The University of Auckland

Business School &

The Polytechnic University of Catalonia

(CFIS-ETSETB)

An agent-based simulation model of a market for dynamic spectrum allocation

Master Thesis

Ester López

Supervisor: Dr. Fernando Beltrán

Barcelona, February 2011

Acknowledges

This project would not have been possible without the kind invitation from the CODE research centre at the University of Auckland. Those who made it possible, Margo, Jairo and Jose Luis, have my deepest gratitude. Also Pau, who gave me the courage to make the final decision to go that great country, New Zealand.

Many thanks to Fernando for its infinite patience and for guiding on the right direction during the whole project. And Catharina, for all the small things that made the project progress smoothly.

I would like to thank all my friends back at Spain, who kept supporting me from the distance; and the new friends that I made in New Zealand, who made my experience there marvellous.

Finally, thanks to my family and Marc, for being my source of strength and motivation, never loosing the confidence in me.

Abstract

The dramatic increase in the demand for spectrum-based services and devices, combined with the long-term exclusive model of spectrum management, has lead to an apparent spectrum scarcity. To overcome this scarcity, a more dynamic spectrum management has been proposed from the scientific community, using cognitive radio networks and Dynamic Spectrum Management (DSM). But before seeing the deployment of these new technologies, social and economic consequences have to be analysed. This work proposes an agent-based simulation model for modelling the market under a DSM panorama, so that the proper policy changes can be performed. The design principles of the proposed model are simplicity, to provide a fast prototype; and modularity, so that it can be easily modified to include new market mechanisms, roles' behaviours or any other change that would be potentially analysed.

.

Contents

1 Introduction

Recently, we have seen a continuous increase in the number of wireless-oriented devices. We expect for the oncoming devices to be cordless, such as headphones, security cameras, and even our car keys. Moreover, we also intend to obtain higher data rate transmissions when accessing network contents, and preferably in an economic way. What we used to obtain from our personal computer, at home, with a wired connection we want it available from our laptops, mobiles and PDAs. Independently of where we are. Wirelessly.

Each wireless device has a different working principle; some need a dedicated channel, others use spread-spectrum techniques to become more noise-resistant. But in the end, all of them have a set of electromagnetic requirements that must be satisfied in order to work properly. These requirements can be simplified in some restrictions on the noise and interference level in the operating band. We have to take into account that the air is a shared medium, where it is impossible to isolate one device's transmission from another's. Every country government has designed a frequency plan, whereby it is specified how each frequency band can be used. As regards to the commercial uses of the spectrum, governments authorise each operator through an auction for the spectrum bands, which will grant the winning entity the use of the band for a long period of time (usually 10 years) and the geographic area to use it. On the other hand, there are some unlicensed bands which can be used by anyone, while satisfying some transmission restrictions (power level).

The constant increase of spectrum-based services and devices has shown the inefficiency of the current policy, whereby the unlicensed bands are saturated and prevent new operators to obtain a license to provide new services, giving the false sensation that the whole spectrum is already totally used up. In fact, there are spectrum utilization studies that show that the typical band occupancy is around 15% [1].

The scientist community is making an effort to give solutions to this problem, and some technological solutions have been already proposed and designed. These solutions are based on a dynamic spectrum access, where the transmissions parameters can vary depending on the environment and device conditions. The first step of this work has been the design of software-defined radios (SDR), which are radios whose radio-frequency parameters, such as frequency, power or modulation, can be specified by software, so their working frequency is not fixed, but variable, making it able to operate in a very versatile way. The next step has been the design of cognitive radios, which are radios that are environment sensitive and can sense the spectrum utilisation, detecting unused bands and selecting them, as well as the best RF parameters to perform its transmission.

Despite of already existing technological possibilities, a policy and legal change is needed to allow them to be used. However this kind of change needs some guarantees of its utility and operation, not only at technological level, but also at social and economic level.

Several approaches have been taken to analyse the economic factors that take part in dynamic spectrum management situations, using game theory, optimization and economics theory, to move forward enabling these technologies. In this work, we propose a novelty approach, based on agents, so it can provide answers to the economic effects that dynamic spectrum management would have.

The goal of this project is to develop a basic market model based on agents, which will allow us to have a close view into the economics questions and mechanisms that appear when we use a dynamic spectrum allocation instead of the current one.

2 Literature Review

Radio spectrum is a public resource used as a communication channel in its wide sense. A communication using this channel results in an electromagnetic radiation emission at radio frequencies (between 30 kHz and 300 GHz). The emission of a single radio may cause unacceptable interference levels for nearby radios that may be sharing an overlapping spectrum band. Therefore, some kind of regulation of the spectrum access to balance the conflicting interests of the parties involved is called for.

Currently radio spectrum regulation is used to mitigate all such undesirable effects using two basic spectrum access models:

1. ***Command and control*** model, whereby a centralized regulatory body assigns portions of spectrum to an entity to use while following a detailed set of rules. The entity has exclusive and nearly eternal access to the assigned spectrum band. This is the common access model used for the spectrum bands that the government needs, such as the bands used by the military, for radio astronomy or aeronautical operations.
2. For commercial uses of the spectrum, spectrum band licenses are usually sold in an auction. The winner of the auction is the exclusive owner of the spectrum band license for long periods of time (usually 10 years). The access model in this case is the ***long-term exclusive*** model, which manages the spectrum considering 3 dimensions: space, frequency and type-of-service, which restrains the spectrum band use. In several European and Asian countries the norm in the current state-of-the-art is to specify further technology details. Ofcom in UK and FCC in the U.S.A. are gradually adopting looser regulations, without specifying the technology. However, service agnostic licensing is still not usual [2, 3].

On the other hand, there has been a dramatic increase in overall demand for spectrum-based services and devices, making wireless communications everyone's mainstream source of connectivity. The factors that have motivated this demand are: the shift of economies towards the communications-intensive service sector, the increasingly mobile workforce, and the fact that users have quickly embraced wireless devices due to their convenience and their increased efficiency [4].

Besides the constant increase in demand, the increasing dynamism of the applications complicates the spectrum management. The current policy, based on typical worst case predictive interference models, makes it look like there is no free spectrum to use, with only few unlicensed bands that are overcrowded. Meanwhile, studies like the one conducted in New York in 2002, show that the spectrum scarcity is just apparent, being the typical channel occupancy less than 15% [1]. As a result, it is important to update the current spectrum policies to policies that manage the spectrum considering the dynamic nature of its use [5, 6].

