Multiobjective Network Design Optimisation Using Parallel Evolutionary Algorithms

Susana Duarte Flores and Benjamín Barán Cegla
Centro Nacional de Computación, Universidad Nacional de Asunción.
San Lorenzo, Paraguay. C.C. 1439.
sduarte@cnc.una.py, bbaran@cnc.una.py

ABSTRACT

This paper proposes a new parallel asynchronous version of the Strength Pareto Evolutionary Algorithm (SPEA) implemented over a network of inexpensive personal computers. The objective of the implementation is the optimal design of communication networks in the presence of multiple and even conflicting objectives. It tries to find the best topologies for the network, given the location of the Communication Centres (nodes), and the cost and reliability of the potential links between them. The proposed tool provides the system designer with a complete set of valid solutions, easing the process of decision making. In this set, known as the *Pareto optimal set*, all solutions are feasible and no solution is the best, if all objectives are being considered. Experimental results validate the suggested approach.

Keywords: multiobjective optimisation, network design, asynchronous parallel multiobjective evolutionary algorithms.

RESUMEN

Este trabajo propone una innovadora versión paralela asíncrona del *Strength Pareto Evolutionary Algorithm* (SPEA), implementada sobre una red de computadoras personales de bajo costo. El objetivo de la implementación es el diseño óptimo de redes de computadoras en presencia de múltiples objetivos que incluso pueden estar en conflicto entre sí. Se trata de obtener las mejores topologías posibles para la red, dada la ubicación de los centros de comunicación (nodos), y el costo y confiabilidad de los enlaces potenciales entre ellos. La herramienta propuesta provee al diseñador de sistemas de un completo conjunto de soluciones válidas asistiendo al proceso de toma de decisiones. En este conjunto, conocido como *conjunto Pareto óptimo*, todas las soluciones son factibles y ninguna puede considerarse mejor a las demás si se toman en cuenta todos los objetivos. Resultados experimentales demuestran la validez de la propuesta sugerida.

Palabras claves: optimización multiobjetivo, diseño de redes, algoritmos evolutivos paralelos asíncronos.

1 INTRODUCTION

The design of communication networks is a typical problem from the field of operational research. Clearly, it can be classified as a multiobjective problem. The objective of the designer is to optimise a set of conflicting objectives: reliability, cost, delays, throughput, capacity, etc, while maintaining restrictions over another set of requirements: minimum reliability, maximum cost, maximum acceptable delay, minimum speed, etc. This problem is known to be NP-Hard [2].

Many approaches have been designed to address this problem, some of them based on various kinds of graph perturbation heuristics [14, 1], and others founded in techniques from artificial intelligence (tabu search [12], simulated annealing [13] and genetic algorithms [5, 6, 10, 9]). An interesting summary of these methods can be found in [9]. To shorten the discussion it is useful to say that:

a) none of them treats the problem as a multiobjective problem, but they would rather choose an objective to optimise, leaving the others as restrictions;
b) all of them can be applied only to networks of restrained magnitude, and in very restricted situations. As the size of commercial systems grows there is a complete lack of tools to aid in the designing process; and the methodology of trial & error that has been applied is neither effective nor efficient.

In the present work we propose an implementation of two versions of the Strength Pareto Evolutionary Algorithm (SPEA) [16], (a sequential and a parallel one), and we examine and contrast the results obtained with both. We have chosen the SPEA because it implements elitism through the maintenance of an external population of best solutions found during the whole generational loop; then, convergence is guaranteed [15].

The rest of this work is organised in the following way: section 2 presents a very concise outline of the theory of multiobjective optimisation. Section 3 introduces the problem to be solved with its restrictions and generalities. Section 4 discusses the test problem. Section 5 and 6 contain descriptions of the implementations (sequential and parallel version, respectively). Section 7 includes performance metrics used for the testing procedure. Section 8 presents experimental results. And, finally, section 9 emits some conclusions and directions for further work.

2 MULTIOBJECTIVE OPTIMISATION PROBLEMS

Multiobjective optimisation can be applied to a wide range of problems. In particular we consider the problem of the backbone design of a computer network. One of the key objectives of the designer is to minimise total costs, while another one is to maximise reliability. Depending on the type of normal traffic, other objectives can be minimise delays, maximise throughput, etc. Every objective can be expressed explicitly as an objective to optimise or can be included as a restriction for the optimisation process. Considering this fact a multiobjective problem can be defined as follows.

Multiobjective Optimisation Problem **(MOP).** A general MOP includes a set of *n* parameters (decision variables), a set of *k* objective functions, and a set of *m* restrictions. Objective functions and restrictions are functions of decision variables. This can be expressed as:

$$\begin{array}{ll} \textit{Optimise} & \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_k(\boldsymbol{x})) \\ \textit{subject to} & \boldsymbol{e}(\boldsymbol{x}) = (e_1(\boldsymbol{x}), e_2(\boldsymbol{x}), \dots, e_m(\boldsymbol{x})) \geq \boldsymbol{0} \\ \textit{where} & \boldsymbol{x} = (x_1, x_2, \dots, x_n) \in \boldsymbol{X} \\ & \boldsymbol{y} = (y_1, y_2, \dots, y_k) \in \boldsymbol{Y} \end{array} \tag{1}$$

and where $\boldsymbol{x}$ is the decision vector and $\boldsymbol{y}$ is the objective vector. $\boldsymbol{X}$ denotes the decision space and the objective space is denoted by $\boldsymbol{Y}$. Depending on the problem at hand "Optimise" could mean minimise or maximise.

