End-To-End Reliability-Dependent Pricing of Network Services

Bruno Tuffin*
IRISA, Rennes, France

Hector Cancela†
Departamento de Investigación Operativa, Facultad de Ingeniería,
Universidad de la República, Uruguay

Pablo Rodríguez Bocca ‡
Departamento de Investigación Operativa, Facultad de Ingeniería,
Universidad de la República, Uruguay

November 15, 2004

Abstract

This paper is a first step in the direction of charging telecommunication network access based on the reliability of end-to-end paths. Indeed, in the literature congestion pricing is usually considered as the solution to address the needs of quality-of-service-demanding new applications such as multimedia. Nevertheless, with the widespread diffusion and high capacities of optical fiber, some authors have argued that this pricing scheme will not have to be applied, as capacity will still be ahead of demand. In this paper, we introduce a new direction, noting that even when there is spare capacity, availability can still be a concern. We argue that a charging scheme depending on reliability could be interesting and illustrate how it could be implemented. We also provide a genetic algorithm aiming at extending the network so that the provider's revenue is maximized. We observe the behavior of our method over a specific network topology, and study the robustness of the solution with respect to the problem data.
Keywords: Pricing, Reliability, Simulation, Genetic algorithms.

*Bruno.Tuffin@irisa.fr
†cancela@fing.edu.uy
‡prbocca@fing.edu.uy

Fijación de tarifas de servicios de comunicaciones en redes basados en la confiabilidad de terminal a terminal

Abstract

Este trabajo es una primer etapa con vistas a la fijación de tarifas de acceso a las redes de comunicaciones en base a la confiabilidad de los caminos de punta a punta de la red (de terminal a terminal). En efecto, la fijación de tarifas basada en la congestión se considera habitualmente en la literatura como la solución para resolver las necesidades de aplicaciones exigentes en calidad de servicio, tales como en el caso de transmisiones multimedia. Sin embargo, con la capacidad importante introducida por la fibra óptica, varios autores estiman que este esquema no llegará a ser aplicado dado que la capacidad instalada excederá la demanda. En este informe, introducimos un nuevo método de fijación de tarifas teniendo en cuenta que, aunque la capacidad sea suficiente, la disponibilidad de las conexiones sigue siendo una preocupación válida. Pensamos que un método de fijación de precios basado en la confiabilidad puede ser interesante, e ilustramos como implementarlo. Damos también un algoritmo genético para extender una red existente de manera de maximizar el beneficio económico. Observamos el comportamiento del método propuesto en un caso particular, estudiante en particular la robustez de la solución respecto a los datos del problema.
Palabras clave: fijación de tarifas, confiabilidad, simulación, algoritmos genéticos.

1 Introduction

Devising a new charging scheme for telecommunication networks has become a hot topic in the scientific community. The main motivations discussed in the literature are that:

- the exponential growth of the traffic creates congestion. The current flat-rate charge used in the Internet is an incentive for overusing the resources and a usage or congestion-based scheme would be fairer.
- Also, the network has to deal with applications having different quality of service (QoS) requirements. For instance, voice and video over IP require small delays and jitter, but can support some losses, whereas e-mail or file transfers do not support losses but are not delay sensitive. Due to the congestion problem, a service differentiation has to be devised, like in IntServ [3] or Diffserv [2] architectures. A pricing scheme has to be attached to it, otherwise each customer will choose the best available service class.

Shortly, among other possibilities, a pricing scheme can be based on separating the network in totally separated sub-networks with different access charges [31], charging for service priority at each node of the network [5, 19, 30], or bidding for bandwidth [28, 36]. Note also that transfer rates can be adjusted according to the willingness to pay of the user and according to the network congestion [24, 25, 29], this area is the subject of a huge literature. We encourage to read the exhaustive surveys [7, 15, 21, 39] for more information.

In this paper, we devise a radically different pricing scheme, based on the growing idea that with the introduction of optic fiber, the backbones will be over-dimensioned, so that congestion will not occur. For this reason, it seems interesting to look at charging the network access, based on connection reliability (see for instance [12]).

We consider a network topology, where each link is assumed to have an infinite capacity (corresponding to over-dimensioning) but may fail with a given probability. Each pair of nodes has then a probability to be connected. Even if computing this probability is an NP-hard problem in its general form [32, 40], it is assumed here that its estimation is known, using efficient Monte Carlo simulation for instance [4, 16, 34]. Even if it is not the purpose of this paper, we briefly describe it in section 3.3. The price for each connection between a source s and a destination t depends on the reliability of the connection between s and t. Of course demand varies also with this price, so that the goal of the network is to set up a price that maximizes its revenue. In a second stage, the problem is to extend an already existing network in order to increase the service provider's revenue (a greedy approach would be to try to sequentially add links in order to improve the reliability of its network, and then, maybe, its revenue; this method can result in suboptimal solutions). As solving analytically this problem is intractable in general, we attempt to approach the solution with a genetic algorithm. Genetic Algorithms, proposed by Holland in 1975 [22], are an Evolutionary Computation technique. Their utility in problems of reliable

network design has already been shown [8, 9, 10, 11]. We calibrate our method and evaluate the robustness of the problem with respect to the uncertainty of some parameters on some problems, with special attention to the so-called VTHD (Very High Broadband IP/WDM test platform) network topology.

The remainder of the paper is organized as follows. In Section 2, we present the model, give all the definitions, and discuss some simple forms of the demand function. In Section 3, we examine the problem of extending the network (in terms of links) in order to increase the network revenue. Section 4 is devoted to numerical illustrations of the approach, and finally Section 5 gives our perspectives of future work and conclusions.

2 Model

We consider the network of an Internet Service Provider (ISP), represented by an undirected communication network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ (but a directed graph could also be considered) consisting of a set of nodes $\mathcal{N}$ and a set of connecting links $\mathcal{E}$. Let m be the number of links and n the number of nodes of $\mathcal{G}$. Due to the widespread use of optic fiber, we assume that each link will be over-provisioned, so that we can say that each link has an infinite capacity. This strong assumption is far from the current trend, wwhich assumes that congestion occurs often in the network and that it will be more and more present. Following our assumption, we believe that customers will be more interested in "guaranteed" connections, so that pricing based on reliability will become relevant.

We consider that for each edge $l \in \mathcal{E}$ of the network, we can choose between different technology types, which have different cost and probability of failure. It is assumed that there is a family $\mathcal{T}$ of technology types, where for each link $l \in \mathcal{E}$, $T(l) \subseteq \mathcal{T}$ is the set of all the possible technology types applicable to link l. So, for each link $l \in \mathcal{E}$ and each technology type of this link $t \in T(l)$, we assign an (independent from others) probability of failure $q_l(t)$ and a cost $c_l(t)$ (which can depend on the link length, the geography, technology amortization, operation and management associated costs, etc.). For simplicity, we assume that nodes do not have cost and that they do not fail.

Only a subset of nodes $\mathcal{K} \subseteq \mathcal{N}$ have connection demands; we call these nodes *terminals* (these nodes form the access network, and the other nodes form the backbone). To each pair of terminals (s, t) with $s, t \in \mathcal{K}$, we associate a total connection demand rate $\tilde{\lambda}_{s,t}$, a duration (assumed to be exponential with rate $\mu_{s,t}$), and a reliability $r_{s,t}$, which correspond to the probability that nodes s and t are connected (computed from algorithms in [4, 16, 34] for instance). Inter-arrivals and connection durations are assumed to be independent. To each pair (s, t) of nodes is associated a utility function, modeled by a random variable $U_{s,t}(r)$, expressed in financial terms, of getting a connection with reliability r. The overall level of satisfaction is then $U_{s,t}(r) - p$ where p is the connection price. A customer will enter the network if and only if $U_{s,t}(r) \geq p$. The random variable $U_{s,t}(r)$ is characterized by is distribution: we denote by $F_{r_{s,t}}$ its cumulative distribution function and we define $\bar{F}_{r_{s,t}} = 1 - F_{r_{s,t}}$.

Our goal is then to find out the optimal prices, for each pair (s,t), in terms of the reliability $r_{s,t}$, maximizing the network revenue

$$G(\mathcal{G}) = \sum_{(s,t)\in\mathcal{K}} n_{s,t} p_{s,t}. \tag{1}$$

over the set of prices $p_{s,t} \geq 0 \ \forall s,t$, with $n_{s,t}$ mean number of online (s,t)-connections.

The actual arrival rate of connections between s and t, $\lambda_{s,t}(p_{s,t})$, is given by

$$\lambda_{s,t}(p_{s,t}) = \tilde{\lambda}_{s,t} P(U_{s,t}(r_{s,t}) \geq p_{s,t}) = \tilde{\lambda}_{s,t} \bar{F}_{r_{s,t}}(p_{s,t}).$$

Also, according to classical queueing theory for the $M/M/\infty$ queue (see for instance [38]), we have

$$n_{s,t} = \frac{\lambda_{s,t}}{\mu_{s,t}},$$

so that

$$G(\mathcal{G}) = \sum_{(s,t)\in\mathcal{K}} \frac{\tilde{\lambda}_{s,t}}{\mu_{s,t}} p_{s,t} \bar{F}_{r_{s,t}}(p_{s,t}).$$

If we use first order conditions while maximizing this revenue (as the price $p_{s,t}$ is necessarily positive otherwise the revenue between s and t would be zero, meaning that the Langrange multiplier is zero), i.e., $\partial G/\partial p_{s,t} = 0 \ \forall s,t$, (assuming that it gives the solution) we get

$$\begin{aligned} \frac{\partial}{\partial p_{s,t}}(p_{s,t}\bar{F}_{r_{s,t}}(p_{s,t})) &= 0, \text{ i.e.,} \\ \bar{F}_{r_{s,t}}(p_{s,t}) + p_{s,t}\frac{\partial \bar{F}_{r_{s,t}}(p_{s,t})}{\partial p_{s,t}} &= 0. \end{aligned} \tag{2}$$

2.1 Demand Function

2.1.1 Two general examples

In a general way, solving Equation (2) can easily be carried out numerically, using Newton's method for instance. In the following, we give some analytical results for some particular cases of demand function.

Example 1 : assume that, $\forall s,t$,

$$F_{r_{s,t}}(p_{s,t}) = 1 - e^{-\alpha(r_{s,t})p_{s,t}}$$

with α given positive function. Following Equation (2), we have

$$e^{-\alpha(r_{s,t})p_{s,t}}(1 - p_{s,t}\alpha(r_{s,t})) = 0,$$

which gives

$$p_{s,t} = \frac{1}{\alpha(r_{s,t})}.$$

It is straightforward to check that it provides a maximum. ■

Example 2 : assume now that, $\forall s,t$,

$$F_{r_{s,t}}(p_{s,t}) = \left(\frac{p_{s,t}}{M_{s,t}}\right)^{\alpha(r_{s,t})+1}$$

with $0 \leq p_{s,t} \leq M_{s,t}$. Equation (2) becomes

$$1 - \left(\frac{p_{s,t}}{M_{s,t}}\right)^{\alpha(r_{s,t})+1} - p_{s,t}^{\alpha(r_{s,t})+1} \frac{\alpha(r_{s,t})+1}{M_{s,t}^{\alpha(r_{s,t})+1}} = 0,$$

giving

$$p_{s,t} = M_{s,t}(\alpha(r_{s,t})+2)^{-1/(\alpha(r_{s,t})+1)}.$$

Here too, it can be easily verified that it provides a maximum. ■

2.1.2 Utility linear in the reliability

In many economic applications, the utility is linear in its argument (here the reliability) so that

$$U_{s,t}(r) = U_{s,t} + \gamma_{s,t} r$$

with $\gamma_{s,t}$ translating the reliability in financial terms, as the monetary value of a reliability unit (so that the utility increases with r) and $U_{s,t}$ random variable not depending on r. Let $F^*_{s,t}$ be the distribution function of random variable $U_{s,t}$ (but not depending on r this time). Equation (2) becomes

$$\bar{F^*}_{s,t}(p_{s,t} - \gamma_{s,t} r) + p_{s,t} \frac{\partial \bar{F^*}_{s,t}(p_{s,t} - \gamma_{s,t} r)}{\partial p_{s,t}} = 0.$$

Of course, $p_{s,t} \geq \gamma_{s,t} r_{s,t}$ since the utility is supposed to be positive (if $p_{s,t} < \gamma_{s,t} r_{s,t}$, increasing it to $\gamma_{s,t} r_{s,t}$ would not reduce the demand).