This necessity has stimulated some advances in policy changes, starting with the above mentioned report by the Spectrum Policy Task Force, which recommends the FCC to "*evolve from the current approach to spectrum policy to a more integrated, market-oriented approach*" [5]. Later, in 2002, a Notice of Proposed Rule Making was published, which proposes to allow unlicensed spectrum access to the white-spaces in the TV bands. In 2008 the FCC approved it.

These progresses would not be possible without the necessary technological advances. There are three main concepts that make a more dynamic management of the spectrum possible:

1. Software defined radio, a radio in which radio frequency operating parameters including but not limited to frequency range, modulation type or output power can be set or altered by software.
2. Cognitive radio networks, a type of radio network in which the behaviour of each radio is controlled by a cognitive control mechanism to adapt to changes in topology, operating conditions, or user needs. Their nodes do not have to be cognitive radios, but receive instructions from a node with such capabilities.
3. Dynamic Spectrum Access is the real-time adjustment of spectrum utilization in response to changing circumstances and objectives [3].

Using the possibilities offered by the technology and the direction that the regulatory framework is heading to, we analyze the economical relationships that appear when a more dynamic management of the spectrum becomes a reality.

2.1 Economic Models Reviewed

Before a major change in the current spectrum policy, dynamic spectrum management has to prove not only its technological feasibility, but its economic and social desirability. The establishment of Dynamic Spectrum Management (DSM) will generate many changes in the spectrum-related market; on the one hand the increased available bandwidth and lower entry costs will facilitate the development and introduction of new services boosting the technological development on the area. On the other hand, DSM breaks the relation between the huge investment for the spectrum rights and service offering, making possible the introduction of new business models with new agents, where the infrastructure, spectrum rights and spectrum access/use may belong to different parties [7].

It is important to analyse the effect on the economy and society of the introduction of these new business models, so that the correct policy changes are performed. As a first step, several models defining new market structures, agents and their relationships have been proposed in the literature. Next subsections present a sample of models that served us as a starting point. We can classify these models into models that only consider the spectrum buyers and sellers, and those where the spectrum sellers will use it to provide some service, so their behaviour is ultimately related to the end users. We refer to the first type of models as one-level models, as they only consider one transaction level, and the second type as two-level model, as there are two transaction levels, closely related.

Before describing the models, we introduce some common and basic vocabulary to facilitate their comprehension.

- **Primary user** or **primary service operator**: It is the spectrum's right owner, who does not use it all the time or in its entirety. As a result, it is willing to sublease part of its spectrum band to others, in exchange for a monetary reward.
- **Secondary user** or **secondary service operators**: they are users of the spectrum who do not own a spectrum license but want to use it; therefore, they are willing to pay to the primary users in order to gain access to the spectrum.
- **Spectrum broker**: entity that is specialised in the spectrum management in secondary markets. Its functions may range from matching spectrum demands with spectrum supply, to finding the optimal allocation of a concrete spectrum band among the secondary users, to managing a pre-assigned spectrum band dedicated to DSA, etc.
- **Spectral efficiency**: is the information rate that can be achieved while transmitting over a given spectrum band. It is measured in bps/Hz.
- **Discount factor**: it relates to the compensation a service provider gives to their premium users when their QoS level is not satisfied due to a shortage of bandwidth [8].
- **Utility function**: it measures the utility of the different agents obtained during a transaction. This function is used afterwards to model and predict their behaviour, as the agents want to maximize their own utility.

2.1.1 One-level Models

One-level models take into consideration the possibility of spectrum holders who underutilize their spectrum band and decide to sell part of this spectrum band to secondary users for monetary gains. Different agents are defined to map these spectrum sellers, such as the primary users, who hold spectrum access rights or a government agent who manages a concrete band for dynamic spectrum access. In the same way, the spectrum buyers are wireless service providers or final users depending on the model considered. But all the models coincide in that the asset traded is a portion of the spectrum, that is, a spectrum channel. We can sub-classify further the models considering which set of agents' interests compete on the model: none, the buyers', the sellers' or both.

2.1.1.1 *No competition*

Most of the models that do not consider competition among sellers and buyers, are based on market equilibrium, that is, the solution considered when the demand matches the supply.

Control-theoretic approach

On [9] there is a single primary service (seller) operating in multiple frequencies, each of which is serving a set of primary users. The primary service seeks to earn additional revenue from a secondary service (buyer) who wishes to purchase some bandwidth from the primary service.

The behaviour of both is defined by their utility functions. The utility of the primary service depends on the price charged to the secondary service and the quantity of spectrum sold. It includes a discount factor that must be subtracted when the performance of the primary users is affected by the bandwidth subleasing, due to insufficient bandwidth to serve their connections. Therefore the profit obtained from frequency band i available to the primary service is defined by:

$$\pi_i = x_1 N_{p,i} + P_i Q_i - x_2 N_{p,i} \left(B_i^{req} - \frac{k_i^{(p)} (W_i - Q_i)}{N_{p,i}} \right)^2$$

The first term is the revenue obtained by the primary users, as $N_{p,i}$ is the number of primary users connected to the frequency band i and x_1 is the price charged to them. The second term is the revenue from secondary users: P_i is the price charged to the secondary service and Q_i is the quantity of bandwidth the secondary service has subleased. The next term represents the discount factor due to QoS performance degradation: B_i^{req} is the primary user spectrum requirement, $k_i^{(p)}$ is the spectral efficiency for the primary users and W_i is the size of this spectrum band. x_2 is a constant.

The optimal price as a function of the bandwidth demand can be obtained through the profit derivate.

On the other hand, the utility of the secondary service is defined by a quadratic function of the bandwidth obtained:

$$U = \sum_{i=1}^{2} Q_i k_i^{(s)} - \frac{1}{2} \left(\sum_{i=1}^{2} Q_i^2 + 2\Delta Q_1 Q_2 \right) - \sum_{i=1}^{2} P_i Q_i \, , assuming\ 2\ frequency\ bands$$

Where $k_i^{(s)}$ is the spectral efficiency of the secondary service when using frequency band i, and Δ indicates the impact of price in spectrum substitutability. Likewise, the demand is obtained by differentiating the utility function, obtaining a bandwidth quantity as a function of the price offered by the primary service.