The set of restrictions $\boldsymbol{e}(\boldsymbol{x}) \geq \boldsymbol{0}$ determines the set of feasible solutions $\boldsymbol{X}_f$ and its corresponding set of feasible objective vectors $\boldsymbol{Y}_f$.

From this definition it follows that every solution for the problem consists of a *n*-tuple $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, that yields the objective vector $\boldsymbol{y} = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_k(\boldsymbol{x}))$, where every $\boldsymbol{x}$ must comply with the set of restrictions $\boldsymbol{e}(\boldsymbol{x})$. The optimisation problem consists in finding the $\boldsymbol{x}$ that has the "best" $\boldsymbol{f}(\boldsymbol{x})$. In general, there is not one "best" solution, but a set of solutions, none of which can be considered better than others if all objectives are considered at the same time. This derives from the fact that there could be –and mostly there are– conflicts between the different objectives that compose the problem. Thus a new concept of optimality should be established for MOPs.

In common mono-objective optimisation problems the set of feasible decision variables is completely ordered by the objective function *f*. The goal is simply to find the value –or set of values– that lead to the optimal values of *f*. In contrast, in multiobjective optimisation the feasible decision vector set is partially ordered (i.e. there exist a decision vector $\boldsymbol{a}$ and a decision vector $\boldsymbol{b}$ and $\boldsymbol{f}(\boldsymbol{a})$ cannot be considered better than $\boldsymbol{f}(\boldsymbol{b})$, neither $\boldsymbol{f}(\boldsymbol{b})$ is better than $\boldsymbol{f}(\boldsymbol{a})$. Then, mathematically the relations $=$, $\leq$ and $\geq$ should be extended. This could be done in the following way:

Given two decision vectors $\boldsymbol{u} \in \boldsymbol{X}$ and $\boldsymbol{v} \in \boldsymbol{X}$

$$\begin{aligned}&\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{v}) \text{ if and only if } \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) = f_i(\boldsymbol{v})\\ &\boldsymbol{f}(\boldsymbol{u}) \geq \boldsymbol{f}(\boldsymbol{v}) \text{ if and only if } \forall i \in \{1, 2, \ldots, k\}: f_i(\boldsymbol{u}) \geq f_i(\boldsymbol{v})\\ &\boldsymbol{f}(\boldsymbol{u}) > \boldsymbol{f}(\boldsymbol{v}) \text{ if and only if } \boldsymbol{f}(\boldsymbol{u}) \geq \boldsymbol{f}(\boldsymbol{v}) \wedge \boldsymbol{f}(\boldsymbol{u}) \neq \boldsymbol{f}(\boldsymbol{v})\end{aligned} \tag{2}$$

The relations $\leq$ and < could be defined in similar ways.
Then, given two decision vectors of a MOP, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ they comply to one of three conditions: either $\boldsymbol{f}(\boldsymbol{x}_1) > \boldsymbol{f}(\boldsymbol{x}_2)$, or $\boldsymbol{f}(\boldsymbol{x}_1) > \boldsymbol{f}(\boldsymbol{x}_2)$, or $\boldsymbol{f}(\boldsymbol{x}_1) \not\geq \boldsymbol{f}(\boldsymbol{x}_2) \wedge \boldsymbol{f}(\boldsymbol{x}_2) \not\geq \boldsymbol{f}(\boldsymbol{x}_1)$. And this is expressed with the following symbols.
Pareto Dominance. Given two objective vectors $\boldsymbol{a}$ y $\boldsymbol{b}$,

$$\begin{array}{lll}\boldsymbol{a} \succ \boldsymbol{b} \text{ (}\boldsymbol{a}\text{ dominates }\boldsymbol{b}\text{)} & \text{if and only if} & \boldsymbol{a} > \boldsymbol{b}\\ \boldsymbol{b} \succ \boldsymbol{a} \text{ (}\boldsymbol{b}\text{ dominates }\boldsymbol{a}\text{)} & \text{if and only if} & \boldsymbol{b} > \boldsymbol{a}\\ \boldsymbol{a} \sim \boldsymbol{b} \text{ (}\boldsymbol{a}\text{ and }\boldsymbol{b}\text{ are not comparable)} & \text{if and only if} & \boldsymbol{a} \not\geq \boldsymbol{b} \wedge \boldsymbol{b} \not\geq \boldsymbol{a}\end{array} \tag{3}$$

Definitions for the minimisation problem ($\prec$, $\sim$) could be derived in analogous fashion.

