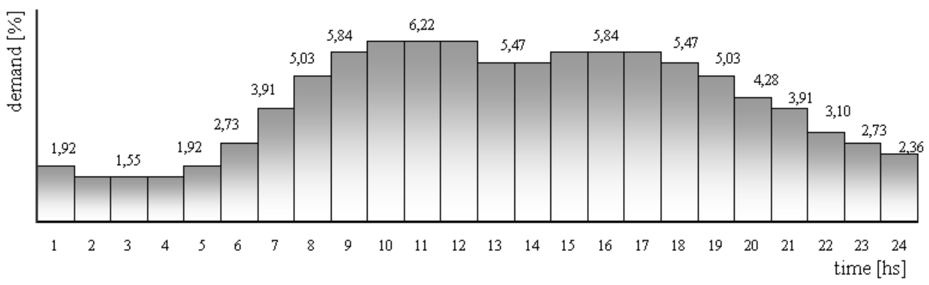

Figure 2: Water demand curve.

- An electricity cost structure with $C_h = 2C_l$.

With these specific values, sequential and parallel implementations of six algorithms were developed. Algorithms used in this work are: Multiple Objective Genetic Algorithm (MOGA) [16], Niched Pareto Genetic Algorithm (NPGA) [17], Non Dominated Sorting Genetic Algorithm (NSGA) [18], Strength Pareto Evolutionary Algorithm (SPEA) [14], NSGA-II [19] and Controlled Elitist NSGA-II (CNSGA-II) [20]. These algorithms were selected for consistency with previous authors' works [8, 10]. Also, these algorithms are good representative examples of different ages in MOEA research [21–23]. Detailed information on each implemented MOEA can be found in the referenced papers. A general background on various implementations of evolutionary algorithms is provided in [24]. It is expected that many possible solutions obtained by the algorithms do not fulfill hydraulic and technical constrains. Therefore, a heuristic method was combined with each implemented MOEA in order to transform a general solution into a feasible one [8].

For each considered MOEA, 10 different executions were carried out in their sequential and parallel versions using 1, 2, 4, 8 and 16 processes placed on different machines. Each execution used a different random seed. In parallel runs, a migration probability (p_{mig}) of 0.5 was used i.e., good solutions are transmitted to other processors in around half of the generations. In these executions, the maximum number of non-dominated solutions interchanged (n_{mig}) was 10% of the genetic population's size. Considering the parallel platform used, these values represent a good experimental trade-off between the frequency of interchanges and the number of interchanged

Algorithm	**Parameter**						
	p_m	p_c	N	N'	σ_{share}	t_{dom}	t_{red}
CNSGA-II	0.01	0.8	100	-	-	-	0.7
NSGA-II	0.01	0.8	100	-	-	-	-
SPEA	0.01	0.8	100	100	-	-	-
NSGA	0.01	0.8	100	-	0.41	-	-
NPGA	0.01	0.8	100	-	0.41	10%	-
FFGA	0.01	0.8	100	-	0.41	-	-

Table 2: Experimental MOEA parameters

solutions. A parallel MOEA using only 1 processor differs from a sequential execution because it uses a collector process storing non-dominated solutions as the evolutionary process proceeds, introducing a sort of elitism.

The implemented MOEAs use the following parameters [24] (see Table 2):

- Genetic population size (N): 100
- External population size (N'): 100
- Crossover probability (p_c): 0.8
- Mutation probability (p_m): 0.01
- Niching radius (σ_{share}): 0.41
- Domination pressure (t_{dom}): 10
- Reduction rate (t_{red}): 0.7

To conduct the experiments a cluster of twenty personal computers connected by a 100 Mbps Ethernet LAN was used. Each node in the cluster has one 700 MHz AMD K6-2 CPU with 128 MB of RAM and runs the Red Hat Linux 7.3 operating system and PVM 3.4.4. Programs were implemented using the C language.

4.2 Metrics

Having several optimization criteria, it is not clear what *quality of a solution* means. For example, it may refer to the closeness to the optimal front, the number of solutions, the spread of solutions, etc. In fact, in [25] three general goals are identified:

1. The size of the obtained Pareto front should be maximized, i.e., a wide range of Pareto solutions is preferred.
2. The distance of the obtained Pareto front to the true Pareto-optimal front should be minimized.
3. A good distribution of solutions, usually in objective space, is desirable.