The solution proposed in this paper is where the demand matches the supply.

Market-equilibrium-based approach

[10] proposes the same market model as the one described previously: one single primary service offering bandwidth opportunities to one secondary service provider (Figure 1). However in this case the utility on the secondary service is considered at the user level, assuming all of the bandwidth purchased by the secondary service is equally divided among the secondary users. The proposed user utility function is a logarithmic function of the bandwidth:

$$U(b) = \ln b + c_3$$

Where b is the bandwidth allocated to the secondary user and c_3 is a constant. The total utility for the secondary service is:

$$\sigma_s = c_3 n_s + n_s * \ln\left(\frac{b_s}{n_s}\right) - p * b_s$$

n_s is the number of secondary users, *p* is the price per bandwidth unit and b_s is the bandwidth purchased to the primary service.

Just as before, the supply and the demand are obtained by differentiating the involved utility functions, and a market equilibrium solution is obtained.

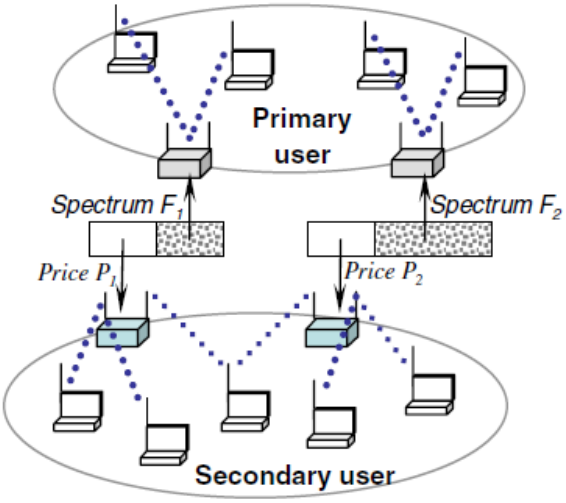


Figure 1: Market structure of [9] **and** [10]

2.1.1.2 *Competition among sellers*

The models described below consider competition among the spectrum sellers, using different approaches to obtain an optimal price for the spectrum. Competition is caused by the existence of multiple sellers that provide the same product to the buyers; thus, their revenues are influenced by other sellers' strategies. Some of these models go a step further and propose cooperative strategies among the sellers, to gain higher revenues, colluding with others to set an optimal price.

Cooperative game approach

In [11] we have 3 different service providers, based on 3 different technologies (WLAN, WMAN and CDMA), who provide services to a end user with 3 network interfaces, so it can connect to the 3 service providers simultaneously (Figure 2). In this case the end user has a bandwidth requirement, and asks for it to the 3 different service providers. The problem is formulated as a bankruptcy game, where the service providers have to split up among them the bandwidth requirement. The solution obtained is based on calculating the core and using the Shapley value of a cooperative game to select the amount of bandwidth each service provider will offer. This model is quite peculiar as it is based on the quantity of bandwidth offered instead of the price charged by the service providers.

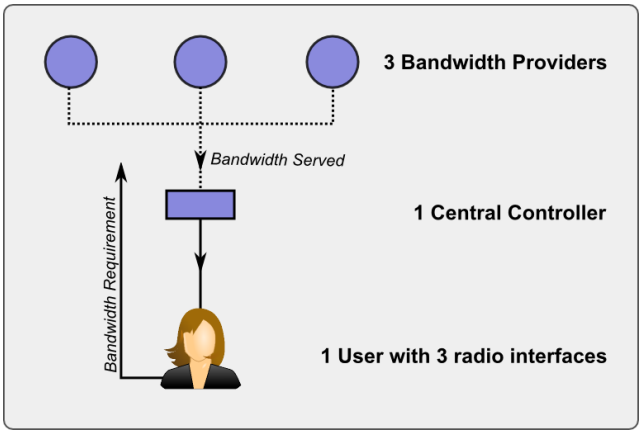


Figure 2: Market structure of [11]

Game theoretic analysis and comparison

In [8], [12] and [13] the models described are almost the same: there are a few service providers who offer two types of connections: premium and best-effort connections (Figure 4). The premium connections are fixed and assume fixed revenue for the service provider, but if their QoS requirements are not satisfied because too much bandwidth is being subleased to secondary services, a discount factor is applied. On the other hand the best-effort connections (or secondary users) are not attached to a concrete service provider so they may churn to another to maximize their payoff. As in the first model reviewed, the bandwidth demand function is obtained based on maximizing the utility function, which is modelled by a quadratic utility function of the allocated bandwidth:

$$\mathrm{U}(\mathbf{b}) = \sum_{i=1}^{N} b_i k_i^{(s)} - \frac{1}{2}\left(\sum_{i=1}^{N} b_i^2 + \sum_{j \neq i} 2v b_i b_j\right) - \sum_{i=1}^{N} P_i b_i$$

Once obtained the demand, 4 different approaches have been followed to reach the solution in regards to the price setting by the service providers (Figure 3):

a. Neglecting that they are in fact competing, and using a market equilibrium approach, as in the first model reviewed.
b. Considering a simultaneous play game, where the solution is the Nash equilibrium.
c. Considering a leader-follower game and the solution is the Stackelberg.
d. Considering all the service providers collude, so the global revenue is maximized. In that case the optimal price is obtained through an optimization approach. In [8] the revenue is divided among them using the Shapley value afterwards.

When comparing the different solutions, collusion is proved to be the most profitable. However it implies higher communication among the service providers and when using dynamic algorithms to reach equilibrium is the least stable. On the other hand is the market equilibrium solution, which implies no communication among service providers and high stability.