At this point the concept of Pareto optimality can be introduced. A solution is said to be Pareto optimal or "non inferior" if any objective cannot be improved without degrading others.
Pareto Optimality: A decision vector $\boldsymbol{x} \in \boldsymbol{X}_p$ and its corresponding objective vector $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{Y}_f$ is non-dominated with respect to a set $\boldsymbol{A} \subseteq \boldsymbol{Y}_f$ if and only if

$$\forall\ \boldsymbol{a} \in \boldsymbol{A} : (\boldsymbol{y} \succ \boldsymbol{a} \vee \boldsymbol{y} \sim \boldsymbol{a}) \tag{4}$$

When $\boldsymbol{y}$ is non-dominated with respect to the whole set $\boldsymbol{Y}_f$ –and only in that case– $\boldsymbol{x}$ is a **Pareto optimal solution,** $\boldsymbol{x} \in \boldsymbol{X}_p$ (the **Pareto optimal set**) and the corresponding $\boldsymbol{y}$ is part f the **Pareto optimal front** $\boldsymbol{Y}_p$.
Dealing with Pareto optimal solutions, it is clear that they are non-comparable. This points to the fact that a MOP does not always have a single solution, but a set of compromise solutions. None of these solutions can be defined as "the best", unless other information is added (i.e. weight of every objective).

Pareto Optimal set and Pareto front. Given the set of feasible decision vectors $\boldsymbol{X}_f$, the function $p(\boldsymbol{X}_f)$ returns the set of non dominated decision vectors that belong to $\boldsymbol{X}_p$, i.e.:

$$p(\boldsymbol{X}_f) = \{\ \boldsymbol{x} \in \boldsymbol{X}_f \mid \boldsymbol{x} \text{ is non dominated with respect to } \boldsymbol{X}_f\ \} \tag{5}$$

The set $\boldsymbol{X}_p = p(\boldsymbol{X}_f)$ is known as the Pareto optimal set, while the corresponding set of objective vectors $\boldsymbol{Y}_p = \boldsymbol{f}(p(\boldsymbol{X}_f))$ constitutes the Pareto optimal front.

3 STATEMENT OF THE PROBLEM

A network can be modelled by a probabilistic undirected graph [6] G = (V, L, p), where:

- V is the node set.
- L is the arc set. The cardinality of L is also the number of possible links and can be expressed as

$$n = |L| = \frac{|V|\left(|V|-1\right)}{2} \tag{6}$$

- And p is the reliability of links.

The problem of network design consists of choosing the links given the communication centres locations, and the potential links with their cost and reliability. The resulting network should acquire a certain set of values for the objectives and comply with another set of requirements.
From the definitions mentioned in the previous section, it is obvious that the problem of backbone network design optimisation can be expressed as a multiobjective optimisation problem. As the problem can be as big as a designer states it (i.e., he can choose as many objectives as he wants, he can have as many kind of links as technology and budget lets him), there is a need to place limits on it. In the present work, it is stated as the optimisation of two objectives (k = 2): reliability and cost. The fact that every network topology must be connected is expressed by restricting reliability to positive values, then the proposed solutions must meet only a minimum reliability requirement (m = 1). It is assumed one bi-directional link between each pair of nodes (redundancy is not allowed), thus the potential links between every pair of nodes are the decision variables. Every decision variable $\boldsymbol{x}$ is composed of a n-tuple (x_1, x_2, ..., x_n).
The constraints on redundancy and number of objectives are only apparent and do not make the problem less general, as the addition of new objectives is a trivial problem, even though it may require more computational resources. Also,

redundant links can be treated as another kind of link, with its own cost and reliability [10]. Then, the expression of the problem is:

$$\begin{array}{ll} \textit{Optimise} & \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ \textit{subject to} & e_1(\mathbf{x}) > \mathbf{0} \end{array} \qquad \textbf{(7)}$$

where:

- $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{X}$ is the decision vector; every $x_i \{0, 1, \ldots, t\}$ represents a (type of) link between a pair of nodes and t is the number of different link types (0 is used to indicate the absence of connection);
- $\mathbf{y} = (y_1, y_2,) \in \mathbf{Y}$ is the objective vector;
- $f_1(\mathbf{x})$ is the reliability corresponding to the configuration $\mathbf{x}$;
- $f_2(\mathbf{x})$ is the cost function of the same configuration $\mathbf{x}$;
- $e_1(\mathbf{x})$ refers to the minimum acceptable reliability;

Although parameters like speed, capacity and throughput are important for innovative applications, the main network design objectives are cost and reliability [10]. Both functions were studied in almost all papers found referring to design optimisation problems. Sometimes, the problem was declared as the minimisation of cost, subject to a reliability constraint; while some others as the maximisation of reliability subject to a cost constraint. Even some times [6] a weighted sum approach of both objectives was suggested. But the multiobjective nature of the problem has not been previously evidenced.

The concept of reliability depicts the probability of a system to have an expected performance over a time interval. So, the reliability of a system depends on its configuration and the reliability of its components. There are many methods and metrics to measure the reliability. For our instance of the problem, to ensure that there is always a communication path between every pair of nodes in the network, the all-terminal reliability metric was chosen (i.e. the network forms at least a spanning tree) [2, 10]. The reliability calculation is done via Monte Carlo simulations because there are not other methods that can give good results in acceptable time (the problem of computing the reliability of a network is, in its time, NP-Hard [2]). The pseudocode for the reliability calculation procedure is:

```
Procedure Reliability($\mathbf{x}$)
Begin
    $i = 0$
    $counter = 0$
    While $i < Number\_MC\_replications$
        Randomly generate a network NET, from $\mathbf{x}$
        If NET forms a spanning tree
            $counter = counter + 1$
        End If
        $i = i + 1$
    End While
    $Computed\_reliability = counter/Number\_MC\_replications$
End
```

Pseudocode 1. Reliability Calculation

The cost of every configuration is calculated adding up the costs of every link added to the topology. Each link has a cost that is the product of the distance it covers and the cost per distance unit, i.e. the cost of a configuration $\mathbf{x}$ is:

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} dist\_unit_i * \cos t(c_i, i) \qquad \textbf{(8)}$$

where:

- $dist\_unit_i$ is the distance –in units– that the link i covers; and
- $cost(c_i, i)$ is the cost –per unit– of the type of link c_i.