Example 1 bis: let, $\forall s,t$, $F^*_{r_{s,t}}(p) = 1 - e^{-\alpha_{s,t} p}$.

If we assume that $p_{s,t} - \gamma_{s,t} r > 0$, the F.O.C. gives

$$e^{-\alpha_{s,t} p_{s,t}}(1 - p_{s,t}\alpha_{s,t}) = 0,$$

which gives

$$p_{s,t} = \frac{1}{\alpha_{s,t}}.$$

Due the form of the derivative, it provides the maximum. ■

Example 2 bis: following the Example 2, $\forall s,t$,

$$F^*_{s,t}(p) = \left(\frac{p}{M_{s,t}}\right)^{\alpha_{s,t}+1}, \tag{3}$$

with $0 \leq p \leq M_{s,t}$. the F.O.C. gives

$$1 - \left(\frac{p_{s,t} - \gamma_{s,t} r_{s,t}}{M_{s,t}}\right)^{\alpha_{s,t}+1} - (p_{s,t} - \gamma_{s,t} r_{s,t})^{\alpha_{s,t}+1} \frac{\alpha_{s,t}+1}{M_{s,t}^{\alpha_{s,t}+1}} = 0,$$

giving

$$p_{s,t} = \gamma_{s,t} r_{s,t} + M_{s,t}(\alpha_{s,t} + 2)^{-1/(\alpha_{s,t}+1)}. \quad (4)$$

which provides the maximum for the same reason than in the previous example. ■

In the rest of this paper, we work with demand function (3).

3 Extending the Network, based on Requests

The next step is to wonder whether and how the network can plan its topology. The idea is the following : consider a family $\mathcal{F}$ of graphs such that $\forall \mathcal{G} = (\mathcal{N}, \mathcal{E}') \in \mathcal{F}$, the set $\mathcal{N}$ of nodes is the same, but the set of links $\mathcal{E}'$ is a subset of possible links $\mathcal{E}$ ($\mathcal{E}' \subseteq \mathcal{E}$).

Remember that for each link $l \in \mathcal{E}$, we can choose between different technology types $t \in T(l)$ (where $T(l) \subseteq \mathcal{T}$), which have different operating cost $c_l(t)$ and probability of failure $q_l(t)$. In order to completely define a network $\mathcal{G} = (\mathcal{N}, \mathcal{E}')$ in our model, we have to choose a technology type for each link $l \in \mathcal{E}'$, we express this with the assignment function $a : \mathcal{E}' \to \mathcal{T}$ (where $a(l)$ means the technology type chosen for the l link, $a(l) \in T(l)$).

From the network point of view, the goal is to determine the topology $\mathcal{G} = (\mathcal{N}, \mathcal{E}')$ (and the assignment function a) maximizing the benefits

$$G(\mathcal{G}) - \sum_{l \in \mathcal{E}'} c_l(a(l)).$$

3.1 Problem Definition

Now, we can summarize the formal problem and the notations used throughout the rest of the paper.

Inserting in revenue equation (1) the price (4) and distribution function (3) ($\bar{F}_{r_{s,t}}(p_{s,t}) = 1 - F^*_{s,t}(p_{s,t} - \gamma_{s,t} r_{s,t})$) we have

$$G(\mathcal{G}) = \sum_{(s,t) \in \mathcal{K}} \frac{\tilde{\lambda}_{s,t}}{\mu_{s,t}} [\gamma_{s,t} r_{s,t} + M_{s,t}(\alpha_{s,t} + 2)^{-1/(\alpha_{s,t}+1)}] \frac{(\alpha_{s,t} + 1)}{(\alpha_{s,t} + 2)}.$$

We then have the problem definition

$$\text{Maximize} \sum_{(s,t) \in \mathcal{K}} \frac{\tilde{\lambda}_{s,t}}{\mu_{s,t}} [\gamma_{s,t} r_{s,t} + M_{s,t}(\alpha_{s,t} + 2)^{-1/(\alpha_{s,t}+1)}] \frac{(\alpha_{s,t} + 1)}{(\alpha_{s,t} + 2)} - \sum_{l \in \mathcal{E}'} c_l(a(l)) \quad (5)$$

where $\forall s, t \in \mathcal{K}$ and $\forall l \in \mathcal{E}$,

- $\tilde{\lambda}_{s,t}$ total connection demand rate
- $\mu_{s,t}$ duration rate
- $r_{s,t}$ reliability in $\mathcal{G} = (\mathcal{N}, \mathcal{E}')$

- $\gamma_{s,t}$ utility of reliability term
- $c_l(t)$ cost of technology type t of link l, where $t \in T(l)$
- $\alpha_{s,t}, M_{s,t}$ constants of the demand function
- $a : \mathcal{E}' \to \mathcal{T}$ assignment function, with $a(l)$ is the technology type chosen for link l.

Solving this problem is intractable in general. Indeed, finding the optimal topology requires the computation of the reliability of each end-to-end connection, this problem being itself NP-hard. As an attempt to approach the solution, simulated annealing could be used (see for instance [13, 23, 6]). In this work we use an evolutionary computation method to estimate the solution.

3.2 Evolutionary Computation

The Evolutionary Computation methods are inspired by nature's capability to evolve, where each individual wants to be adapted in the best way to its environment. The Evolutionary Computation algorithms work with a set *-population-* of solutions *-individuals-*. In each iteration of the algorithm the individuals are selected and mutated, trying to improve their fitness to improve their fitness; in a biological context, this process has been interpreted as being guided by the goals of survival and preservation of the species.

There is a variety of different Evolutionary Computation algorithms, we can classify them into three categories: Evolutionary Programming (proposed by Fogel et al. in 1966 [17, 27]), Evolutionary Strategies (proposed by Renchenberg in 1973 [33]) and Genetic Algorithms (developed by Holland in 1975 [22]).

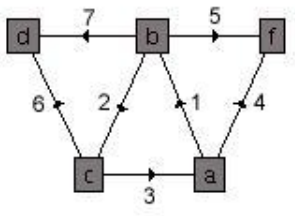
To approach the solution of the problem of extending the network we use a genetic algoritm. Several references [8, 9, 10, 11], to the application of genetic algorithms in problems of reliable network design justify our selection.

3.2.1 Genetic algorithm

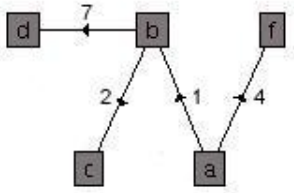
In every iteration of the genetic algorithm, a number of operators is applied to the individuals of the current population to generate the individuals of the next generation. The basic operations are *crossover* (where two or more individuals are recombined) and *mutation* (where an individual is slightly modified). A *selection* of the best individuals, based on their fitness, takes place between two consecutive generations. An individual with higher fitness has a higher probability to be chosen as a member of the next generation. In order to improve the fitness of the population at each generation, it is important to preserve the diversity of individuals, so the selection can not be very elitist.

Below is the skeleton of a simple genetic algorithm described by Goldberg and applied in this paper (see [18] and [41] for details).

```
initialize parameters
pop = generate initial population
loop
selectedPop = select(pop)
crossPop = crossover(selectedPop)
pop = mutate(crossPop)
repeat until a good solution is found or a max number of generations was reached
```



(a) Encoding example $\|\mathcal{T}\| = (2, 2, 2, 1, 3, 1, 3)$.



(b) Encoding instance $X = (1, 1, 0, 1, 0, 0, 3)$.

Figure 1: Encoding example.

In order to define our genetic algorithm, the following points have to be specified: encoding, fitness function, initial population, stopping criterion, and operators (selection, crossover and mutation).

3.2.2 Encoding and Fitness Function

Generally, in genetic algorithms, an individual (i.e. a solution) is expressed by a fixed size string of an alphabet, called *genotype* or *chromosome*; and the genotypes are formed of *alleles* (characters of the alphabet). Therefore, we need a function in order to encode all the possible solutions in strings (and the inverse function in order to decode). In our problem, a feasible solution is a graph where some links, and their technology type, are selected from all possible links. Our genotype (solution encoded) is an array of size given by the amount of edges, where we have an allele for each possible link that can be in the network. The alphabet of each allele is an integer between zero and the maximum number of technology types of this link, where zero means that this link does not appear in the solution, and other value that we use the technology type with this value to this link.

Figure 1 shows an encoding example, where the family $\mathcal{F}$ of possible graphs is represented in Figure 1(a) (where $\|\mathcal{T}\|$ is the maximum number of technology types for each link, i.e. a vector of $|T(l)|\ \forall l \in \mathcal{E}$) and an instance X in Figure 1(b); it is possible to see, for example, that the technology type 1 is chosen for link 1, link 3 does not appear in the solution and the technology type 3 is chosen for link 7.

In each generation, the genetic algorithm tries to maximize the fitness function of the population (due to the selection operation), while our problem is to maximize the objective function (the benefits of the network, given by Equation 5); therefore, it seems natural to take the objective function as the fitness function, but we have to look at some restrictions and considerations when choosing it.

The first consideration is that the fitness function can not be negative (due to the selection operation), but the objective function can be negative if we choose a very expensive network. Then, in order to have a positive fitness function, we add to the objective function the maximum cost of the network (the sum of most expensive link costs).

$$\sum_{(s,t)\in\mathcal{K}} \frac{\tilde{\lambda}_{s,t}}{\mu_{s,t}}[\gamma_{s,t}r_{s,t}+M_{s,t}(\alpha_{s,t}+2)^{-1/(\alpha_{s,t}+1)}]\frac{(\alpha_{s,t}+1)}{(\alpha_{s,t}+2)}+\sum_{l\in\mathcal{E}}\max_{t\in T(l)} c_l(t)-\sum_{l\in\mathcal{E}'} c_l(a(l)). \tag{6}$$

Another consideration must be made: when the problem has constraints, they are usually put inside the fitness function, like a penalization. Our problem specification does not include constraints, so this technique is not applied here.

Finally, Goldberg [18] has introduced two problems related to utilization of fitness function during the selection: the "indifference to the diversity" and the "premature convergence", the solution adopted in this work to solve both problems, is a linear scaling of the fitness function (see [18] for details).