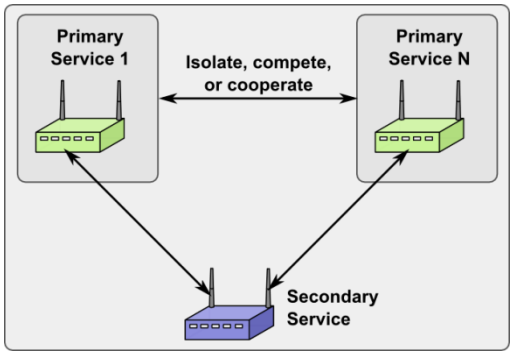


Figure 3: Relationships analysed in [13, 14]

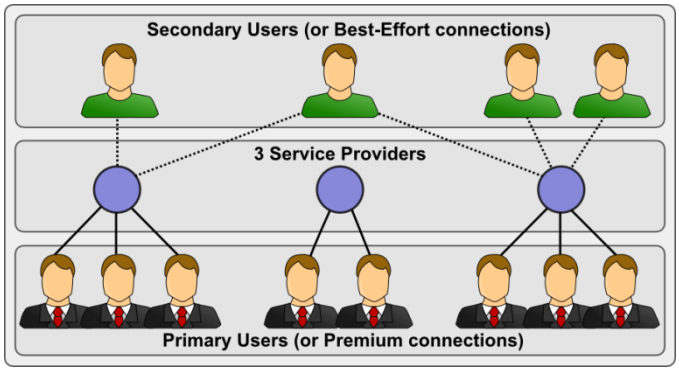


Figure 4: Market structure in [8], [12] and [13]

2.1.1.3 *Competition among buyers*

In these models, the problem considered is that the price of the bandwidth increases as the demand increases or, alternatively, the total available bandwidth must be shared out among all the users:

Competitive spectrum sharing approach

In [15] and [16] a primary user sells spectrum to the secondary users setting the unitary price through the following function:

$$c(\boldsymbol{b}) = x + y * \left(\sum_{i=1}^{N} b_i\right)^{\tau}$$

Where $c(\boldsymbol{b})$ is the cost per unit of spectrum, $\boldsymbol{b}$= [b_1, b_2,b_3..b_N] is the bandwidth demand vector, x and y are constats and $\tau \geq 1$, rendering the pricing function convex. The bandwidth demand is modelled as a Cournot game, where the Nash equilibrium is considered the solution. The revenue function used as payoff is:

$$p_i(\boldsymbol{b}) = r_i k_i b_i - b_i * c(\boldsymbol{b})$$

Where r_i is a constant that measures the value the user gives to the transmission and k_i is the spectral efficiency.

The main drawback of this approach is that to solve the problem you need to solve a set of differential functions, which vary depending on the number of secondary users. So even

though the same solution may be helpful to analyse the effects of the different parameters, to analyse the same situation under different number of users, you have to solve the equation system once for each number of users.

Collusion-resistant multi-winner auction

In [17] a collusion-resistant multi-winner auction is proposed. There is a spectrum broker with N spectrum bands available and M secondary users who bid for them. The allocation algorithm considers interference; therefore, the same spectrum band can be assigned to multiple secondary users, so the auction is multi-winner. To avoid collusion among the secondary users, the algorithm works with virtual bidders, converting the auction into a single winner auction, concretely a second price auction. After the allocation, the price established for the winner virtual bidder is split among the users that form that virtual bidder using a Nash bargain solution.

Efficient spectrum allocation algorithms

[18] proposes two efficient allocation algorithms based on a model where a spectrum broker manages a coordinated spectrum band. The spectrum broker assigns short term spectrum leases to base stations situated in its controlled region. The coordinated spectrum band is divided in K channels. Each base station estimates its spectrum demand, and performs a spectrum request represented by the pair $<d_{min}, d_{max}>$; where d_{min} is the minimum number of channels the base station requires and d_{max} the maximum number of channels. On the other hand, the spectrum broker can estimate the interference level between any two base stations given their location. Based on this information the spectrum broker generates an interference graph and uses one of the two designed algorithms to allocate the bandwidth. The first allocation algorithm assigns to the base station their minimum requests and minimizes the interference between them. The second one restricted to the interference graph maximizes the overall demand serviced among the different base stations.

2.1.1.4 *Competition among sellers and buyers*

The following models consider the competition among buyers as well as among sellers, complicating the problem formulation.

Multiple sellers and multiple buyers model

The model proposed on [19] consists in multiple licensed users (primary users) selling spectrum opportunities to multiple unlicensed users (secondary users). The utility obtained by all users based on their allocated bandwidth is modelled by a logarithmic function:

$$U(c) = u_1 \log(u_2 * c * k)$$

Where *c* is the spectrum size and *k* the spectral efficiency; u_1 and u_2 are constants that depend on the application type.

Primary users decide the portion of spectrum they want to share with the secondary users as well as the price (per user per time) they charge to secondary users. Competition among primary users is modelled using game theory as a non cooperative game where each primary

user maximizes its own net utility. Its net utility is the utility obtained by the spectrum they are not sharing plus the revenue obtained by sharing part of their spectrum:

$$N_i(\boldsymbol{c},\boldsymbol{p}) = U(C_i - c_i) + p_i n_i(\boldsymbol{c},\boldsymbol{p})$$

Where **c**= [c1,c2…cN] and **p**=[p1,p2…pN] are the vectors of primary users' strategies: shared spectrum size and price. C_i is the total spectrum available to primary user *i* and $n_i(\boldsymbol{c},\boldsymbol{p})$ is the number of secondary users that select primary user *i* spectrum opportunities for the given strategies. As a non-cooperative game, Nash equilibrium is the solution considered.

In contrast competition among secondary users is modelled as an evolutionary game. Secondary users are grouped based on the channel conditions they experience and the available spectrum opportunities, so users from the same group face the same conditions. Therefore the end users expect to obtain at least the same net utility as the users in their same group. Their net utility is expressed as the utility obtained by their allocated bandwidth minus its cost:

$$\pi_i^a = u_1 \log\left(u_2 * \frac{c_i k_i}{n_i}\right) - p_i$$

Where π_i^a refers to the net utility of the secondary user on area a, when chooses primary user *i*. n_i is the number of secondary users that choose this primary user and p_i is the price it charges. The solution of the evolutionary game is the evolutionary equilibrium, where all the users in the same region obtain the same revenue.

Multi-stage pricing game

In [20] there are multiple primary users who have license to operate in N channels in total, and the collection of all these channels is considered as a spectrum pool (Figure 5). These channels are auctioned to secondary users using an open ascending price auction. Each primary user has a set of available channels, $\mathbf{A}_i$; each of which supposes a cost c_i^j; therefore the primary user's utility is:

$$\mathrm{U_{p_i}} = \sum_{j \in A_i} (\phi_i^j - c_i^j)\alpha_j$$

Where ϕ_i^j is the price the primary user *i* is charging for the leased band *j* and $\alpha_i = \{0,1\}$ indicates whether the band has been leased or not.