In order to solve the problem the following assumptions are necessary [5]:

- The nodes are perfectly reliable (failure of nodes can be simulated by a failure of its incident links).
- The cost and reliability of each potential link are known.
- The links can be in only one of two possible states: operational or failed.
- Links fail independently, i.e. the failure in a link does not imply the failure of another.
- No repair is considered, i.e. when a link fails it is not repaired and put into operation afterwards.

4 TEST PROBLEM

The test problem is based on the expansion of the ULAK-NET network, first published in [5]. It is a simplified version of a real network design problem conceived to link, using distinct types of fiber, 19 universities and research centres located in 9 different cities of Turkey. It was chosen because it is the largest published example found during our research. Besides, the results of this example were available; therefore, they were used to compare them with our experimental results.

Table 1 shows the distance matrix in kilometres for each pair of nodes. Three types (t = 3) of fiber optic links are considered; their costs and reliabilities are (333 $/km, 96%), (433 $/km, 97.5%) and (583 $/km, 99%) respectively. Then, the size of the search space is composed of $3.8*10^{81}\left(t^{\frac{|V|*(|V|-1)}{2}}\right)$ individuals of the form (x_1,…, x_{171}) with their corresponding cost and reliability.

	v1	***v2***	***v3***	***v4***	***v5***	***v6***	***v7***	***v8***	***v9***	***v10***	***v11***	***v12***	***v13***	***v14***	***v15***	***v16***	***v17***	***v18***	***v19***
v1	-	111	126	120	122	115	116	132	346	968	343	344	106	107	105	454	613	828	1261
v2		-	15	15	17	5	6	243	458	1079	454	456	10	11	5	565	724	939	1342
v3			-	15	17	13	14	258	473	1094	469	471	25	26	23	580	740	954	1357
v4				-	2	5	6	248	460	1082	456	457	12	13	15	570	730	943	1353
v5					-	8	9	251	463	1085	459	460	15	16	18	573	733	946	1355
v6						-	1	246	457	1080	454	455	10	9	12	568	728	940	1350
v7							-	245	456	1079	453	454	9	8	11	567	727	939	1351
v8								-	384	383	380	381	235	236	240	322	542	831	1301
v9									-	766	3	4	450	451	453	580	542	487	920
v10										-	763	764	1074	1075	1077	1345	1307	972	624
v11											-	1	450	451	453	582	544	489	921
v12												-	449	450	452	583	545	490	922
v13													-	1	4	560	720	932	1337
v14														-	3	561	721	933	1338
v15															-	563	723	934	1340
v16																-	469	898	1424
v17																	-	553	1079
v18																		-	526
v19																			-

Table 1. Distance matrix for the test problem.

5 DESCRIPTION OF THE IMPLEMENTATION

For the application of the Multiobjective Evolutionary Algorithm (MOEA), each possible solution $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, was coded using a string of integer numbers, $x_i \in \{0, 1, ..., t\}$. To obtain the string, an adjacency matrix of the graph that models the network was written [7]. As this matrix is symmetrical, only the upper triangular part was inserted into the chromosome. For example, to code the network of figure 1, the matrix of figure 2 was used.

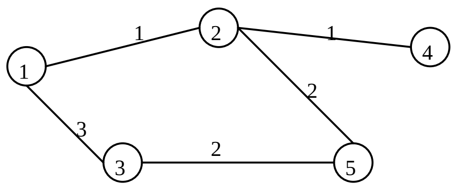

Figure 1. Graphical representation of a computer network backbone.

	1	**2**	**3**	**4**	**5**
1	0	1	3	0	0
2	1	0	0	1	2
3	3	0	0	0	2
4	0	1	0	0	2
5	0	2	2	0	0

Figure 2. Adjacency matrix for network of figure 1.