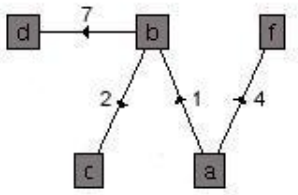
3.2.3 Initial Population and Stopping Criterion

The first generation, called "Initial Population" plays an important role in the performance of the genetic algorithm. It is better to start with good solutions in order to test the possible best individuals and to have an algorithm converging quickly, but the Initial Population should better have enough diversity to explore all the search space. In the simple genetic algorithm, described by Goldberg, the Initial Population is chosen uniformly at random. In our work a genotype of the Initial Population is chosen as follows: if a link has a zero cost technology type, we choose uniformly between possible technology types; if it has not a zero cost technology type, first we decide wether to add or not this link by sampling a parametric Bernoulli random variable (the parameter is called initial rate, and fixed at 0.8), second if the link is added, we chose uniformly between possible technology types. In the problem of network extension there are pre-installed links (then with zero cost) and we want to evaluate the benefit of choosing another technology type for these links. In addition, if we have a link with zero cost, it is always better to include it than to omit it. These reasons have motivated this initial population method. In Section 4.2, we test other variants for the initial population method, and evaluate their performance.

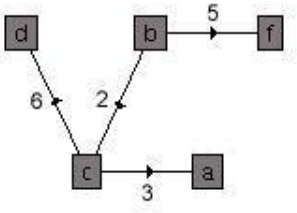
The simplest Stopping Criterion which can be used in a genetic algorithm is to fix the number of generations; this strategy is used by most applications. Another possibility is to continue iterating on new generations while there is an improvement in the solution. In this work, we have implemented the two versions and chosen one in a calibration stage (discussed in Section 4.1.1).

3.2.4 Selection, Crossover and Mutation

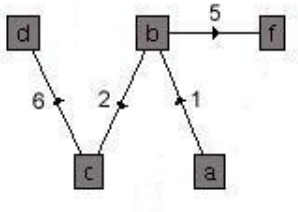
As previously said, selection is a process in which individuals are copied (or die) in the next generation, according to their fitness function values f. The



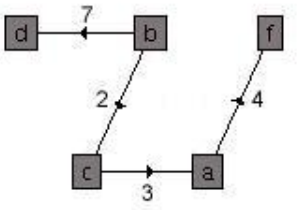
(a) Parent encoding instance $X = (1, 1, 0, 1, 0, 0, 3)$.



(b) Parent encoding instance $Y = (0, 1, 1, 0, 1, 2, 0)$.



(c) Child encoding instance $W = (1, 1, 0, 0, 1, 2, 0)$.

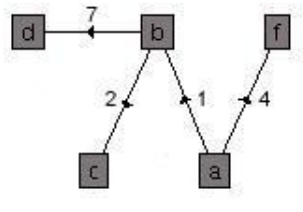


(d) Child encoding instance $Z = (0, 1, 1, 1, 0, 0, 3)$.

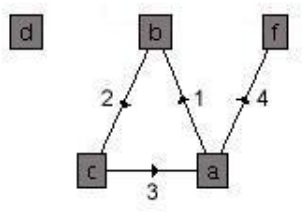
Figure 2: Crossover example.

Selection operator may be implemented in numerous ways; in this work we use the "roulette wheel" selection, where, for each individual, the probability of it being in the next generation is proportional to the population total fitness $\frac{f_i}{\sum f}$. In addition, the *Selection* includes a simple elitism, where the best genotype of the present population is selected directly.
Crossover and Mutation are also implemented in a simple way. In the single point *Crossover* operation, two strings -the parents- of the current population are chosen at random, and an integer position i along the strings is selected uniformly at random between 1 and the string length less one. Two new strings -the children- are created, in the new generation, by swapping all alleles of the parents between positions $i + 1$ and length of string. Crossover is applied to two randomly selected strings with a probability p_c (if this does not happen, the parents are copied exactly to the next generation).

For example, if we cross $X = (1, 1, 0, 1, 0, 0, 3)$ and $Y = (0, 1, 1, 0, 1, 2, 0)$ (shown in Figures 2(a) and 2(b)) in position 3, we have the two children $W = (1, 1, 0, 0, 1, 2, 0)$ and $Z = (0, 1, 1, 1, 0, 0, 3)$ (shown in Figures 2(c) and 2(d)).



(a) Parent encoding instance $X = (1, 1, 0, 1, 0, 0, 3)$.



(b) Child encoding instance $X' = (1, 1, 2, 1, 0, 0, 0)$.

Figure 3: Mutation example.

$X = (1, 1, 0$	$1, 0, 0, 3)$
$Y = (0, 1, 1$	$0, 1, 2, 0)$
$W = (1, 1, 0$	$0, 1, 2, 0)$
$Z = (0, 1, 1$	$1, 0, 0, 3)$

The *Mutation* operator explores new solutions, introducing changes in the population to diversify the search. A genotype mutation is a random alteration of all its alleles with mutually independent probabilities p_m. All alleles of all genotypes of the population can be mutated. In the simple mutation operator, described by Goldberg, the alleles are mutated uniformly at random, that means that there is not difference in treatment between zero (the link is not present in the string) and other values (the link is present in the string). Taking into account our encoding schema, this introduces a bias towards the presence of links in the solution; an alternative is to consider other mutation operators where the zero values are treated in a different way than the rest. In Section 4.1.2, we test one such variant of mutation operation, and we show that it has worse performance than the standard simple mutation.

Continuing with the example, suppose that we mutate $X = (1, 1, 0, 1, 0, 0, 3)$ in the positions 3 and 7, a possible new solution is $X' = (1, 1, 2, 1, 0, 0, 0)$ (shown in Figure 3). In this new solution, there is not link to node d, perhaps the demand in-out of this node does not justify the costs of its connection.

An important remark has to be pointed out: the problem has very simple constraints, and our operations preserve the feasibility of the solutions. This is a nice feature of the problem, as feasibility is especially hard to maintain for graph problems because the operations tend to disrupt the GA solution encoding.

3.3 Computation of Network Reliability

The main assumption of this work is that the pricing scheme is based on connection reliability; our genetic algorithm needs to compute the network reliability

many times, therefore, we need to compute it efficienctly. The problem of calculating the reliability of each end-to-end connection is NP-hard, and it is in itself one active area of research. There are three main approaches: exact calculation, upper and lower bound expressions and efficient simulation. In our work, the network reliability is estimated with crude Monte Carlo simulation (see for instance [35, 20]) and with the Generalized Antithetic Variable method proposed in [14]. The Generalized Antithetic Variable algorithm performs better in computational time and in precision than the crude Monte Carlo technique, particularly in the case of highly reliable systems, therefore our final method uses this estimation.

3.3.1 Basics

In our model, we consider the network as a graph, $\mathcal{G} = (\mathcal{N}, \mathcal{E}')$. In this section we suppose that the technology type for each link has been chosen previously (this is the case, as before evaluating each topology we decide the links and their type), and (without loss of generality), we will not explicitly denote it in our notation. In this case, we can define a binary random variable x_l for each link l, called the state of the link: $x_l = 0$ means that the link l is down (failed) and $x_l = 1$ means that the link works perfectly. The probability of failure is $q_l = Pr(x_l = 0)$. The state of the network is completely characterized by the vector X whose components are the x_l's. The 2-terminal reliability measure of the network is needed in our model, from a general point of view, we can fix two nodes s and t and formalize this measure as a binary function Φ, called the structure function. $\Phi(X) = 1$ if and only if s and t are connected in the graph defined by X. Finally, we will denote by $r_{s,t}$ the 2-terminal reliability measure between s and t, and that is $r_{s,t} = Pr(\Phi(X) = 1) = E(\Phi(X))$.

The Simple Monte Carlo technique consists of generating N independent state samples $X^1, ..., X^N$ and evaluate the reliability $r_{s,t}$ by the unbiased estimator

$$\widehat{r_{s,t}} = \frac{1}{N} \sum_{n=1}^{N} \Phi(X^n).$$

In order to estimate the variance of this estimator we use the unbiased estimator

$$\widehat{v} = \frac{\widehat{r_{s,t}}(1 - \widehat{r_{s,t}})}{N - 1}.$$

3.3.2 Generalized Antithetic Variable method

Instead of generating N independent samples (like standard Monte Carlo), the Generalized Antithetic Variable method generates B independent blocks of L samples each one (notation: $X^{(b,1)}, ..., X^{(b,L)}$ samples of block b). The L samples of a block are chosen in a dependent way that decreases the global variance (respect to the standard Monte Carlo). In order to estimate the reliability $r_{s,t}$

we use the unbiased estimator

$$\widehat{r_{s,t}}' = \frac{1}{B}\sum_{b=1}^{B}\left(\frac{1}{L}\sum_{l=1}^{L}\Phi(X^{(b,l)})\right).$$

To estimate the variance $VAR(r_{s,t})$ we use the unbiased estimator

$$\widehat{v}' = \frac{1}{B(B-1)}\sum_{b=1}^{B}\left(\frac{1}{L}\sum_{l=1}^{L}\Phi(X^{(b,l)})\right)^2 - \frac{1}{B-1}\widehat{r_{s,t}}'^2.$$

In order to apply our genetic algorithm, we need the 2-terminal reliability between any pair of terminals $s, t \in \mathcal{K}$. We use the same samples to estimate all needed reliabilities $r_{s,t} : \forall s, t \in \mathcal{K}$ at the same time. This introduces a covariance between reliability estimators. Suppose that the structural function of the 2-terminal reliability between s' and t' is Φ', therefore, we estimate the covariance $COV(r_{s,t}, r_{s',t'})$ with the unbiased estimator

$$\widehat{c}' = \frac{1}{B(B-1)}\sum_{b=1}^{B}\left(\frac{1}{L^2}\sum_{l=1}^{L}\Phi(X^{(b,l)})\sum_{l=1}^{L}\Phi'(X^{(b,l)})\right) - \frac{1}{B}\widehat{r_{s,t}}'\widehat{r_{s',t'}}'.$$

This method has been introduced as a generalization of the dagger sampling algorithm [26]. In dagger sampling, the simulations were always made with the maximum possible value of L, called L_d. If we suppose that $\forall l \in \mathcal{E}, q_l \geq 0.5$ then L_d is the least common multiple of the $\lfloor \frac{1}{q_l} \rfloor$'s. For example, if we work with failure probabilities: 0.1, 0.01 and 0.0001 (like our following validation example), the maximum L value is $L_d = 10.000$.

4 Numerical Illustrations

In this section, we evaluate the performance of the proposed algorithm by means of various examples. The performance is taken in the sense of effectiveness and efficiency, where effectiveness means to be close to the optimal solution, and efficiency means computational effort.

4.1 Experimental Design

In our simulations design, the outcomes of an experiment are called the *response variables*, the factors that affect the response variables are called *parameters* and the possible values of that parameters are the *levels*. The experimentation consists in two main steps: calibration and validation. The goal of calibration is to find "best levels" for the parameters, and the goal of validation is to study the performance of the method with more detail. The calibration is made using random graph topology problems, and the validation with the experimental VTHD network [1].

The calibration runs were executed on a SunFire 280R, with two 1.2 GHz UltraSPARC III Cu processors, 2 GB of main memory, and SolarisTM 8 operating system. The validation experiments were computed on a Sun SPARCcenter-2000, with sixteen processors, 4 GB of main memory, and SolarisTM 8 operating system.

4.1.1 Random graph topologies

In the calibration stage we work with 10 randomly chosen problems (shown in Appendix 6). For each problem, we evaluate the following parameters: mutation rate p_m, crossover rate p_c, population size P, generation number G (and "stopping criterion"), Monte-Carlo block size B and value L. The possible levels for these parameters are shown in Table 1.