Secondary users obtain revenue for each channel they lease:

$$\mathrm{U_{s_i}} = \sum_{j=0}^{N} (v_i^j - \phi^j)\beta_i^j$$

Where v_i^j is the revenue the secondary user *i* obtains when he leases the channel *j*, ϕ^j is the price paid for it and β_i^j indicates whether the band *j* has been leased to secondary user *i* or not.

To avoid collusion, both primary users and secondary users set a threshold value for the price they are willing to approve to go on with the transaction (minimum price or reserve price for the primary users and the maximum price or bid for the secondary users). Users obtain these values using a belief learning algorithm.

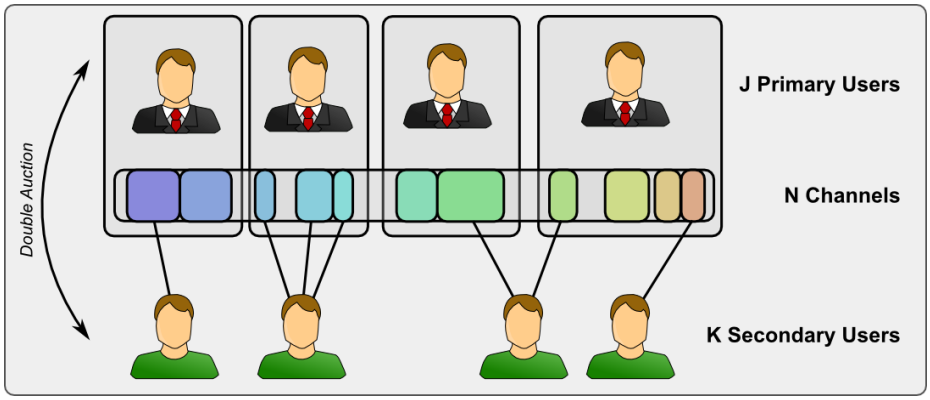


Figure 5: Market structure of [20]

Competitive pricing

[21] defines a model with multiple primary users and multiple secondary users. M primary users lease their underused spectrum to N secondary users in order to get some extra benefits (Figure 6). They compete among them by setting an optimal price per unit of bandwidth. On the other hand the secondary users select the best primary user based on their quality evaluation and price and request for an optimal amount of bandwidth. Both strategies are obtained from the utility function of each agent, which is for the secondary user:

$$U_{ik} = \sum_{k=1}^{M} I(i,k) * [q_{ik} * U_s(B_{ik}) - B_{ik} * \lambda_k]$$

Where $I(i,k)$ can only be 1 or 0, which indicates secondary user *i* have leased primary user *k* spectrum or not. q_{ik} Is the quality of the primary user *k* and λ_k is its price.$U_s(B_{ik})$ is the benefit of leasing a spectrum portion of size B_{ik}, and satisfies the following relation:

$$U_s(0) = 0;$$
$$U_s(B) = U_s(B_{max}) \quad for\ B \geq B_{max}$$

Similarly, the primary user's utility is defined as:

$$U_k^p = \sum_{i=0}^{N} I(i,k) * [B_{ik} * \lambda_k - B_{ik}\, c_k]$$

Where c_k is the fixed cost for leasing a unit of spectrum and the rest of parameters are the same of the secondary user's equation.

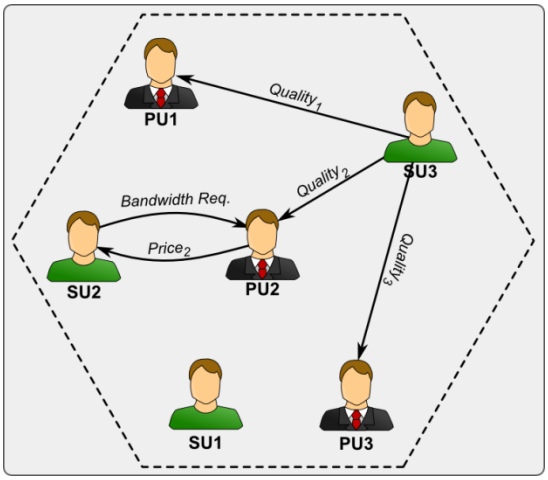


Figure 6: Market model in [21]

2.1.2 Two-level models

Two-level models consider the influence of the network load when purchasing spectrum from the spectrum seller, so that the desired amount of bandwidth or the price the service providers are willing to pay for it, vary depending on the actual number of users they serve.

2.1.2.1 *Competitive spectrum sharing and pricing*

[22] considers a scenario where routers in the secondary users' network form a mesh network, which is overlaid on networks of several primary service providers, to relay the secondary users' traffic through multiple hops to destination (Figure 7). Primary users compete among themselves to maximize their revenues by choosing an optimal price. Secondary users compete for spectrum usage to choose the source rate to maximize their utilities. A game theory model is used to solve these problems as a non-cooperative game; in there, Nash equilibrium is considered as the solution.

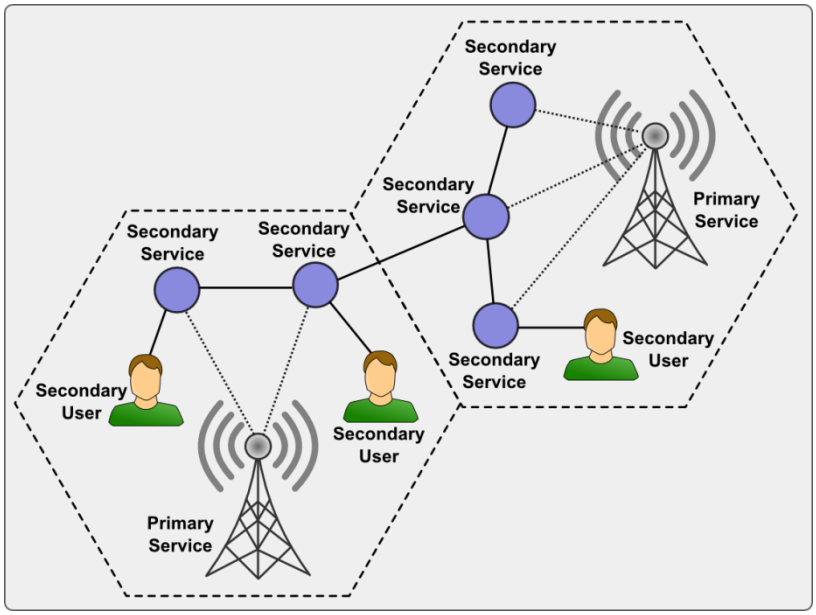


Figure 7: Market structure of [22]

2.1.2.2 *Competitive spectrum bidding and pricing*

[23] Identifies key research issues related to spectrum trading in IEEE 802.22-Based cognitive wireless networks, presenting a hierarchical spectrum trading model. On this model, Wireless Regional Area Network (WRAN) service providers purchase TV bands from TV broadcasters through a double auction, whereby the price per band is determined by the number of bands demanded (Figure 8).