The final representation $\boldsymbol{x}$, taken from the upper triangular part –and not considering the diagonal–, is the string 1300012022.
Continuing with the test problem the calculation of reliability is accomplished with Monte Carlo simulations [8, 10]. Only 10000 replications were made due to the high computational cost. Solutions not achieving the minimum reliability requirement are not inserted in the external population, even though they may be part of the purported Pareto set. Anyway, they remain in the current population because feasible children can be generated from them. As it was formerly stated, the total cost of a network is the sum of the costs of its links.
Following the definition of SPEA [16], two populations of individuals are kept, the first one (depicted as P) is known as the current population, while the second one or external non-dominated set P' maintains every non-dominated individual found so far.
The process of finding the non-dominated individuals in P is based on the concept of dominance expressed in section 2. Every time a new non-dominated individual is found, it is compared against the members in P'; if it is a new solution, it is inserted into P'. The number of individuals in P is N and remains constant during the whole generational loop, but the number of individuals in P' may change. P' can not have more individuals than a previously stated number of N'. If the size of P' is greater than N', clustering should be performed. The process of clustering [11, 16] has been implemented, but was never used because the maximum size of the external population was never reached.
SPEA differs from the traditional genetic algorithm only in the way fitness is assigned to individuals. The computation of the fitness value follows the procedure explained in [16]. Every member of P' has a fitness equal to the number of individuals in P it dominates, plus one; while every member of P has as fitness the sum of fitness of members of P' that dominates him. In this way it is ensured that members of P' have a better fitness value than members of P. Notice that this is in the context of fitness minimisation. The fitness assignment process, as well as clustering, induces the maintenance of diversity [16].
Selection is implemented with binary tournaments, and the next generation is created via one point crossover.
The mutation operator takes $m\%$ individuals from the population and changes every allele from its chromosome with probability 0.3.
The parameters of the SPEA are the following:

- Population size (N): 100 individuals.
- External non-dominated set size (N'): 100 individuals.
- Maximum number of generations (g_{max}): 5000.
- Crossover probability (p_c): 1.
- Mutation rate (r_m): 0.3.
- Percentage of population mutated in each generation ($m\%$): 5%.

The initial population for the algorithm was generated randomly, but individuals with less links have a greater probability of being inserted into the initial population (this approach has shown its usefulness to speed up convergence).
A stop criteria has also been implemented. The algorithm continues with its generational cycle if new individuals are being inserted into P' every 10 generations, or if the maximum number of generations (5000) has not been reached. Those numbers were chosen for the first implementation, but a complete study still should be done.
When the algorithm stops it has its results in P', which is called the known Pareto set $\boldsymbol{X}_{known}$; the corresponding objective vectors $\boldsymbol{Y}_{known} = \boldsymbol{f}(\boldsymbol{X}_{known})$ is the known Pareto front. Pseudocode 2 presents the sequential version.

```
Procedure NetworkDesignSeqSPEA()
Begin.
    Read initial input parameters for the SPEA: $s, g_{max}, p_c, r_m, m\%$
    Read parameters of the problem: *cost* and *reliability* of links
    Read randomly generated initial population $P$
    $Generation\_Counter = 1$
    While $StopCondition$ is not reached and $Generation\_Counter < g_{max}$
        Compute the value of each objective function for each individual
        Find non-dominated individuals in current population $P$
        Update external non-dominated set $P'$
        If the number of externally stored non-dominated solutions exceeds a given maximum $N'$
            prune $P'$ by means of clustering
        End If
        Calculate the fitness of each individual in $P$ and $P'$
        Select individuals from the union set $P + P'$ until the mating pool is filled
        Apply specific crossover and mutation operators to generate the new set $P$
        $Generation\_Counter = Generation\_Counter + 1$
    End While
    Write out the individuals from $P'$ as $\boldsymbol{X}_{known}$ with their corresponding $\boldsymbol{Y}_{known}$
End
```
Pseudocode 2. Sequential version of SPEA

6 PARALLEL VERSION

As the calculation of objective values, specially the computation of reliability, is very time consuming, the execution time of the proposed solution can be improved running it in a distributed environment.

Moreover, the total implementation costs can be reduced significantly if we use a network of inexpensive personal computer instead of a massively parallel supercomputer.

The main algorithm consists of two kinds of processes, an organiser and several PSPEAs (Parallel Strength Pareto Evolutionary Algorithm processes). There is only one organiser, with the responsibility of creating all the PSPEAs and collecting the final results. The PSPEAs do the real work.

The pseudocode for the organiser process is the following:

```
Procedure NetworkDesignOrganiserParallelSpea()
Begin
    Spawn $H$ PSPEAs
    $flag\_counter = 0$
    While $flag\_counter < H$
        Wait for flag from $H$ processes
        If a flag is received
            Collect the results from the process that has sent the flag and kill him
            $flag\_counter = flag\_counter + 1$
        End If
    End While
    Do the union operation over all sets obtained from the PSPEAS
    Apply the Pareto dominance concept over the resulting set to obtain $\boldsymbol{X}_{known}$ with its corresponding $\boldsymbol{Y}_{known}$
    Write final result out.
End
```
Pseudocode 3. Parallel version of SPEA. Organiser Procedure.

Straightaway, the PSPEAs are discussed. Given a distributed system with H processors, in each processor h, $h \in \{1, .., H\}$, two populations are kept $P_h(g)$ and $P'_h(g)$. The population $P_h(g)$ contains the members generated by crossover in the previous generation g-1; while $P'_h(g)$ is the external set of non-dominated solutions found from the beginning of the generational loop, until generation g is reached.

Once found, the new solutions for $P'_h(g)$ in each processor h, at generation g, they are broadcasted to all the other processors. This procedure is known as sending and reception of migrants. The receiving processors accept all the migrants, as long as their memory capacity is not exceeded.