Parameter	Levels		
B	50	100	200
L	3	50	100
P	50	100	150
p_c	0.75	0.85	0.95
p_m	0.001	0.005	0.01
G	≤ 200		

Table 1: Parameter levels in the calibration stage.

In each simulation only one parameter varies, the default initial level for each parameter are shown in Table 2.

Parameter	Default Level
B	100
L	50
P	100
p_c	0.85
p_m	0.003
G	50

Table 2: Default parameter levels in the calibration stage.

There are other parameters that could be considered, for example, initial rate (fixed at 0.8) and scaled fitness factor (fixed at 2), but, intending to decrease the number of simulations, we did not consider them.

For the parameters p_m, p_c, P, and G, the response variables are: optimal value, computational time and convergence rate. The Monte-Carlo parameters (B and L) are calibrated in a different way: we only consider a trade-off between computational time and the estimation uncertainty.

The first simulations are used to calibrate Monte-Carlo parameters. Increasing the parameters B and L results in a better reliability estimation (at the cost of a higher computational time). Also, when the reliability is either very high or very low (like can happen in bigger problems), it is necessary to use large

values for parameters B and L. On the other way, the optimum L parameter value depend directly of all failure probabilities $q_l(t)$. In [14] it is shown that in highly reliable systems, the use of high values of L does not give us an important variance reduction (and it takes more computational time). We have evaluated the variance reduction on problems with different sizes and different reliability performances. Our conclusion is that if we work with problems involving approximately 30 nodes and 40 links, and probability of failure greater than 0.01, an acceptable trade-off is B=100 and L=50. This is extensively discussed in Subsection 4.3 (Table 8 displays the average time of reliability computation).

With the others parameters we evaluate each level for each calibration problem, and select the one with the best average behavior (in the sense of optimal results) and which does not need a "too large" computational time. We work as follows: we run the 10 calibration problems, varying only one level of each parameter respect to the default initial level shown in Table 2. For each level, we summarize in Table 3 the 10 solutions in an average relative error, an average execution time and an average best iteration (iteration where the best solution is found). The error is the relative difference between the solution and the best known solution shown in Table 9.

Parameter	Level	Average Relative Error (%)	Average Time (min.)	Average Best Iteration
P	50	2.67	28.30	42
	100	0.79	46.58	38
	150	0.75	71.96	40
p_c	0.75	1.28	80.92	39
	0.85	0.79	74.94	38
	0.95	0.57	83.68	44
p_m	0.001	1.69	28.74	40
	0.005	0.94	33.30	36
	0.010	0.51	34.54	36
G	50	0.79	46.58	38
	200	0.49	87.39	125
	Quality	1.64	21.27	28

Table 3: Average solutions for each level. Result of run the 10 calibration problems, varying only one level of each parameter, respect to the default initial level. For each level, the 10 solutions are summarize in an average relative error, an average time and an average best iteration. Time is in minutes, and the Error is relative difference between the solution and the best known solution.

It is possible to see that time increases proportionally with the population size P , but that the gain in relative error when P goes from 100 to 150 does not justify the increased time. Therefore, we choose $P = 100$. As we expect, there is a little difference in execution time between the different levels of p_c and p_m. For these parameters we choose that that has less relative error, i.e. $p_c = 0.95$ and $p_m = 0.01$. In general is not advisable to use a very high mutation rate, because, in this situation, noise is added to the solutions and the algorithm can not converge to good solutions. We think that $p_m = 0.01$ is a relatively high

value, which has nonetheless shown good performance for our test cases (see the convergence in Figure 6).

In the simple genetic algorithm, the number of generations is fixed, this is the simple "Stopping Criterion" used in most applications, which we will call the "Quantity Stopping Criterion". Another possibility is iterating on new generations while there is an improvement in the solution, this we call the "Quality Stopping Criterion". Using more generations means the algorithm has more time to find good solutions. But the computational time grows linearly with the number of generations, so that this is an important limitation. The problem with the "Quality Stopping Criterion" we used is that, for some instance problems, it finishes prematurely. To avoid this problem, we use the "Quantity Stopping Criterion", and we choose a generation value $G = 100$.

Final calibrated level for each parameter are summarized in Table 4.

Parameter	Calibrated level
B	100
L	50
P	100
p_c	0.95
p_m	0.01
G	100

Table 4: Final parameter levels in calibration stage.

4.1.2 VTHD: Very High Broadband IP/WDM test platform

The VTHD (Very High Broadband IP/WDM test platform) network is a French project, whose main goal is to investigate the applications of a new generation of Internet and Intranet networks [1]. In our context it is a pictorial simple application of our method. The VTHD network uses two main technologies types for its links: the backbone part of the network uses a IP/WDM architecture, with STM1/4 and STM16 links (in this work we suppose a probability of failure of 0.01 for this links); the access part of the VTHD network uses Giga-Ethernet links (with a probability of failure of 0.1).

To the actual network (shown in Figure 4(a)), we add some possible links to the backbone (dotted lines in Figure 4(b)), and simulate the genetic algorithm in different situations. Especially, we use our method in three different scenes. The three problems have the same specification (the same parameters of the demand, utility, etc.), but they differ in the possibilities of network extension. In the first problem (called VTHD1), we evaluate the benefits of extending the backbone of the network with five strategic links (dotted lines in Figure 4(b)), the best solution of this problem is easily known because the network extension has only 32 possibilities. The second and third problems (called VTHD2 and VTHD3 respectively) add in addition the possibility of upgrading the access network with IP/WDM links. The difference between these two problems is the cost of the new possible access links (in VTHD2 problem we use reasonable

costs, and in VTHD3 problem we consider very high costs for these upgrades). It is very hard to evaluate the optimal solution for these two problems because they have too many possibilities (exactly 2^{25} possibilities). The full specification of the three problems is given in Appendix 7.

The performance metrics (response variables) in this stage are: computational time, convergence rate and obviously optimal approach.

The difference in execution time between the three problems is not significant. Each mutation takes in average only 2.72 milliseconds, and each crossover takes 0.10 milliseconds. The mutation and the crossover are often executed in the execution of a genetic algorithm (exactly 10000 times the mutation and 5000 times the crossover, because we have a population of size 100, and 100 generations of evolution). The selection operator needs the fitness of the population to be computed, therefore, before each selection, we have to calculate the fitness of the new individuals, that implies a reliability estimation. A reliability estimation takes in average 12,86 seconds, the consequence is that the algorithm execution time is approximately 36 hours [1]. The execution time can be drastically reduced if we change the reliability evaluation method (this is discussed in Section 5).

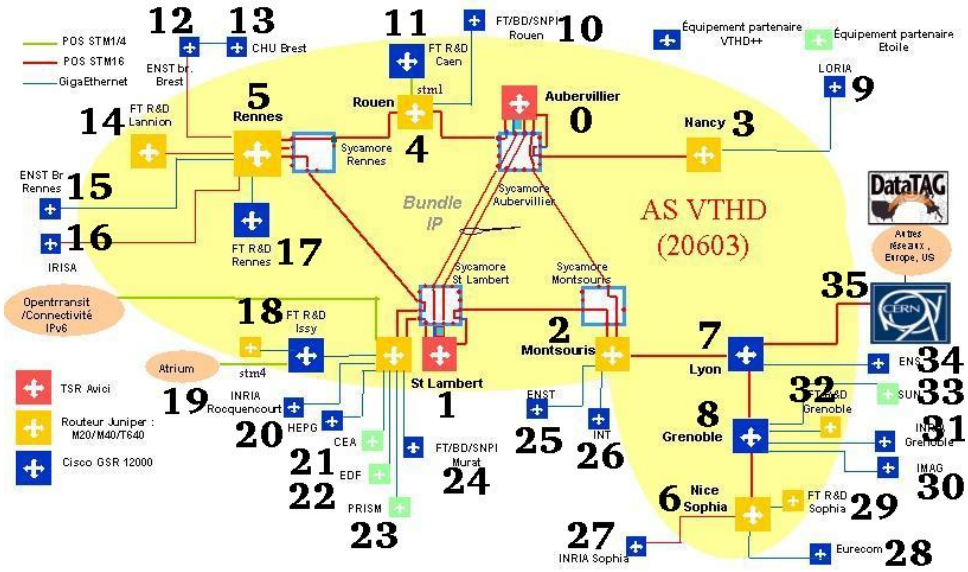
For the VTHD1 problem, the best genotype is found in the 25th generation, and its benefit is 1483.75. This genotype corresponds to the exact solution of the problem, which was known in advance.

For the VTHD2 and VTHD3 problems, we have a bound of the best benefit, because the exact solution of VTHD1 is applicable to these problems. In the VTHD2 problem, the solution found is similar to the VTHD1 bound, with the same backbone and where some links of the access network upgraded with IP/WDM technology type; exactly 13 of these links are upgraded. The benefit is 1533.86, with an overall cost of 130. We know a better solution for the VTHD2 problem (which was obtained in one of a set of exhaustive tests), this best known solution has a benefit of 1543.45, with a cost of 100.00.

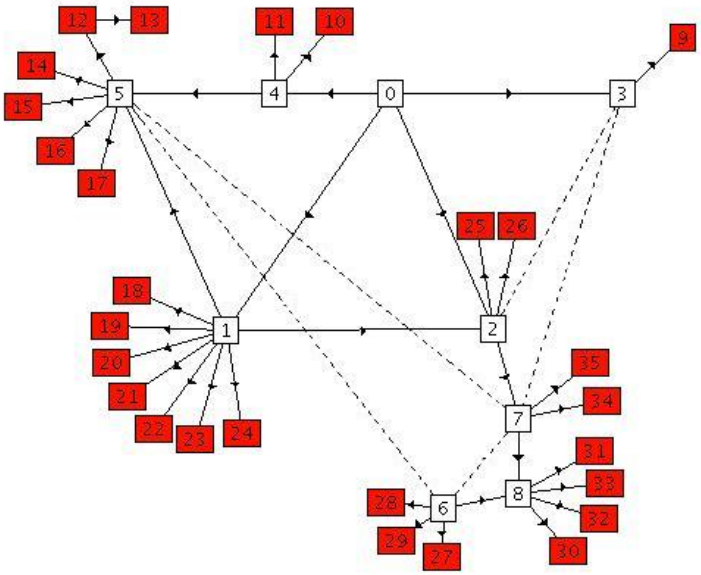
The genetic algorithm has a good performance in the VTHD3 problem, obtaining a solution with a benefit of 1483.75. This solution is the the best known solution for this problem, and corresponds to the VTHD1 optimum (remember that we used very high link costs in this problem, this means that we expect there will be no upgrade in the access network). Table 5 summarizes the principal results, and Figure 5 shows the solution found for the VTHD1, VTHD2 and VTHD3 problems.

The convergence of the population fitness throughout the execution of a genetic algorithm is very important, because it can not be very slow (to avoid the need of a long execution and the indifference to the diversity) and it cannot be very fast (to avoid a premature convergence and to preserve a diversity of individuals). In Figure 6 we show the evolution of the average fitness and best fitness of the population, and we consider that the selected parameters in calibration give us a good balance in the convergence.

[1] Remember that the validation is made in a Sun SPARCcenter-2000, if we execute these problems in a SunFire 280R the execution time would be approximately 4 hours.