Then, in order to evaluate experimental results from the implemented algorithms a set of metrics is used, this comparison itself being multi-objective. Explanations of the selected metrics can be found in [26]. However, it is important to note their most important aspects:

- Overall non-dominated vector generation ($ONVG$): This metric reports the number of solutions in PF_{known}. It is expected that good algorithms have a large number of solutions.

  $ONVG$ metric is defined as:
  $$ONVG = ||PF_{known}|| \tag{7}$$
  where $||\cdot||$ represents cardinality.

- Overall true non-dominated vector generation ($OTNVG$): counts the number of solutions in PF_{known} that are also in PF_{true} and is defined as:

$$OTNVG = ||\{\ x \mid x \in PF_{known}\ \wedge\ x\ \in\ PF_{true}\ \}|| \tag{8}$$

- Maximum Pareto Front Error (ME): This metric indicates the largest distance between a point in PF_{known} and its nearest neighbor in PF_{true}. Thus, all the other points in P_{known} are closer to P_{true} than this worst-case distance. A value of 0 is ideal. The ME metric is formally defined as follows:

$$ME = max(\{d_1^{min},\ d_2^{min}\ ,\dots,\ d_n^{min}\}) \tag{9}$$

  where d_i^{min} is the Euclidean distance (in objective space) between each vector in PF_{known} and its nearest neighbor in PF_{true} and n is the number of points in PF_{known}

- Spacing (S): this metric serves as indicator of the distribution of solutions in PF_{known} and is based on the average (arithmetic mean) distance of each point from its nearest neighbor.

  The spacing metric is mathematically defined as:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d^{min}} - d_i^{min})^2} \tag{10}$$

  where the average $\overline{d^{min}}$ is defined as:

$$\overline{d^{min}} = \frac{\sum_{i=1}^{n} d_i^{min}}{n} \tag{11}$$

  An ideal value for the spacing metric is 0.

Since some of these metrics require PF_{true} to be computed, an approximation of it was calculated from the non-dominated solutions in the union set of all obtained results. This experimental set is taken as the PF_{true} of reference.

$ONVG$ and $OTNVG$ are used in combination to measure the size and quality of the set of calculated solutions. Both are considered because $ONVG$ alone is a very poor indicator of the comparable quality between two sets. A given MOEA could produce a hundred of non-dominated points that are very close to the true Pareto Front, while another MOEA could produce a thousand points far from PF_{true}. With $ONVG$, the latter would appear to be better. Since an approximation set is used, $OTNVG$ appears to be a better metric to compare the quality of the solutions. Note that using both metrics, an error rate can be easily computed.

4.3 Results

Table 3 presents the average values for $ONVG$ obtained in 10 executions of each different implementation. The first column indicates if the implementation is sequential or parallel (pMOEA), providing the number of processors; the first row indicates the considered MOEA. It can be seen

Run	FFGA	NPGA	NSGA	NSGA-II	CNSGA-II	SPEA
Sequential	32.8	63.7	94.8	175	157.1	209.4
1 Processor	75.1	93.4	160.2	202.6	175.8	233.7
2 Processors	95.6	121.2	193.6	240.4	192.7	265.3
4 Processors	123.7	155.2	258.9	278	228.7	254.1
8 Processors	148.4	188.4	313.8	292.7	255.3	254.9
16 Processors	176.5	240.5	395.3	277.5	289.8	303.9

Table 3: Average values of $ONVG$ metric for the different runs

MOEA	**Run**	**Average**	**Median**	**Minimum**	**Maximum**	**Standard Deviation**
SPEA	Sequential	7.30	1.0	0.0	34.0	11.441
pSPEA	1 Processor	7.40	1.0	0.0	34.0	11.530
	2 Processors	18.40	5.0	0.0	44.0	19.603
	4 Processors	27.70	16.0	0.0	67.0	25.408
	8 Processors	33.40	23.0	3.0	98.0	30.160
	16 Processors	69.20	51.0	17.0	162.0	42.150
NSGA-II	Sequential	52.80	53.0	28.0	72.0	14.046
pNSGA-II	1 Processor	59.40	59.0	35.0	86.0	15.721
	2 Processors	76.80	75.0	47.0	103.0	17.838
	4 Processors	107.90	103.0	95.0	131.0	12.252
	8 Processors	117.10	109.0	80.0	172.0	27.614
	16 Processors	100.30	94.0	71.0	144.0	21.505
CNSGA-II	Sequential	40.70	41.0	0.0	54.0	16.418
pCNSGA-II	1 Processor	42.40	42.0	0.0	61.0	17.418
	2 Processors	56.50	64.0	6.0	76.0	20.807
	4 Processors	77.60	76.0	59.0	91.0	8.847
	8 Processors	102.80	105.0	80.0	114.0	10.412
	16 Processors	121.60	121.0	104.0	143.0	10.135

Table 4: Statistics values for $OTNVG$ metric

that the average $ONVG$ value grows with the number of processors. The use of a separate process storing non-dominated solutions improves the performance in this metric of first generation non-elitist MOEAs (FFGA, NPGA and NSGA), making them competitive with elitist approaches, especially as the number of processors grows. In fact, the best result for this metric is obtained by the parallel implementation of NSGA using 16 processors.