For the sequential version the population is composed of N individuals; as the parallel version is implemented in H identical processors, the size of each population P_h will be N/H. When migrants are received, the population grows; returning to its normal level after the genetic operators (as selection) are applied.
The pseudocode is as follows:

```
Procedure NetworkDesignParallelSpea()
Begin.
    Read initial input parameters for the PSPEA: $s, g_{max}, p_c, r_m, m\%$
    Read parameters of the problem: cost and reliability of links
    Read randomly generated initial population $P$
    Generation_Counter = 1
    While StopCondition is not reached and Generation_Counter < $g_{max}$
        Compute the values of every objective function for each individual
        Receive migrants from other processes and add them to current population $P$
        Find non-dominated individuals in current population $P$
        Update external non-dominated set $P'$
        Selectively broadcast all new solutions from $P'$
        If the number of externally stored non-dominated solutions exceeds a given maximum $N'$
            Prune $P'$ by means of clustering
        End If
        Calculate the fitness of each individual in $P$ and $P'$
        Select individuals from the union set $P + P'$ until the mating pool is filled
        Apply specific crossover and mutation operators to generate the new set $P$
        Generation_Counter = Generation_Counter + 1
    End While
    Inform the organiser that process is done by sending a flag
    Send the individuals from $P'$ to the organiser
    Wait for the kill signal sent by the Organiser
End.
```

Pseudocode 4. Parallel version of SPEA. PSPEA Procedure.

The proposed method has emerged from preliminary discussions and has proved its effectiveness. Notwithstanding, subsequent experiments founded in other ideas will be held.

7 PERFORMANCE METRICS

To evaluate experimental results using the two versions, an appropriate test suit metrics is used, because no single metric can entirely capture performance, effectiveness and efficiency for multiobjective evolutionary algorithms.
Since most of these metrics reflect the likeness between the true Pareto optimal front ($\boldsymbol{Y}_{true}$) and the computed Pareto front $\boldsymbol{Y}_{known}$, a good approximation of the true Pareto optimal front is built by gathering all non-dominated individuals from all sets. In other words, for the following results, the real Pareto Optimal front is approximated by the best known solutions of all our experiments.
The test suit comprises the following metrics:

1) Overall Non-dominated Vector Generation (*ONVG*) [15]: simply counts the number of solutions in the Pareto front $\boldsymbol{Y}_{known}$

$$ONVG \overset{\Delta}{=} |\, Y_{known} \,|_c \tag{9}$$

where $|\ |_c$ denotes cardinality.

2) Overall true Non-dominated Vector Generation (*OTNVG*): counts the number of solutions in the Pareto front $\boldsymbol{Y}_{known}$ that are also in the true Pareto optimal front $\boldsymbol{Y}_{true}$.

$$OTNVG \overset{\Delta}{=} \left| \left\{ y \mid y \in Y_{known} \quad \wedge \quad y \in Y_{true} \right\} \right|_c \tag{10}$$

3) Overall Non-dominated Vector Generation Ratio (*ONVGR*) [15]:

$$ONVGR \overset{\Delta}{=} \frac{ONVG}{|\,Y_{true}\,|_c} \tag{11}$$

It denotes the ratio between the number of solutions in $\boldsymbol{Y}_{known}$ to the number of solutions in the true Pareto front $\boldsymbol{Y}_{true}$. Since the objective is to obtain a set $\boldsymbol{Y}_{known}$ as similar to the true Pareto front as it is possible, a value near to 1 is desired.

4) Error Ratio (*E*) [15]:

$$E \overset{\Delta}{=} \frac{\sum_{i=1}^{N} e_i}{ONVG} \tag{12}$$

$$e_i \begin{cases} 0 & \text{if a vector in } \mathrm{Y}_{\text{known}} \text{ is also in the true Pareto Front } \mathrm{Y}_{\text{true}} \\ 1 & \textit{otherwise} \end{cases}$$

This ratio reports the proportion of objective vectors in $\boldsymbol{Y}_{known}$ that are not members of $\boldsymbol{Y}_{true}$. Therefore an error ratio close to 1 indicates a poor correspondence between the obtained and the true Pareto front, i.e. $E = 0$ is desired.

5) Generational Distance (*G*) [15]:

$$G \overset{\Delta}{=} \frac{\left(\sum_{i=1}^{N} d_i^2\right)^{1/2}}{ONVG} \tag{13}$$

where d_i is the Euclidean distance (in objective space) between each objective vector **F** in $\boldsymbol{Y}_{known}$ and its nearest correspondent member in the true Pareto front $\boldsymbol{Y}_{true}$. A large value of *G* indicates $\boldsymbol{Y}_{known}$ is far from $\boldsymbol{Y}_{true}$ being $G = 0$ the ideal situation.

8 EXPERIMENTAL RESULTS

The results presented here were obtained from successive runs over a 10Mbps Ethernet network composed of up to 8 personal computers, each one with a AMD K6-2 350 MHz processor, with 128 MB of RAM. The program code is entirely written in C, and the parallel implementation was done using PVM (Parallel Virtual Machine) running over LINUX (Mandrake 7.0). As an example Table 2 presents the values of objective functions obtained using a 4 processors configuration, while figure 3 shows the corresponding Pareto front.