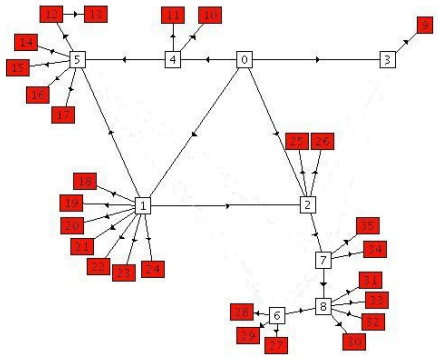


(a) VTHD Network, translation between node ids and places.

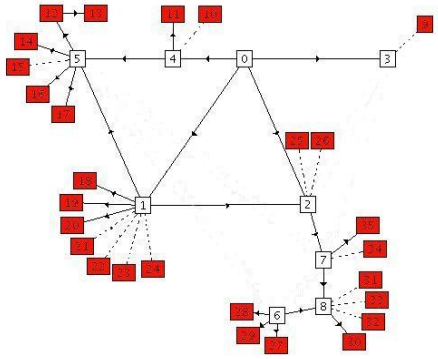


(b) VTHD Network, model with possible extended links(dotted lines). Red (dark) nodes are terminals

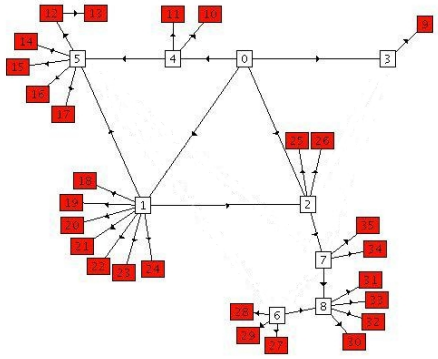
Figure 4: Validation Problem: Very High Broadband IP/WDM test platform [VTHD].



(a) VTHD1 Solution.

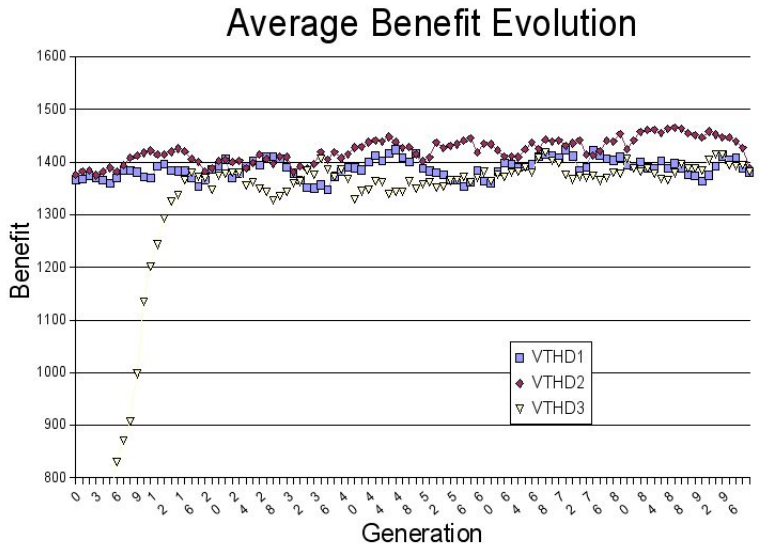


(b) VTHD2 Solution.

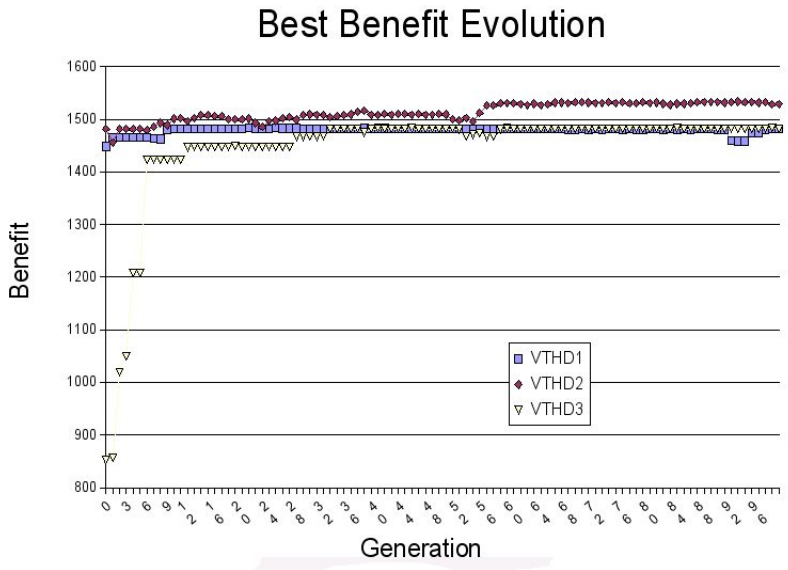


(c) VTHD3 Solution.

Figure 5: Validation Problem Solutions. Dotted lines represent the access links that upgrade the technology type in the solution.



(a) Average Benefit Convergence in Evolution.



(b) Best Benefit Convergence in Evolution.

Figure 6: Benefit Convergence in Evolution. In each generation, we display the average benefit of the generation (a), and the best benefit of the generation (b).

Problem	Best Known Revenue	Best Fitness	Maximum Cost	Cost	Benefit
VTHD1	1483.75	1643.75	160.00	0.00	**1483.75**
VTHD2	1543.45	1903.86	370.00	130.00	**1533.86**
VTHD3	1483.75	3743.75	2260.00	0.00	**1483.75**

Table 5: VTHD Problem Solutions. The maximum cost of the network is the sum, for all link, of most expensive technology type costs. The Benefit is our objetive function (5).

4.2 Other alternatives for generating the Initial Population

The Initial Population plays an important role in the performance of the genetic algorithm, and particularly in its convergence. It is better to start with good solutions in order to test the possible best individuals and to have a quickly convergent algorithm, but the Initial Population needs also sufficient diversity to explore all the search space. A simple trade-off is to start with good solutions and add a random noise to each one.

As described in Section 3.2.3, we devise an Initial Population construction method (named P2) which consists in building individuals by considering in turn each link. If this link has a technology type of zero cost, we choose uniformly between possible technology types; if it has not a zero cost technology type, first we decide to add or not this link by sampling a parametric Bernoulli random variable (the parameter is called initial rate, and fixed at 0.8), second if the link is added, we choose uniformly between possible technology types. The resulting individuals in the population will always include the links which can be used at zero cost (but maybe with a different technology type).

We have tested other Initial Population methods. The simple random method (denoted P0) consists in choosing uniformly at random for each link its allele value, independently from any other information; then, individuals might not include links which have zero cost. The last method (denoted P1) is based in the assumption that it is always better to use the installed (zero cost) links, because they will give better reliability and therefore better revenue. Therefore, to generate the individuals in the Initial Population we first include all zero-cost links, and we choose uniformly at random the allele values for the rest of the links.

Table 6 shows the tested Initial Population methods, the average population fitness, its standard deviation, and two diversity metrics. Both metrics compute the diversity as the average "difference" between individuals in the population. In the case of the first metric, called "link diversity", we define the difference between the individuals as the number of links which are present in one of the individuals annot in the other (normalized by the total number of feasible links). In the case of the second metric, called "technology diversity", we define the difference between the individuals as the number of links which are present in one of the individuals and not in the other, or which being present in both individuals, have different technology types (always normalized by the total

number of feasible links).

Initial Population Method	Average Fitness	Standard Deviation	Link -diversity	Technology -diversity
		VTHD1		
P0	1017.09	220.95	0.33	0.33
P1	1526.86	30.94	0.04	0.04
P2	1525.08	30.95	0.04	0.04
		VTHD2		
P0	1241.72	212.4	0.31	0.47
P1	1735.09	33	0.04	0.04
P2	1745.84	33.96	0.04	0.29
		VTHD3		
P0	2333.42	239.48	0.31	0.47
P1	3625.09	33	0.04	0.04
P2	2705.24	194.84	0.04	0.29

Table 6: Initial Population methods, the average population fitness, its standard deviation, and two types of diversity metrics.

As can be observed, method P0 always results in much lower values for the average fitness of the initial population, with high values of diversity. Methods P1 and P2 obtain very similar values for the average fitness, except for the last test case (VTHD3). As the methods are similar, they obtain the same link diversity values; but method P2 has better technology diversity in the generated populations. The evolution operators (selection, crossover, mutation) define the evolution and the convergence of the method (defining the solution quality). There is an equilibrium between the evolution operators and the quality of initial population. For example, if we use a very good initial population and highly disruptive evolution operators, it is possible that the population will evolve to worsen its fitness.

Figure 7 shows a case where this phenomenon can be observed, with the new Initial Population method P1, without elitism, and a very disruptive mutation. As the Initial Population method gives good quality individuals, the first generations have a good benefit, but after a few generations the population fitness stabilizes or degrades, particularly in the problem with costly links (VTHD3).

The results of additional tests showed that methods P1 and P2 were better than method P0; and that in some cases, method P1 resulted in the behavior previously mentioned, while method P2 was more robust (we think due to its improved diversity).

4.3 Knowledge Uncertainty Effect

The simulations of calibration and validation are based on the assumption that we have the exact specification of the problem. In general, it is very difficult to know with certainty all the input data of a problem, specially if we work with inputs like the potential demand of a service. In this section we study the effect of imperfect knowledge about input data, that we call *"knowledge uncertainty"*.

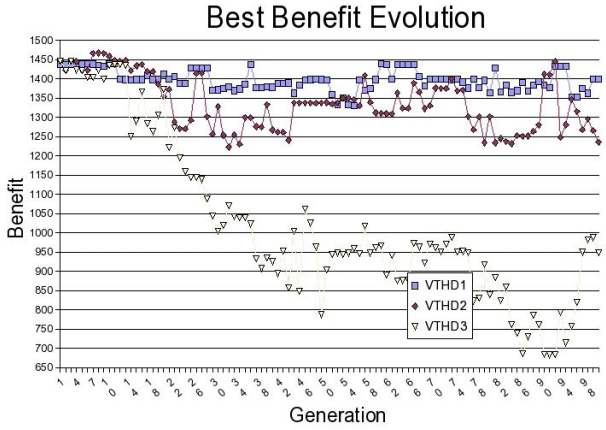


Figure 7: Best Benefit Divergence in Evolution. Bad selection of Initial Population method and evolution operators.

Especially, we want to see the effect of the following factors: *demand function* (parameters $M_{s,t}$ and $\alpha_{s,t}$, $\forall s,t \in \mathcal{K}$), *reliability* (parameters $\gamma_{s,t}$, $\forall s,t \in \mathcal{K}$) and *cost* (parameters $c_l(t)$, $\forall l \in \mathcal{E}, \forall t \in T(l)$), over the solution of the problem. In addition, in order to get a better idea of the robustness of our method, we show the effect of the *random seed* over the decision problem.

Generally, the uncertainty of the input data consists of several components, most of them being unknown and difficult to model. In this work, we consider a simple approach, we add a uniform noise to the input data, and we observe the effect of this noise in the solution.

We have run a set of simulations to see the effect of each factor (demand function, reliability, cost, and random seed). In the *demand function simulations*, for each pair of terminals $s,t \in \mathcal{K}$, we perturb the parameters $M_{s,t}$ and $\alpha_{s,t}$ of the demand function (3) with a uniform distribution between the half and the double of the true value (being true the parameter of the original problem).

We work in the same way with the *cost computation simulations*, where for each technology type t of each link (with cost greater than zero) $l \in \mathcal{E}$, we perturb the cost $c_l(t)$ with a uniform distribution between the half and the double of the true value.