In spite of the growing number of solutions for parallel implementations of FFGA, NPGA and NSGA, they do not find any true Pareto solution. Specifically, $OTNVG = 0$ for every executed run of these 3 algorithms. Thus, these algorithms do not provide any solution to the true Pareto Front of reference.

In view of the inefficacy of the above MOEAs, only results obtained by SPEA, NSGA-II and CNSGA-II are discussed in what follows. A complete set of results taking into account other algorithms and metrics can be found in [23].

MOEA	**Run**	**Average**	**Median**	**Minimum**	**Maximum**	**Standard Deviation**
SPEA	Sequential	0.2825	0.1856	0.1539	0.4661	0.1310
pSPEA	1 Processor	0.2882	0.2430	0.1580	0.4661	0.1206
	2 Processors	0.2258	0.1648	0.1230	0.4259	0.1096
	4 Processors	0.2559	0.2591	0.1657	0.3431	0.0613
	8 Processors	0.2449	0.2420	0.1796	0.3514	0.0527
	16 Processors	0.2168	0.2002	0.1390	0.3436	0.0561
NSGA-II	Sequential	0.1118	0.1004	0.0776	0.1667	0.0276
pNSGA-II	1 Processor	0.1460	0.1363	0.0801	0.3006	0.0605
	2 Processors	0.1161	0.1040	0.0756	0.2018	0.0353
	4 Processors	0.1367	0.1239	0.0583	0.2711	0.0570
	8 Processors	0.1148	0.1031	0.0860	0.1494	0.0229
	16 Processors	0.1437	0.1268	0.0876	0.2434	0.0452
CNSGA-II	Sequential	0.1334	0.1213	0.0747	0.3439	0.0766
pCNSGA-II	1 Processor	0.1569	0.1231	0.0906	0.4086	0.0936
	2 Processors	0.2045	0.1791	0.1239	0.4421	0.0908
	4 Processors	0.1497	0.1263	0.0880	0.2442	0.0558
	8 Processors	0.1440	0.1283	0.0867	0.2394	0.0423
	16 Processors	0.1486	0.1239	0.0876	0.2897	0.0605

Table 5: Statistics for ME metric

In Table 4 some statistical values for the $OTNVG$ metric are presented. The average value is computed by adding the $OTNVG$ values of 10 executions and dividing by 10. Median, maximum, minimum and standard deviation values [27] are also computed. The median is the middle value of the set when ordered by value.

As can be noted, the best average value for this metric is obtained by a parallel implementation of CNSGA-II using 16 processors. On average an execution of pCNSGA-II with 16 processes reports 121.6 solutions that belongs to the true Pareto-optimal set. There are three times more solutions than with the corresponding sequential run.

Considering the $OTNVG$ metric for SPEA, the effect of parallelization is even more impressive. For SPEA, the average value is biased by some very good executions; thus, the median value is a better indicator for the distribution of $OTNVG$. Note that the SPEA median value for the $OTNVG$ metric is just 1 for the sequential implementation, increasing to 51 for pSPEA with 16 processors.

Minimum and maximum values of $OTNVG$ also improve with the number of processors. Note that as minimum and maximum pCNSGA-II values increase, the differences between them are reduced, as happens with the standard deviation, i.e., pCNSGA-II becomes more stable. On the contrary, the same difference and the standard deviation increase with the number of processors for pSPEA implementations, while no clear relation is observed with pNSGA-II. In conclusion, considering the number of true Pareto-optimal solutions found, pCNSGA-II may be considered as the most stable implementation as the number of processors increases.

Table 5 shows values for the metric ME. As can be noted, the average values are of the same order as the number of processors increases, yet the maximum ME decreases for the general case.