Simultaneously, WRAN service providers compete with each other by adjusting the service price charged to WRAN users. Using game theory and modelling the problem as a non-cooperative game, the optimal price as well as the optimal amount of TV bands can be obtained through the Nash equilibrium.

WRAN service providers' selection by the users is based on their utility; they select the service provider in order to maximize it:

$$\pi_i = U\left(\frac{W * k_i * c_i}{n_i}(1 - FER_i)\right) - p_i$$

The utility obtained when choosing service provider *i*, is the utility of the bandwidth allocated ($U(\cdot)$) minus the price charged by the service provider, p_i. *W* is the size of the TV band, k_i is the spectral efficiency and c_i is the number of TV bands purchased by the WRAN service provider *i*. n_i is the number of users that select this service provider and FER_i is the frame error rate.

On the other hand, the WRAN service provider utility is:

$$P_i = u_i(\boldsymbol{c}, \boldsymbol{p}) * p_i - c_i p^{(t)}(\boldsymbol{c})$$

Where $u_i(\boldsymbol{c}, \boldsymbol{p})$, is the number of users that select the WRAN service provider *i* and $\boldsymbol{p}^{(t)}(\boldsymbol{c})$ is the price charged by the TV broadcasters per TV band. $\mathbf{c}$ and $\mathbf{p}$ are the vectors of number of channels demanded and price charged by the service provider respectively.

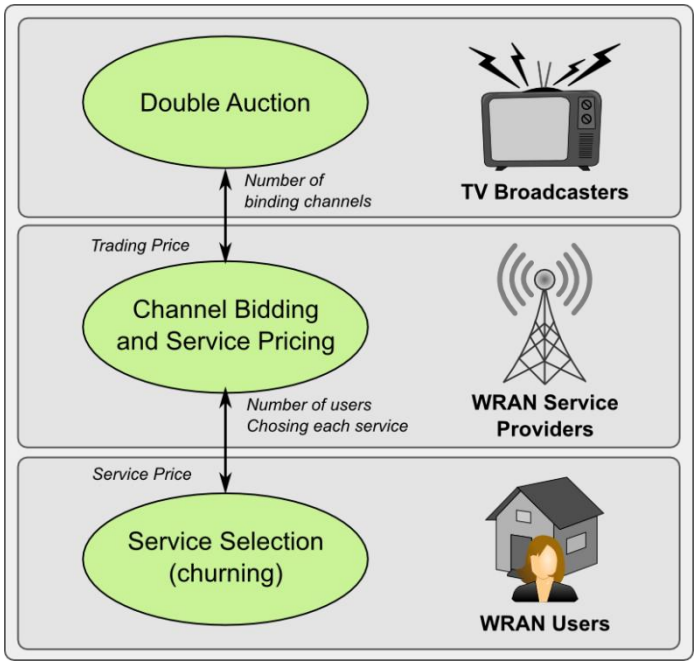


Figure 8: Market model in [23]

2.1.2.3 *Flexible ownership based spectrum management*

[24] presents the D-Pass model, in which spectrum portions are allocated to operators on a short-term basis by a Spectrum Policy Server (SPS) (Figure 9). Prior to each short term allocation, the SPS determines the optimal spectrum partition to maximize a system-related function. The operators are charged by the SPS for the amount of bandwidth allocated. The operators compete with each other for users present in the system through demand responsive pricing, in the form of an iterative bidding scheme reminiscent of simultaneous ascending auctions. At the beginning of each auction's iteration, the operators make offers of rates and prices to the users, who respond with the probabilities of accepting the service offers made. The user's acceptance probability is a function of the rate and the price:

$$A(R,P) = 1 - e^{-C*u(R)^{\mu}*P^{\epsilon}}$$

Where μ is the utility sensitivity and ϵ is the price sensitivity. *R* and *P* are the rate and price offered respectively and *C* is an appropriate constant. The rate is considered through a utility function $u(R)$ which represents the utility a user achieves when it communicates with rate *R*:

$$u(R) = \frac{\left(\frac{R}{K}\right)^{\zeta}}{1 + \left(\frac{R}{K}\right)^{\zeta}}$$

Where K and ζ are parameters that determine the exact shape of the sigmoid function.

The operators offer concrete prices and rates to maximize their expected profit, which can be mathematically expressed as:

$$Q_i(\overrightarrow{R_i}, \overrightarrow{P_i}) = \sum_{n \epsilon \mathbf{N}} A(R_{i,n}, P_{i,n}) * (P_{i,n} - F_i) - W_i * V$$

Where **N** is the set of users the operator *i* makes offers to, $\overrightarrow{R_i}, \overrightarrow{P_i}$ are the offer vectors which specify the offer to each user. F_i is the fixed operational cost incurred by operator *i* when serving any user. W_i is the total bandwidth allocated to operator *i* and *V* is the price per unit of bandwidth it has to pay to the SPS.