To evaluate the effect of uncertainty in the *reliability*, we use the fact that in the fitness function (6) the reliability $r_{s,t}$ between two terminals $s,t \in \mathcal{K}$ is multiplied by factor $\gamma_{s,t}$. Therefore, we perturb $\gamma_{s,t}$ with a uniform distribution between the half and the double of the true value, which also can be interpreted like if we perturb the "importance" of reliability to the provisioning of the service.

Finally, in order to see the effect of the *random seed*, we simulate the same problem with several seeds.

The results of each set of simulations are summarized in Table 7. It can be observed that the impact of uncertainty is low (considering that we perturb

Evaluated Factor	Effect		
	Average Best Fitness	Standard Deviation (%)	Average Difference in Links
demand function	1657.5842	0.61	2.67
cost computation	1649.0169	2.32	2.83
reliability	**1951.1011**	**5.88**	**4.50**
random seed	1620.1055	1.04	3.33

Table 7: Knowledge Uncertainty Effect: four sets of simulations (each one with 5 simulations). In each set we perturb a parameter with a uniform distribution and evaluate the effect on the solution. We display the standard deviation of solution fitness value, and the average difference in links, that is the average number of different links between any solution pair of the set (the different links of a solution pair are those links that appear in only one of the two solutions).

so much the inputs), except when we modify the $\gamma_{s,t}$ parameters (reliability importance). The reliability estimation uncertainty changes a lot the fitness of the solution (in average 1951.1011 with a standard deviation of 5.88%), therefore our decision of network extension can be wrong (in average, each solution has 4.5 absolute different links than other solution). This makes sense, because our assumption is that the provisioning of the service is only based on connection reliability.

In addition to the uncertainty on the $\gamma_{s,t}$ parameters, this problem has a further difficulty: we have an error in the reliability computations. We then run another test to quantify what happens with the error in reliability estimation. The fitness function considers the reliabilities as parameters, therefore the law of propagation of uncertainty (see for instance [37]) can be applied to the fitness function to have an estimation of his uncertainty (uncertainty based on errors of the reliability computations). From the generic antithetic algorithm (used to estimate the reliability) we have an unbiased estimator of the variance $\mathrm{VAR}(r_{s,t})$ and covariance $\mathrm{COV}(r_{s,t}, r_{s',t'})$ (for details see [14]). In this statistical approach, the fitness variance is the power of two of the uncertainty, in this case: the variance propagation is given by

$$\mathrm{VAR}\left(f(r)\right) = \sum_{s,t\in\mathcal{K}} \left(\frac{\partial f}{\partial r_{s,t}}\right)^2 \mathrm{VAR}(r_{s,t}) + \sum_{s,t,s',t'\in\mathcal{K}} \left(\frac{\partial f}{\partial r_{s,t}}\right)\left(\frac{\partial f}{\partial r_{s',t'}}\right) \mathrm{COV}(r_{s,t}, r_{s',t'}). \tag{7}$$

Table 8 and Figure 8 show the fitness uncertainty resulting from the reliability calculations for different values of the general antithetic algorithm parameters. It shows that the average of uncertainty of fitness function in all the execution of our instance problem. It can be seen that for our chosen parameters ($B = 100, L = 50$) there is an average uncertainty of 3.08% (44.96 in absolute value), that for example is equivalent to the cost of four links of VTHD3 problem. As we expect, increasing the parameter values B and L results in a better reliability estimation (at the cost of a higher computational time). The decision of working with parameter values $B = 100$ and $L = 50$

is based principally in the computational time cost: we calibrate our genetic algorithm with 100 generation of a population of 100 individuals, it means that the reliability estimation run 10.000 times in a problem execution (in our test machine, the total execution time of VTHD1 problem is 4.95 hours [2], see Table 8). Using larger parameter values diminishes the fitness estimation uncertainty, but at the cost of prohibitive time execution. An important remark must be pointed out: the computed fitness uncertainty is only an upper bound of the uncertainty of the method, because our formulae do not consider that, with a very high probability, the fitness of an individual is calculated more than one time during the execution (that indirectly implies a variance reduction). The set of simulations with different random seeds is another way to see the global impact of uncertainty, and as we expect it is smaller than the previous statistical approach. A refinement of the trade-off between uncertainty in genetic algorithm and reliability estimation is out of the scope of the present work.

Parameters	Average Fitness Uncertainty ($\sqrt{\text{VAR}}$)	Average Fitness Porcentage Uncertainty (%)	Average Fitness Computation Time (ms)	Total Execution Time (hs.)
B=100,L=50	**44.96**	**3.08%**	**1782**	**4.95**
B=500,L=50	9.52	0.66%	8991	24.98
B=1000,L=50	4.80	0.32%	17940	49.83

Table 8: Fitness Function Uncertainty resulting from reliability estimation in all the execution of VTHD1 problem.

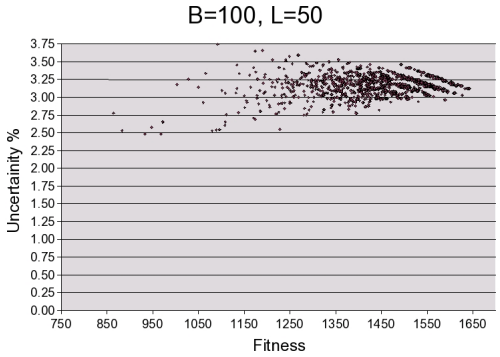
5 Conclusions and Future Work

In this paper, we study a new pricing scheme based on connection reliability. Defining a new charging scheme for telecommunication networks has become a hot topic in the scientific community. The main reasons to do that are: the exponential growth of the traffic creates congestion, and that new applications has different quality of service (QoS) requirements.

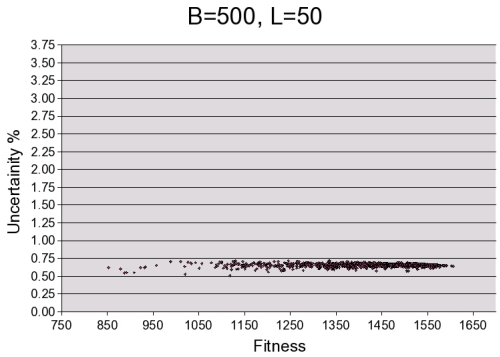
Due to the congestion problem, a service differentiation has to be devised, like in IntServ or Diffserv architectures. A pricing scheme has to be attached to it, otherwise each customer will choose the best available service class. In a radically opposed way, based on the growing idea that with the introduction of optic fiber, the backbones will be over-dimensioned, so that congestion will not occur. We propose here to study the charge of the network access based on connection reliability.

In our new charging scheme, the price for each connection between a source s and a destination t depends on the reliability of the connection between s and t. Of course demand is also varying with this price, so that the goal of the network

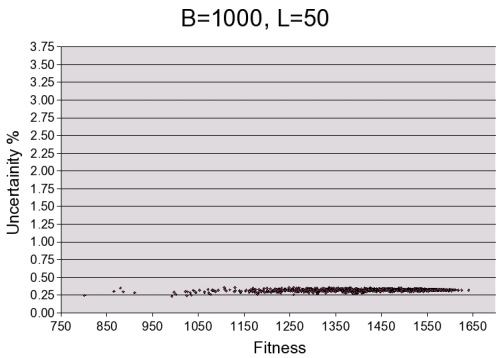
[2]The uncertainty effect is made in a SunFire 280R (like in the calibration step), with two 1.2 GHz UltraSPARC III Cu processors, 2 GB of main memory, and SolarisTM 8 operating system.



(a) B=100,L=50.



(b) B=500,L=50.



(c) B=1000,L=50.

Figure 8: Uncertainty distribution (%) over Fitness Function in all the execution of VTHD1 problem.

is to set up a price that maximizes its revenue. In a second stage, the problem is to extend an already existing network in order to increase the service provider's revenue. Solving this problem is intractable in general. Indeed, finding the optimal topology requires the computation of the reliability of each end-to-end connection, this problem being itself NP-hard. In this work we attempt to approach the solution with a genetic algorithm.

In general, the Genetic Algorithm method is a flexible iterative technique, where it is possible to finish the algorithm at a wished time and still have a good solution. The flexibility is given by some few calibrated simple parameters, for example, if we increase the population size or the generation number we can improve the effectiveness, with a cost of efficiency. These points make the GA an attractive metaheuristic to solve hard problems, like the network design based on price charging.

We calibrated and we validated the performance of the proposed algorithm by means of various examples. The performance is taken in the sense of effectiveness (i.e. near of the optimal one) and efficiency (i.e. computational effort). In the calibration stage we worked with 10 randomly chosen problems, and we defined the parameters of the genetic algorithm. The final level for each parameter are chosen with relative simplicity, because all the calibrated parameters had an clear impact on the performance of the solutions. To validate our method, we run three problems inspired by the VTHD (Very High Broadband IP/WDM test platform) network. One of the problems tries to extend the network with the election of some possible backbone links. The optimal solution to this problem can be easy found. The other two problems, with a bigger solution search space, also can upgrade the access network technology of some access link. The genetic algorithm found the optimal solution to the first problem, and a very good solution for the other ones.

The fitness evaluation is very hard because it needs a reliability estimation, in our work, it is estimated with the General Antithetic Variable method. Our GA evaluate the fitness a lot of times, therefore an important improvement can be made if we diminished the evaluation execution time or the numbers of evaluations. These can be made in different ways:

- We can use another method to the reliability estimation, for example an efficient upper bound expression. This approach is interesting because it can introduce the Service Level Agreement, based in reliability, in a natural way.

- During the execution of the GA it is highly probable that some reliability estimation was done more than one time. To avoid this situation, and increase the algorithm performance we can use the previous computed reliability estimations. Going beyond, it is possible to imagine a heuristic method or a problem reduction method to quickly estimate the reliability from previous computations.

- We can reduce the problem of reliability estimation if we eliminate the final links of our topology (typically the access network) and we calculate

the reliability of the access node to other nodes, only multiplying the probability of failure of the final link by the reliability found in the reduced problem.

- Another interesting approach is to have an decremental error in the reliability estimation. For example, we can increase the sample size of the General Antithetic Variable method in later generations.

In all cases, it is important to evaluate the impact of the reliability estimation error. We evaluate the standard deviation of the fitness function resulting from reliability estimation in all the execution of our validation problem, and we observe that it is high for some of our instance problems. The computed fitness standard deviation is an upper bound of the uncertainty of the GA method, because, with a very high probability, the fitness of a individual is calculated more than one time in the all execution (that indirectly implies an important variance reduction). A refinement trade-off between uncertainty in genetic algorithm and reliability estimation exceeds the present work, but it is important in order to have a robust method.

In general, it is very difficult to know with certainty all the input data of a problem, especially if we work with inputs like the potential demand of a service. In order to quantify the effect of imperfect knowledge about input data we made some additional simulations. Especially, we observed that the impact of uncertainty in the demand function parameters and cost parameters is low, but it is high respect to the reliability parameters (γ parameters). This makes sense, because our assumption is that the provisioning of the service is only based on connection reliability. Therefore, to apply our pricing schema, it is important to know as precisely as possible the clients reliability valuation and to have a very good reliability estimation.

As far as we know, this is the first experimental study that employs a pricing schema based in reliability. There are some future work and research directions:

- To use other GA variants. Particularly: double point crossover, dynamic mutation (to avoid stable evolution) and advanced elitism.

- Define the GA parameters according to the instance problem. For example, increase the number of generations and population size with the number of edges in the network.