MOEA	Run	Average	Median	Minimum	Maximum	Standard Deviation
SPEA	Sequential	0.2825	0.1856	0.1539	0.4661	0.1310
pSPEA	Sequential	0.0321	0.0319	0.0237	0.0444	0.0057
	1 Processor	0.0313	0.0305	0.0265	0.0402	0.0037
	2 Processors	0.0297	0.0273	0.0256	0.0362	0.0041
	4 Processors	0.0277	0.0266	0.0229	0.0352	0.0041
	8 Processors	0.0302	0.0307	0.0230	0.0363	0.0036
	16 Processors	0.0314	0.0302	0.0242	0.0417	0.0053
NSGA-II	Sequential	0.0289	0.0293	0.0222	0.0330	0.0035
pNSGA-II	1 Processor	0.0275	0.0272	0.0206	0.0359	0.0046
	2 Processors	0.0238	0.0228	0.0185	0.0285	0.0032
	4 Processors	0.0218	0.0208	0.0176	0.0298	0.0038
	8 Processors	0.0211	0.0202	0.0188	0.0246	0.0019
	16 Processors	0.0249	0.0242	0.0208	0.0294	0.0027
CNSGA-II	Sequential	0.0303	0.0286	0.0245	0.0446	0.0059
pCNSGA-II	1 Processor	0.0278	0.0249	0.0216	0.0373	0.0056
	2 Processors	0.0332	0.0296	0.0258	0.0471	0.0071
	4 Processors	0.0266	0.0270	0.0205	0.0296	0.0028
	8 Processors	0.0251	0.0243	0.0210	0.0316	0.0039
	16 Processors	0.0241	0.0243	0.0197	0.0290	0.0027

Table 6: Statistics for the Spacing metric

To evaluate this metric the column maximum of Table 5 will be considered. This column provides the value of the worst ME considering 10 executions of a certain MOEA. Therefore, it is an upper bound on the error strip for a given run. Consequently, the minimum of such values provides the best worst-value; it indicates that other solutions of the considered algorithm are closer to the true Pareto-optimal front. With these considerations, it can be noted that pNSGA-II using 8 processors provides solution sets with the lowest upper ME.

The last metric to be considered is Spacing. Table 6 shows results for this metric. It can be seen that values are very close to the optimum (0). Taking into account average S values, parallel MOEAs are better than their sequential versions, and the best value is obtained with pNSGA-II using 8 processors.

In summary, parallel implementation of MOEAs find a larger number of solutions ($ONVG$ & $OTNVG$), better solutions ($OTNVG$ & ME), and are also better distributed (S) than their sequential counterparts.

5 Conclusions

A parallel asynchronous model for multi-objective evolutionary optimization was presented and applied to six recognized MOEAs to solve an optimal pump-scheduling problem considering four minimization objectives. Various executions of parallel and sequential implementations were conducted and their results compared, using a set of metrics.

Algorithm	**Run**	*OTNVG*	*ONVG*	*ME*	*Spacing*
CNSGA-II	16 Processes	121.6	289.8	0.1486	0.0241
NSGA-II	8 Processes	117.1	292.7	0.1148	0.0211
NSGA-II	4 Processes	107.9	278.0	0.1367	0.0218
CNSGA-II	8 Processes	102.8	255.3	0.1440	0.0251
NSGA-II	16 Processes	100.3	277.5	0.1437	0.0249

Table 7: Ranking of top 5 runs

To have a notion of goodness of the different solution sets, the following lexicographical order of average metric values is finally considered:

1. $OTNVG$,
2. $ONVG$,
3. ME,
4. *Spacing*.

Table 7 shows a ranking of algorithms using the aforementioned lexicographic order of metrics. As can be seen, the best position is obtained by CNSGA-II with 16 processors. However, it should be emphasized that another algorithm would be considered the best one using another preference between metrics.

Our experimental results have shown that parallel evolutionary algorithms are capable of providing a larger number of better alternatives for pump scheduling than their sequential counterparts.

We also showed that solutions provided by first generation MOEAs are worse than the ones obtained by elitism-based MOEAs considering the number of solutions they found in a reference PF_{true}.

Improvement of parallel over sequential MOEAs has various reasons. First, parallel implementations explore in larger areas than sequential ones, since they handle a larger number of populations. Second, the cooperation between the different populations produces synergy in searching for good solutions. Finally, the collector procedure reinforces elitism in the evolutionary process of SPEA, NSGA-II and CNSGA-II, storing solutions that otherwise may be lost.

References

[1] L. Mays, *Water Distribution Systems Handbook.* New York: McGraw-Hill, 2000.

[2] T. M. W. et all., *Advanced Water Distribution Modeling and Management.* Waterbury, CT: Haestead Press, 2003.