Reliability	Cost
0.68920	1433898
0.80280	1473659
0.80470	1477188
0.80720	1519613
0.80830	1523142
0.81250	1524807
0.81930	1566877
0.90800	1604727
0.90990	1605527
0.91110	1609056
0.91475	1613852
0.95960	1639359
0.96300	1641158
0.96810	1642023
0.96850	1646352
0.96980	1648017
0.97080	1657808
0.97170	1661337
0.97390	1689776
0.97470	1693106
0.97530	1694771
0.97620	1698300
0.97630	1699965
0.97640	1742035
0.97940	1776381
0.98220	1804194
0.98250	1808523
0.98370	1809323
0.98390	1810988
0.98420	1814517
0.98560	1820844
0.98600	1823508
0.98660	1851947
0.98820	1854477
0.98890	1855277
0.99050	1914106
0.99120	1942881
0.99200	2008323
0.99210	2094903
0.99260	2105892
0.99310	2167986
0.99360	2183481
0.99430	2249748
0.99540	2252568
0.99580	2389966
0.99620	2405748
0.99630	2445266
0.99650	2458872
0.99660	2631366
0.99780	2666686
0.99790	2697633
0.99830	2704293
0.99850	2848482
0.99910	2980683
0.99960	3142521
0.99980	3528468
1.00000	3873123

Table 2. Results obtained from a run with four processors.

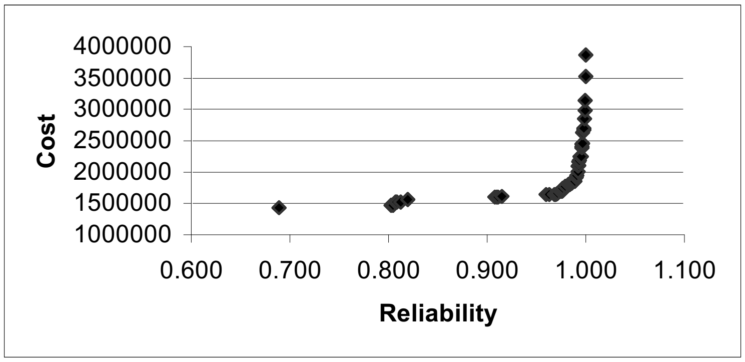

Figure 3. Pareto front obtained from a run with four processors.

Different parallel configurations were used in several runs to obtain the representative results presented in Table 3. The first column is the run number; runs 1 to 5 are executed with only one processor (sequential version); runs 6 to 10 are obtained with two processors, runs 11 to 15 were made with four processors; finally, runs 16 to 20 were implemented over eighth processors. Each run used a different randomly generated population of initial solutions. Column 2 depicts the number of processors in each run.
Column 3 presents the number of Pareto Optimal solutions found in each run denoting the metric defined as ONVG in the previous section. Column 4 contains the number of solutions found that belong to the true Pareto Optimal Front $\boldsymbol{Y}_{true}$ or OTNVG. Column 5 is the computation of ONVGR, as presented in section 7. Column 6 and 7 are total error and generational distance, respectively. Column 8 is the time –expressed in hours– each run lasted. The last column is obtained applying the concept of dominance over columns 3, 4, 5, 6, 7 and 8, i.e., it establishes a ranking among runs, based on time and other test metrics. The ranks of better solutions are lower than that of worse ones.

Run	N° of Proc.	ONVG	OTNVG	ONVGR	E	G	Time	RANK
1	1	41	0	0.7321	1.0000	0.0162	8.640	5
2	1	42	0	0.7500	1.0000	0.0291	8.712	5
3	1	46	0	0.8214	1.0000	0.0214	8.950	5
4	1	46	0	0.8214	1.0000	0.0119	8.450	4
5	1	51	0	0.9107	1.0000	0.0172	9.003	4
6	2	44	0	0.7857	1.0000	0.0109	5.210	4
7	2	44	0	0.7857	1.0000	0.0147	4.960	4
8	2	51	0	0.9107	1.0000	0.0176	5.680	4
9	2	51	0	0.9107	1.0000	0.0079	5.340	3
10	2	57	0	1.0179	1.0000	0.0076	5.020	2
11	4	45	4	0.8036	0.9111	0.0066	2.780	3
12	4	47	5	0.8393	0.8936	0.0065	2.640	2
13	4	52	0	0.9286	1.0000	0.0053	2.859	2
14	4	56	0	1.0000	1.0000	0.0092	3.050	1
15	4	57	37	1.0179	0.3509	0.0024	2.956	1
16	8	41	0	0.7321	1.0000	0.0085	1.498	2
17	8	47	10	0.8393	0.7872	0.0055	1.486	1
18	8	50	0	0.8929	1.0000	0.0046	1.560	1
19	8	58	0	1.0357	1.0000	0.0069	1.689	1
20	8	59	0	1.0536	1.0000	0.0111	1.670	1

Table 3. Results obtained with 1, 2, 4 and eighth processors.

It is interesting to realise that greater number of processors yield to a better rank, i.e. better multiobjective solutions. Moreover, table 4 presents a statistical study of ranks. The sum of ranks as well as average rank, decrease when the number of processors increases. Fourth column presents standard deviation of the sample, while fifth and sixth columns are the maximum and minimum ranks of the set respectively.