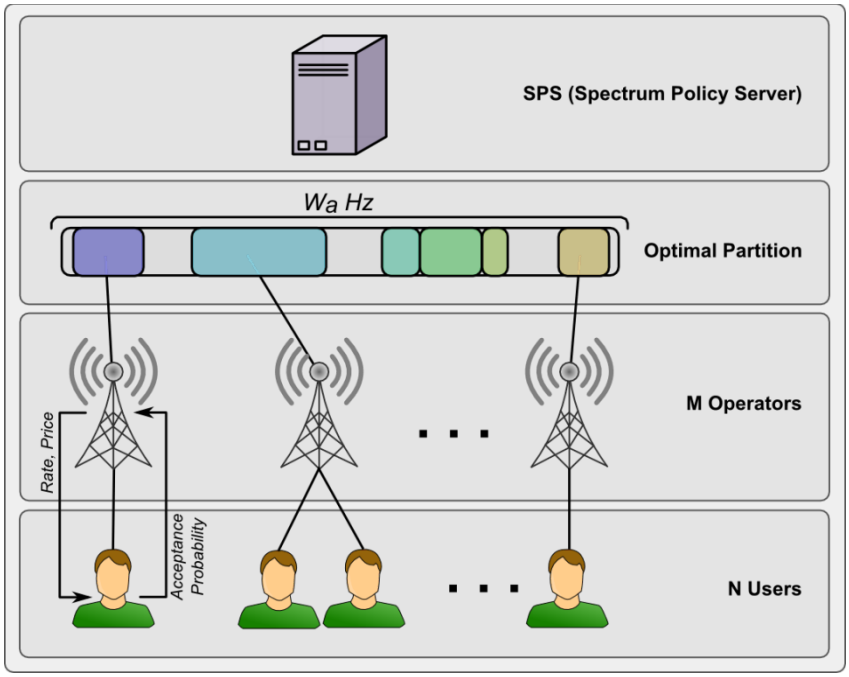


Figure 9: Market structure of [24]

2.1.2.4 *Auction driven dynamic spectrum allocation*

In [14] a spectrum manager implements DSA by periodically auctioning short term spectrum licenses to operators. The spectrum available for DSA is divided into K bands; and each operator submits a bid vector with k components, where the k-th component means how much he offers to pay for an additional band if k-1 bands have been already assigned to him. The auction proposed is a multi-unit Vickrey auction, which allocates the spectrum bands to the highest bids and the price a bidder has to pay for N won bands is the sum of the highest N loosing bids submitted by the other operators.

To decide the value of each bid, the service operators maximize their own utility based on their users' behaviour. Users' utility is measured by the frame-success function, i.e. the probability of receiving a data packet correctly as a function of the signal to interference ratio, which has an "S" shape; and the value it has for the user to transfer it.

This approach implies that the service operator needs to implement a service priority mechanism to determine which users will be served.

Paper	Competitors	Approach	Solution	Levels	Demand
[11]	Bandwidth providers	Game theory	Bankruptcy game → Shapley value	1	-
[16]	Secondary users	Game theory	Cournot game → Nash equilibrium	1	From the revenue function:
[8]	Service providers	Game theory	Simultaneous-play → Nash equilibrium	1	From the utility function, which is a quadratic function of the allocated bandwidth.
			Leader-follower game → Stackelberg equilibrium		
		Optimization	Collusion → optimal price		
[12]	Primary service providers	Game theory	Bertrand game → Nash equilibrium	1	From the utility function, which is quadratic.
		Optimization	Collusion → optimal price		
[22]	Primary service providers + secondary users	Game theory	Non-cooperative games → Nash equilibrium	2	-
[15]	Secondary users	Game theory	Cournot game → Nash equilibrium	1	Derivated from a revenue function
[19]	Secondary users	Game theory	Evolutionary game	1	
	Primary users		Non-cooperative game → Nash equilibrium		
[23]	WRAN service providers	Game theory		2	
[9]	-	Economics	Market equilibrium (demand=supply)	1	Derivates from a quadratic utility function.
[13]	Primary service	Economics	Market equilibrium	1	Derivates from a quadratic utility function.
		Game Theory	Non-cooperative game → Nash equilibrium1		
		Optimization	Optimal price		
[10]	-	Economics	Market equilibrium	1	Derivates from a logarithmic utility function.
[17]	Secondary users	Auction	Virtual bidders + Nash bargain price	1	-
[24]	Operators	Competition for bandwidth: optimization	2 different maximization algorithms proposed	2	Derivates from an acceptance probability, which is an exponential function of the price and a sigmoid utility function of the rate.
		Competition for users: auction	Iterative bidding scheme based on simultaneous ascending auction		
[18]	Wireless service providers	Auction	Efficient algorithms to solve NP-Hard optimization problems	1	-
[20]	Primary users	Auction	Iterative double auction with belief learning to set asks/bids	1	-
	Secondary users				
[21]	Primary users	Optimization	Utility maximization	1	Derivates from a concave utility function.
	Secondary users				
[14]	Operators	Auction	Multi-unit Vickrey Auction	2	Derivates from the QoS

Different approaches have been presented to analyse the emerging market structure that appears under the DSM panorama, but the solution proposed by them, either lacks of a further analysis in the market results or they are presented under concrete circumstances (number of agents participating), as their analysis requires the resolution of a set of equations.

It is the goal of this project to develop a modular analysis tool that allows researchers to use it to analyse and compare the market outcomes under different scenarios, where most of the parameters can be configured at run time, without the need of additional effort.

3 A Model of Dynamic Spectrum Trading

Cognitive radio networks and dynamic spectrum access allow a much more intensive spectrum usage overcoming the apparent spectrum scarcity problem. There are two different models for spectrum access that are enabled by these technologies: (1) dynamic exclusive-use and (2) shared-use of primary licensed spectrum. On the exclusive-use, the owner has exclusive right to use the spectrum under certain rules, enabling the appearance of secondary markets. As it is dynamic, the spectrum is managed in finer scales of time, space, frequency and use than the current model, which assigns the spectrum in a large area and long-term basis. On the shared-use model the spectrum is owned by a licensee or primary user, and is shared with other users, usually known as secondary users. A secondary user can access the spectrum in an underlay way, using ultra wide band and low power techniques or in an overlay fashion, sensing the spectrum and detecting the spectrum opportunities to perform its communication. As the shared-use model cannot provide performance guarantees, the exclusive-use model is preferred for commercial use and is the one that we have considered in this thesis. In our model the available spectrum for secondary access is managed by the spectrum broker, who sells spectrum access rights to the service providers on a periodic basis. During the period every service provider will have the exclusive right to access the band allocated, using it to provide service to the end users. On the other hand the end users purchase the services from the service providers and are free to choose the service provider that maximizes their utility.

3.1 Agents

The model we present here considers a system with 3 main agents:

1. Spectrum broker,
2. Service providers, and
3. End users.

As the implementation of the model requires a flexible, efficient way to manage communication between end users and service providers, we also introduce another agent, the service broker.