[3] L. E. Ormsbee and K. E. Lansey, “Optimal Control of Water Supply Pumping Systems,” *Water Resources Planning and Management Journal*, 1994.

[4] K. E. Lansey and K. Awumah, “Optimal Pump Operations Considering Pump Switches,” *Water Resources Planning and Management Journal*, vol. 1, no. 120, 1994.

[5] D. Mackle, A. Savic, and G. A. Walters, “Application of Genetic Algorithms to Pump Scheduling for Water Supply,” in *GALESIA 95*, (London, UK), 1995.

[6] D. A. Savic, G. A. Walters, and M. Schwab, “Multiobjective Genetic Algorithms for Pump Scheduling in Water Supply,” in *AISB International Workshop on Evolutionary Computing. Lecture Notes in Computer Science 1305*, (Berlin), pp. 227–236, Springer-Verlag, Apr. 1997.

[7] W. Schaetzen, “A Genetic Algorithm Approach for Pump Scheduling in Water Supply Systems,” tech. rep., Water Systems group, School of Engineering, University of Exeter, United Kingdom, 1998.

[8] C. von Lücken, A. Sotelo, and B. Barán, “Multiobjective Evolutionary Algorithms in Pump Scheduling Optimisation,” in *Proceedings of the Third International Conference on Engineering Computational Technology* (B. Topping and Z. Bittnar, eds.), (Stirling, United Kingdom), Civil-Comp Press, 2002.

[9] C. von Lücken and B. Barán, “Multi-objective Evolutionary Algorithms Experimental Comparison,” in *Conferencia Internacional de Tecnologías y Aplicaciones Informáticas*, (Asunción, Paraguay), 2001.

[10] C. von Lücken, B. Barán, and A. Sotelo, “Pump Scheduling Optimisation Using Parallel Multiobjective Evolutionary Algorithms,” in *XXVII Conferencia Latinoamericana de Informática CLEI-2003*, (La Paz, Bolivia), 2003.

[11] F. M. Dolqachev and N. N. Pashkov, *Hydraulics and Hydraulic Machines.* Mir, 1985.

[12] E. Cantu-Paz, “Designing Efficient and Accurate Parallel Genetic Algorithms,” Tech. Rep. 2108, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1999.

[13] D. A. van Veldhuizen, J. B. Zydallis, and G. B. Lamont, “Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 144–173, Apr. 2003.

[14] E. Zitzler and L. Thiele, “Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach,” *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 257–271, November 1999.

[15] D. A. van Veldhuizen and G. B. Lamont, “On Measuring Multiobjective Evolutionary Algorithm Performance,” in *2000 Congress on Evolutionary Computation*, vol. 1, (Piscataway, New Jersey), pp. 204–211, IEEE Service Center, July 2000.

[16] C. M. Fonseca and P. J. Fleming, “Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization,” in *Proceedings of the Fifth International Conference on Genetic Algorithms* (S. Forrest, ed.), (San Mateo, California), pp. 416–423, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, 1993.

[17] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, vol. 1, (Piscataway, New Jersey), pp. 82–87, IEEE Service Center, June 1994.

[18] N. Srinivas and K. Deb, "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, pp. 221–248, Fall 1994.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, April 2002.

[20] K. Deb and T. Goel, "Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence," in *First International Conference on Evolutionary Multi-Criterion Optimization* (E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, eds.), pp. 67–81, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[21] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, pp. 173–195, Summer 2000.

[22] R. Purshouse and P. Fleming, "The Multi-Objective Genetic Algorithm Applied to Benchmark Problems—An Analysis," Tech. Rep. 796, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, August 2001.

[23] C. von Lücken, "Algoritmos Evolutivos para Optimización Multi-objetivo: un Estudio Comparativo en un Ambiente Paralelo Asíncrono," Master's thesis, Universidad Nacional de Asunción, San Lorenzo, Paraguay, December 2003.

[24] C. A. Coello Coello, "A Short Tutorial on Evolutionary Multiobjective Optimization," in *First International Conference on Evolutionary Multi-Criterion Optimization* (E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, eds.), pp. 21–40, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[25] J. Knowles and D. Corne, "On Metrics for Comparing Nondominated Sets," in *Congress on Evolutionary Computation (CEC'2002)*, vol. 1, (Piscataway, New Jersey), pp. 711–716, IEEE Service Center, May 2002.

[26] D. A. van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[27] I. Miller, J. Freund, and R. Johnson, *Probability and Statistics for Engineers, Four Ed.* New York: Prentice Hall, 1998.