Acknowledgement

This research was supported by ARMOR research project, IRISA-INRIA, France, and the PAIR cooperation project. The participation of Pablo Rodríguez was also supported in part by the "Jóvenes Investigadores program" of CSIC, UDELAR, Uruguay, and by the Internship program of INRIA, France.

We also acknowledge the use of "Dieste", the Sun SPARCcenter-2000 multiprocessor machine, which was donated by the Banco de la República Oriental del

Uruguay (BROU) to the Facultad de Ingeniería, Universidad de la República, Uruguay.

References

[1] Vthd: Very high broadband ip/wdm test platform for new generation internet applications. http://www.vthd.org, 1999.

[2] S. Blake and al. An architecture for differentiated services. Technical report, IETF RFC 2475, Dec. 1998.

[3] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. Technical Report 1633, 1994.

[4] H. Cancela and M. El Khadiri. On the rvr simulation algorithm for network reliability evaluation. *IEEE Transactions on Reliability*, 52(2):207–212, June 2003.

[5] R. Cocchi, D. Estrin, S. Shenker, and L. Zhang. Pricing in Computer Networks: Motivation, Formulation and Example. *IEEE/ACM Transactions on Networking*, 1(6):614–627, 1993.

[6] D. Connolly. General Purpose Simulated Annealing. *J. Op. Res. Soc.*, 43:495–505, 1992.

[7] L.A. DaSilva. Pricing of QoS-Enabled Networks: A Survey. *IEEE Communications Surveys & Tutorials*, 3(2), 2000.

[8] Darren L. Deeter and Alice E. Smith. Heuristic optimization of network design considering all-terminal relaibility. In *Proceedings of the 1997 Annual reliability and Maintainability Symposium*, pages 194–199, Philadelphia PA, 1997.

[9] Darren L. Deeter and Alice E. Smith. Economic design of reliable networks. *IIE Transactions*, 30:1161–1174, 1998.

[10] Dengiz, F. Altiparmak, and A. Smith. Genetic algorithm design of networks considering all-terminal reliability. In *Proceedings of the 6th Industrial Engineering Research Conference*, pages 30–35, Miami Beach, FL, 1997.

[11] B. Dengiz, F. Altiparmak, and A. E. Smith. Local search Genetic Algorithm for optimal design of reliable networks. *IEEE Trans. on Evolutionary Computation*, 1(3):179–188, 1997.

[12] Mathilde Durvy, Christophe Dior, Nina Taft, and Patrick Thiran. Network availability based service differentiation.

[13] R.W. Eglese. Simulated Annealing: A tool for Operational Research. *European Journal of Operational Research*, 46:271–281, 1990.

[14] M. El Khadiri and G. Rubino. A Monte-Carlo method based on antithetic variates for network reliability computations. Technical report, Publication Interne nro 626, IRISA-INRIA, 1992.

[15] M. Falkner, M. Devetsikiotis, and I. Lambadaris. An Overview of Pricing Concepts for Broadband IP Networks. *IEEE Communications Surveys & Tutorials*, 3(2), 2000.

[16] G. S. Fishman. A Comparison of Four Monte Carlo Methods for Estimating the Probability of s-t Connectedness. *IEEE Transactions on reliability*, 35(2):145–155, June 1986.

[17] L.J. Fogel. Toward inductive inference automata. In *In Proc. of the Int. Federation for Information Processing Congress Conf, Munich*, page pages 395 399, 1962.

[18] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[19] A. Gupta, D.O. Stahl, and A.B. Whinston. Priority Pricing of Integrated Services Networks. In Lee W. McKnight and Joseph P. Bailey, editors, *Internet Economics*, pages 323–352. MIT Press, 1997.

[20] J.M. Hammersley and D.C. Handscomb. Monte carlo methods, Wiley, New York, 1964.

[21] T. Henderson, J. Crowcroft, and S. Bhatti. Congestion Pricing. Paying Your Way in Communication Networks. *IEEE Internet Computing*, September/October:85–89, 2001.

[22] J. Holland. Adaption in natural and artificial systems. Technical report, University of Michigan Press, 1975.

[23] L. Ingber. Simulated annealing: Practive versus Theory. *J. Mathl. Comput. Modelling*, 18:29–57, 1993.

[24] F.P. Kelly. Mathematical modelling of the Internet. In *Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, 2000.

[25] F.P. Kelly, A.K. Mauloo, and D.K.H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[26] H. Kumamoto, K. Tanaka, K. Inoueb, and E.J. Henley. Dagger-sampling Monte Carlo for system unavailability evaluation. In *IEEE Trans. Reliab., R-29(2)*, pages 122–125, 1980.

[27] A. J. Owens L. J. Fogel and M. J. Walsh. Artificial intelligence through simulated evolution. Technical report, John Wiley, New York, 1966.

[28] A.A. Lazar and N. Semret. Design and Analysis of the Progressive Second Price Auction for Network Bandwidth Sharing. *To appear in Telecommunication Systems*, 13, 2001. `http://comet.columbia.edu/~nemo/telecomsys.pdf`.

[29] S.H. Low and D.E. Lapsley. Optimization Flow Control, I: Basic Algorithm and Convergence. *IEEE/ACM Transactions on Networking*, 7(6), 1999.

[30] H. Mendelson and S. Whang. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations Research*, 38(5):870–883, 1990.

[31] A. Odlyzko. A modest proposal for preventing Internet congestion. Technical report, AT&T Labs, 1997.

[32] J.S. Proban and M.O. Ball. The complexity of counting cuts and the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.

[33] I. Rechenberg. Evolution strategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. Technical report, Frommann-Holzboog, 1973.

[34] G. Rubino. Efficient Evaluation of Network Reliability. 7th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, May 1994.

[35] R. Y. Rubinstein. Simulation and the monte carlo method, Wiley, New York, 1981.

[36] N. Semret. *Market Mechanisms for Network Resource Sharing*. PhD thesis, Columbia University, 1999.

[37] B. N. Taylor and C. E. Kuyatt. Guidelines for evaluating and expressing the uncertainty of nist measurement results, 1994.

[38] K.S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley & Sons, 2002. Second Edition.

[39] B. Tuffin. Charging the Internet without bandwidth reservation: an overview and bibliography of mathematical approaches. *Journal of Information Science and Engineering*, 19(5):765–786, 2003.

[40] L.G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.

[41] Darrell Whitley. A genetic Algorithm tutorial. *Statistics and Computing*, 4:65–85, 1994.

6 Appendix: Calibration Problems

In the calibration stage we work with 10 randomly chosen problems. The number of nodes $|V|$, terminals $|K|$ and edges $|E|$ are shown in Table 9. Also, in this Table we display the best known solution for each problem (that is used to calculate the relative error of a solution).

The Figures 9 and 10 show the topology of the 10 problems.

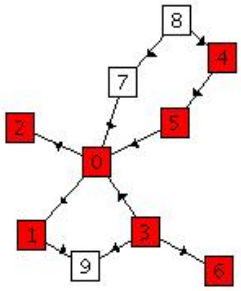
Problem				Best known
No.	$\|V\|$	$\|K\|$	$\|E\|$	solution
1	10	7	11	9.2011
2	14	3	13	11.3057
3	18	6	17	28.5912
4	20	6	16	26.4200
5	13	6	12	36.1494
6	16	7	14	15.7550
7	18	5	18	18.5863
8	10	5	13	12.3404
9	18	5	11	18.3268
10	14	7	10	37.4170

Table 9: Best known solutions.

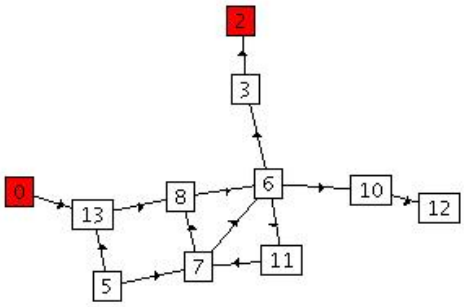
7 Appendix: VTHD Problem

As we show in Section 4.3, the problem is sensitive to parameter values, specially to the reliability cost $\gamma_{s,t}$. To understand the solutions given by our method in Section 4.1.2, is important to have a completely specification of the problems. In this Section, we specified the valitadion problems: VTHD1, VTHD2 and VTHD3.

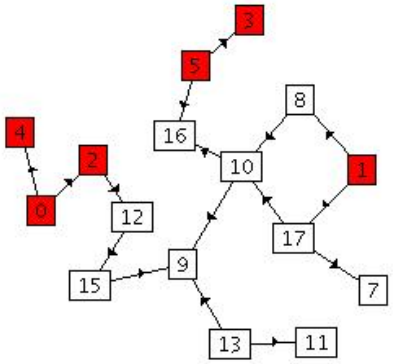
The Table 11 show the parametes: $\gamma_{s,t}$, $\alpha_{s,t}$, $M_{s,t}$, $\lambda_{s,t}$ and $\mu_{s,t}$ $\forall s, t \in \mathcal{K}$. The network topology is in Figure 4, and the cost and probability of failure of each technology type of each link is in Table 10.



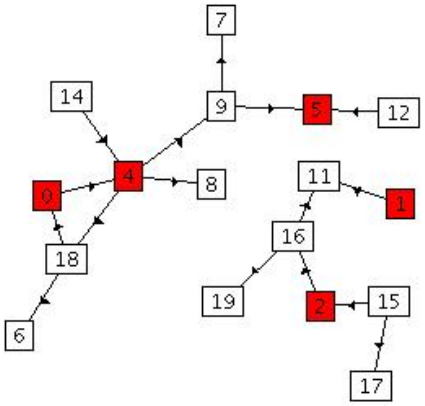
(a) Random Case 1.



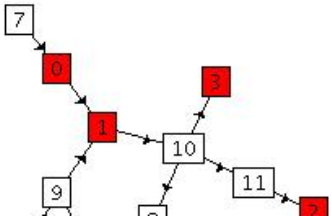
(b) Random Case 2.

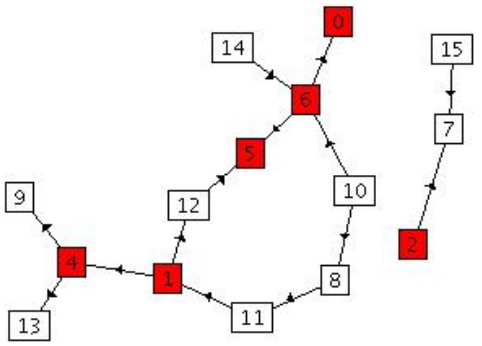


(c) Random Case 3.

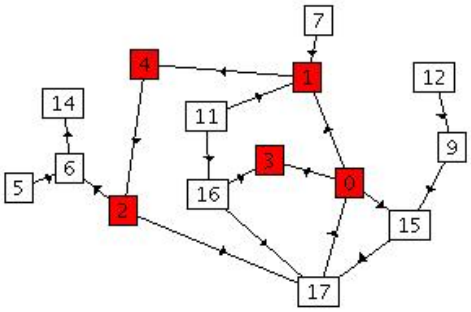


(d) Random Case 4.

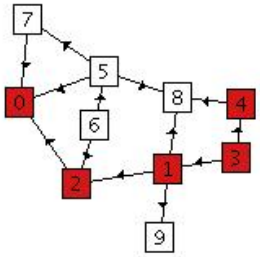




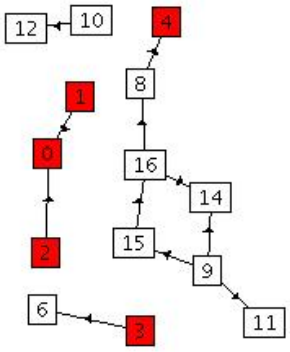
(a) Random Case 6.