N° of Proc.	Sum of Ranks	Average	Standard Deviation	Maximum Rank value	Minimum Rank value
1	23	4.6	0.548	5	4
2	17	3.4	0.894	4	2
4	9	1.8	0.837	3	1
8	6	1.2	0.447	2	1

Table 4. Statistical study of ranks

This table clearly shows that parallelism is beneficial for the multiobjective performance of the proposed algorithm, i.e. if all objectives (test metrics plus execution time) are considered, parallel version outperforms the sequential version. Besides, the model is scalable, as eighth processors acquire better solutions than one, two or four processors.
This result is very promising not only for computer network backbone design, but for other complex engineering design problems, where the computation of objective values makes impossible the enumerative search, and other traditional methods also fail.

6 CONCLUSIONS

The contributions of this work can be summarised as follows:

- **The proposal of a new approach to the network design problem.** This problem has been addressed in several ways but has not been previously treated as the optimisation of multiple objectives. The approach is useful in the sense that it leads to a complete set of optimal solutions instead of the single solution proposed in other works. Furthermore, this proposal can be implemented in interactive design environments where the decision-maker changes parameters dynamically and narrows or widens the search space seeing immediate results.
- **Parallel version of SPEA.** We suggest and implement (for the first time in the literature) a simple and effective way of making a parallel version of a MOEA. The advantages are twofold: execution time is lowered, then the algorithm can be applied to time consuming problems; and the cost of the implementation is decreased, as our implementation runs in a relatively low in price network of personal computers and not in a expensive supercomputer. Also, as shown in the experimental results, the parallel version is capable of obtaining a broader set of solutions than the sequential one.
- We have discussed and implemented **stop criteria**, which is an innovation for MOEAs. This implementation decreases execution time and it lets the algorithm stop before arriving to the stated maximum number of generations. This also allows a study of convergence point for MOEAs, which has not previously been done.

As direction for future work, we can indicate the following:

- Raise the number of objective functions. In this regard, mean and maximum delay have been considered because both can be implemented without much computational effort and without the need of applying complex routing algorithms. Actually, we are carrying out experiments in this field. Throughput is also being examined.
- Make a statistical study to find optimal convergence point for the algorithm. In this respect, other stop criteria, like one that summarises values of test metrics, can be regarded as well.
- Choose and implement other MOEAs; build a parallel version of them and compare the obtained results. In this regard, we are presently working on a sequential and parallel versions of the Non Dominated Sorting Genetic Algorithm (NSGA) [3], applied to the problem treated in this work.
- Apply parallel versions of MOEAs to other complex engineering design problems to test the flexibility and robustness of the approach.

In conclusion, the present work demonstrates the usefulness of parallel asynchronous implementations to solve a multiobjective optimisation problem, resulting in a better performance (specially in execution time). With the newly proposed technique at hand, traditional mono-objective design of telecommunication networks may consider more (complex) objectives, as several quality of services parameters, to obtain more realistic designs in practical situations where a lot of conflicting compromises should be considered.

REFERENCES

1. Boorstyn R. and Frank H. Large-Scale network topological optimization. IEEE Trans. on communication, vol. COM-25, nº 1, pp. 29-47. 1977.
2. Colbourn C. J. “Reliability issues in telecommunications network planning”. Department of Computer Science, University of Vermont. Burlington. USA.
3. Deb K. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In Kasia Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 8. John Wiley & Sons, Chichester, U.K. 1999.
4. Deb K. Non-linear goal programming using multi-objective genetic algorithms. Technical report TR CI-60/98, University of Dortmund, Germany: Department of Computer Science/XL 1999.
5. Deeter D. L., and Smith A. B. Economic design of reliable networks, IIE Transactions, vol. 30, in print. 1999.
6. Dengiz B., Smith A. B. and Altiparmak F. Local search genetic algorithm for optimal design of reliable networks. IEEE Transactions on Evolutionary Computation, Vol. 1, N° 3, Septembre 1997.
7. Horowitz E. and Sahni S. Fundamentals of Data Structures. Reprint edition. W H Freeman & Co. June 1983.
8. Jan R. H. Design of reliable networks. Computers and operations research, vol. 20, nº 1, pp. 25-34. 1993.
9. Konak A. and Smith A. A hybrid genetic algorithm approach for backbone design of communication networks. Proceedings of the 1999 Congress on Evolutionary Computation, Washington D. C., IEEE, 1999.
10. Laufer F. Optimización topológica de redes confiables empleando A-Teams. Proyecto final. Facultad de Ciencias y Tecnología, Universidad Católica Nuestra Señora de la Asunción. Asunción-Paraguay. Febrero del 2000.
11. Morse J. N. Reducing the size of the nondominated set: pruning by clustering. Comput. Oper. Res., vol 7, n° 1 y 2. 1980
12. Pierre S. and Elgibaoui A. A tabu search approach for designing computer network topologies with unreliable components. IEEE Trans. on reliability, vol. 46, nº 3, pp. 350-359. 1997.
13. Pierre S., Hyppolite M. A., Bourjolly J. M. and Dioume O. Topological design of computer communication networks using simulated annealing. Engineering application of artificial intelligece, vol. 8, nº 1, pp. 61-69. 1995.
14. Tanembaum A. S. “Computer Networks”. Prentice-Hall, Englewood Cliffs, New Jersey 1981.
15. Van Veldhuisen D. A. *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. Ohio, EE. UU. May, 1999.
16. Zitzler E. and Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE transactions on evolutionary computation, vol 3, n° 4. November, 1999.