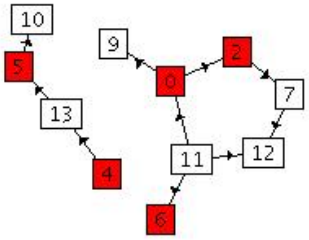
(b) Random Case 7.



(c) Random Case 8.



(d) Random Case 9.



(e) Random Case 10.

Figure 10: Calibration Random Problems (Cont).

Edge		Problem											
		VTHD1 Technology Types				VTHD2 Technology Types				VTHD3 Technology Types			
s	t	cost -	reliability	cost -	reliability	cost -	reliability	cost -	reliability	cost -	reliability	cost -	reliability
0	1	0.00	0.9999	-	-	0.00	0.9999	-	-	0.00	0.9999	-	-
0	2	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
0	3	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
0	4	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
1	2	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
1	5	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
1	18	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
1	19	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
1	20	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
1	21	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
1	22	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
1	23	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
1	24	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
2	3	20.00	0.9900	-	-	20.00	0.9900	-	-	20.00	0.9900	-	-
2	7	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
2	25	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
2	26	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
3	7	30.00	0.9900	-	-	30.00	0.9900	-	-	30.00	0.9900	-	-
3	9	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
4	5	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
4	10	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
4	11	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
5	6	50.00	0.9900	-	-	50.00	0.9900	-	-	50.00	0.9900	-	-
5	7	40.00	0.9900	-	-	40.00	0.9900	-	-	40.00	0.9900	-	-
5	12	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
5	14	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
5	15	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
5	16	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
5	17	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
6	7	20.00	0.9900	-	-	20.00	0.9900	-	-	20.00	0.9900	-	-
6	8	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
6	27	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
6	28	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
6	29	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
7	8	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
7	34	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
7	35	0.00	0.9900	-	-	0.00	0.9900	-	-	0.00	0.9900	-	-
8	30	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
8	31	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
8	32	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
8	33	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900
12	13	0.00	0.9000	-	-	0.00	0.9000	10.00	0.9900	0.00	0.9000	100.00	0.9900

Table 10: VTHD Problem Specifications. Edges and Technology types.

Terminals		Utility of reliability	Demand parameters		Connection and duration rates	
s	t	$\gamma_{s,t}$	$\alpha_{s,t}$	$M_{s,t}$	$\lambda_{s,t}$	$\mu_{s,t}$
9	10	5.57300	79.86713	0.34911	0.66716	0.02633
9	12	0.42260	86.72620	0.51328	0.70939	0.61642
9	18	7.65040	99.73000	0.12610	0.99542	0.97359
9	19	4.27530	60.51875	0.32188	0.80431	0.32188
9	20	2.86620	82.64264	0.32080	0.39863	0.24169
9	22	9.12560	48.05131	0.87071	0.94101	0.93250
9	25	0.00000	1.10343	0.90418	0.84078	0.34842
9	29	1.61930	99.73000	0.16544	0.95078	0.92381
9	30	2.94250	34.84239	0.92921	0.47670	0.19688
9	35	3.57270	34.51328	0.52000	0.96918	0.86726
10	12	0.00140	76.77875	0.34513	0.80824	0.71376
10	13	5.94450	83.19955	0.52899	0.96962	0.87071
10	14	2.24660	49.71011	0.14489	0.96418	0.81394
10	15	3.92130	40.10529	0.26009	0.90409	0.80929
10	18	3.21880	97.07513	0.04707	0.91167	0.41288
10	21	9.75860	51.99998	0.18848	0.92981	0.62229
10	22	0.00140	18.84777	0.84969	0.73442	0.34513
10	23	4.12880	80.74945	0.41040	0.86426	0.84403
10	24	3.77480	92.18798	0.43695	0.83705	0.43695
10	25	4.81170	14.48915	0.16193	0.71789	0.40105
10	26	9.31840	4.75389	0.52190	0.20095	0.17677
10	30	2.41690	4.77025	0.48051	0.95741	0.93806
10	31	7.02550	1.28316	0.78549	0.35412	0.04754
10	33	4.36950	56.36716	0.50230	0.92079	0.87398
10	34	3.49110	22.46565	0.76779	0.86787	0.33971
11	12	5.02300	79.86713	0.43695	0.73587	0.08306
11	16	2.86480	28.64770	0.04700	0.92892	0.91483
11	18	4.36950	33.97149	0.45396	0.47584	0.19469
11	19	1.44890	23.89434	0.60452	0.63592	0.39213
11	20	9.66790	91.48283	0.41922	0.92721	0.78549
11	22	9.97300	37.50990	0.07659	0.06577	0.09900
11	23	2.86480	34.85503	0.52000	0.97999	0.24226
11	24	8.03320	84.40318	0.29602	0.22600	0.18737
11	26	0.48230	45.39644	0.08580	0.50473	0.40044
11	27	3.30070	93.25021	0.89656	0.82819	0.81394
11	30	4.12880	34.84566	0.66066	0.21546	0.17677
11	34	4.97100	41.38052	0.52190	0.97561	0.07659
11	35	1.69670	68.82781	0.05925	0.38793	0.35727
12	14	1.76770	55.72996	0.65816	0.17106	0.04754
12	17	2.06960	80.92918	0.00001	0.38239	0.23894
12	18	6.58160	19.25662	0.33007	0.92889	0.89656
12	24	0.19880	41.04035	0.51328	0.74346	0.62229
12	26	5.13280	76.77875	0.13604	0.92565	0.92188
12	28	8.26430	87.39822	0.48117	0.35145	0.11306
12	33	9.29210	37.74779	0.71964	0.93833	0.09522

Table 11: VTHD Problem Specifications.

13	14	8.15690	19.46903	0.23230	0.86173	0.52000
13	16	6.04520	71.14193	0.70255	0.77283	0.28662
13	18	4.27530	2.93715	0.09522	0.71475	0.51328
13	25	9.31840	14.37777	0.07659	0.53378	0.35617
13	26	6.05190	71.06016	0.16193	0.38078	0.14489
13	32	3.53990	5.92468	0.96679	0.95639	0.29602
13	33	2.42260	62.22918	0.31237	0.50327	0.15263
13	34	9.21880	41.28762	0.09179	0.99893	0.99730
14	22	2.05750	79.86713	0.63717	0.81457	0.68417
14	23	1.26100	23.02256	0.91496	0.34051	0.20575
14	26	7.55640	5.65149	0.14489	0.52185	0.32080
14	29	3.48550	65.81584	0.09179	0.83577	0.77320
14	31	6.28320	60.51875	0.41394	0.51431	0.04226
14	32	4.19220	65.81584	0.79989	0.64283	0.52899
14	34	1.04210	37.50990	0.41288	0.38440	0.35399
15	16	9.73590	3.88299	0.84403	0.73784	0.68155
15	17	0.47000	43.69522	0.17677	0.87991	0.87398
15	18	6.58160	89.65575	0.31256	0.94429	0.71060
15	20	8.26430	60.45221	0.29602	0.26130	0.02937
15	22	1.94690	10.42073	0.23230	0.57723	0.29486
15	26	2.24660	94.08759	0.81569	0.88419	0.84956
15	29	1.43780	4.77025	0.02633	0.77413	0.63717
15	31	1.29690	16.54415	0.70255	0.91263	0.41922
15	34	9.32500	23.89434	0.42753	0.81254	0.78549
16	20	9.32500	41.38052	0.37510	0.86116	0.86116
16	26	1.36040	0.00000	0.84403	0.93346	0.84969
16	29	0.42260	15.26272	0.00001	0.94774	0.80332
16	33	0.56510	34.51328	0.79989	0.87511	0.56367
16	34	2.60090	76.50440	0.09522	0.49696	0.28662
17	22	8.14160	29.42465	0.12969	0.97748	0.32191
17	25	4.12880	41.39418	0.48051	0.98314	0.93806
17	26	3.45130	38.05310	0.12610	0.83122	0.81416
17	29	1.43780	26.87853	0.76504	0.67999	0.45396
17	30	1.52630	5.65149	0.16967	0.56406	0.52000
17	34	3.12370	89.11052	0.34855	0.94369	0.59445
18	20	4.27530	14.48915	0.10080	0.96866	0.92921
18	21	0.56510	4.22604	0.86116	0.33976	0.02633
19	21	3.20800	4.82299	0.38053	0.78190	0.35727
19	22	3.75100	42.75310	0.35727	0.82889	0.41381
19	26	7.27870	7.65930	0.70810	0.99612	0.86726
19	28	3.56170	90.41808	0.09522	0.73977	0.71060
19	30	2.32300	41.04035	0.09179	0.42340	0.23894
20	22	3.12370	20.57509	0.80332	0.20235	0.04823
20	26	1.73940	34.84566	0.92381	0.99252	0.99093
20	28	2.30230	20.69574	0.07659	0.98622	0.52899
20	35	9.14960	91.48283	0.70255	0.94541	0.80929
21	26	9.32490	81.39368	0.82643	0.74680	0.04707
21	33	9.29210	94.08759	0.56367	0.19869	0.12610
22	28	1.87370	51.65982	0.17394	0.63743	0.52899
22	33	8.15690	19.68750	0.97359	0.29425	0.29425

Table 12: VTHD Problem Specifications (cont.).

23	25	2.51770	12.96903	0.33971	0.94754	0.49288
23	26	8.49560	84.96946	0.49288	0.59862	0.16967
23	27	0.85800	40.10529	0.32191	0.79139	0.56367
23	31	4.10400	13.60419	0.32191	0.70616	0.39213
23	34	5.57300	24.22619	0.17500	0.09491	0.00343
25	29	8.44030	32.07952	0.66066	0.86026	0.78549
25	32	6.28320	80.33224	0.60452	0.94424	0.17394
25	33	4.53960	77.32027	0.32188	0.38250	0.16544
26	30	1.29690	86.11593	0.99730	0.92642	0.81394
26	32	7.55640	26.87853	0.97075	0.99794	0.99093
27	30	7.11950	55.72996	0.79867	0.85850	0.81569
27	32	3.56170	93.18405	0.37510	0.78028	0.72787
27	33	5.20000	55.75222	0.77320	0.98454	0.77320
27	34	0.11030	29.42465	0.71142	0.93257	0.63717
28	29	2.42260	24.16903	0.33007	0.56941	0.14378
28	31	0.00010	20.69574	0.32191	0.75523	0.14489
28	32	6.60660	13.60419	0.23894	0.66995	0.16544
28	34	2.68790	12.61049	0.04754	0.89213	0.18737
29	30	6.58160	61.64160	0.02937	0.76431	0.70810
29	31	3.53990	84.96946	0.34513	0.13929	0.12969
30	32	5.16600	3.88299	0.52899	0.83016	0.71195
30	34	8.49690	38.05310	0.00343	0.30737	0.08580
31	33	2.77630	80.33224	0.96679	0.42296	0.37510
31	34	2.94250	68.41682	0.73429	0.96992	0.92381
32	33	4.10400	27.76349	0.26009	0.72995	0.05655
33	34	8.44030	31.23715	0.01103	0.75690	0.49710

Table 13: VTHD Problem Specifications (cont.).