3.1.1 Spectrum Broker

Spectrum Broker manages a certain portion of the spectrum for secondary use. It sells short-term spectrum licenses to the service providers, rendering the spectrum allocation more dynamic. Each license is valid during a cycle, with service providers having to purchase a new

portion of spectrum thereafter. It is clear that a government agency may be such an agent, managing a previously selected band for DSA. Another possibility, which may not be so obvious, is that primary users transfer the spectrum rights of their unused spectrum to a third party, -probably owned by primary users-, so that they can improve their revenues by leasing their unused spectrum. In this case it could be the spectrum broker itself that takes responsibility for sensing, detecting and selecting the spectrum opportunities that will eventually be allocated to secondary users. The portion of band leased by the spectrum broker can vary along the cycles as a result of the load impose on the network by the primary users.

When allocating the spectrum, the interference among the users must be taken into consideration, as two users using the same band are likely to interfere on each other´s transmissions. Interference is determined by the distance between the users and the power of transmission, so once these parameters are known, spectrum is allocated without interference among all the users in the system. In this case two different users can use the same spectrum band as long as they are far enough and its transmission power is low enough. Yet another possibility is to perform an allocation by setting power transmission restrictions to grant there is no interference among the secondary users. Consequently such decision will affect the coverage area of secondary users, as their power level is restricted. In our model, we do not take into account space or power dimensions, assuming only that the managed area is coverage-restricted, so all the service providers interfere. The latter means that non-overlapping spectrum bands must be allocated to each user.

Every DSA cycle the spectrum broker receives the bandwidth requirements from every service provider. If the total demand does not exceed the available bandwidth, the allocation procedure is simple: each service provider gets its expected share of bandwidth. Otherwise some allocation algorithm must be used.

We, in a first instance, propose a purely proportional allocation, with each provider's share proportional to the demanded bandwidth. The following expression shows the allocation:

$$b_{allocated\ ,i} = b_{requested\ ,i} * \frac{b_{available}}{\sum_{j=1}^{numWSP} b_{requested\ ,j}}$$

Where $b_{allocated\ ,i}$ is the bandwidth allocated to the service provider *i*, $b_{requested\ ,i}$ is the original requested bandwidth from service provider *i*, and $numWSP$ is the number of service providers. Clearly, this strategy is not a good one, as it provides incentives to service providers to overestimate their needs, therefore demanding more bandwidth than necessary. As result, a fairer algorithm has been implemented, whereby every wireless service provider is allocated as much spectrum as they want as far as this allocation does not harms other wireless service providers' interests. That can be achieved by allocating every wireless service provider bandwidth in small quantities incrementally until they do not require more bandwidth, or there is no more bandwidth available. It can be considered as an iterative process, for example, let's suppose the spectrum broker has 11 units of bandwidth to allocate, and there are 3 service providers in the market; the first WSP requests 3 units of bandwidth, the second one 4, and the third one, 6. The resulting allocation would be 3 units for the first WSP, and 4

units for the other two service providers. The steps to reach this result can be found in the following table:

Iteration	Remaining bandwidth	Bandwidth allocated to WSP1	Bandwidth allocated to WSP2	Bandwidth allocated to WSP3	
0	11	0	0	0	
1	8	1	1	1	
2	5	2	2	2	
3	2	3	3	3	WSP1 does not want more bandwidth
4	0	3	4	4	

Table 1: Example with the steps of the fair allocation algorithm

Once every service provider's share is decided, its price must be settled. This model proposes to let the price be set by the spectrum broker with service providers being price-takers. We can use [15] to express the unit cost of bandwidth:

$$c(\boldsymbol{B}) = x + y\left(\sum_{j=1}^{numWSP} b_j\right)^{\tau}$$

Where *x* and *y* are non-negative constants and $\tau \geq 1$ (so that the pricing function is convex) and **B**= [b_1, b_2, …] denotes the vector of allocated spectrum quantities; *numWSP* refers, as before, to the number of service providers. As a result, the price so found is higher for higher bandwidth demand. The function shape varies depending on its different parameters, as can be seen in Figure 10.

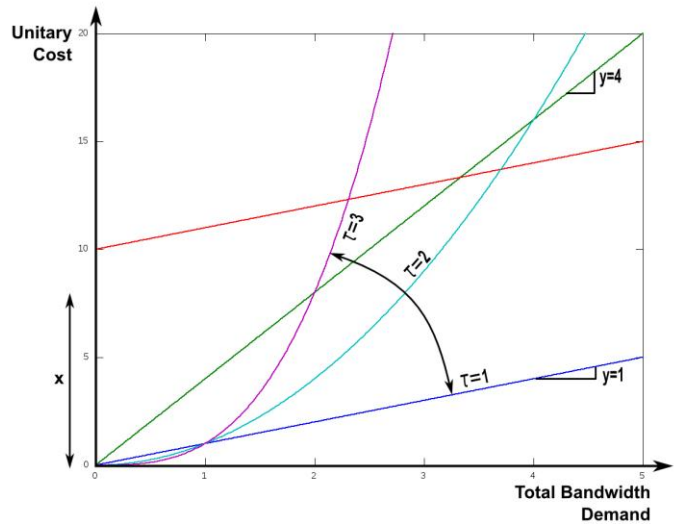


Figure 10: Price per unit of bandwidth as a function of the total demand

The steps described above are repeated continuously for every DSA cycle.

3.1.2 Service Provider

Dynamic Spectrum Management (DSM) allows new entrepreneurs to have a chance in wireless service provisioning markets as it lowers entry barriers by making it easier to purchase spectrum access rights. Such agents are the Wireless Service Providers (WSP). The WSP raise

revenue by providing service to their users. These services cover a full range of internet-based access as well as services offered over the WSP platform, such as videoconference, voice calls, etc. In order to simplify the market model we have considered that all the WSP can provide the same services and we characterize them by the data rate needed to serve them and their duration. The latter simplifies the model as other QoS parameters do not have to be considered, and all the WSP are equivalent for the users as far as they have enough bandwidth to provide them the required data rate. Assuming that the WSP can provide the same services without distinction is reasonable as the IP convergence makes all the applications available on the same network.

What distinguishes one WSP from others is its efficiency, which is the number of bits they can fit on the same bandwidth. The parameter that measures the relation between the available bandwidth and the total data rate available for the WSP is the spectral efficiency:

$$C = k * BW$$

Where *k* is the spectral efficiency, *BW* is its available bandwidth and *C* the data rate it can provide with that bandwidth. The service provider has a base station or access point that regulates the access from the end users, granting them access only if they have enough free bandwidth, that is, only if:

$$\sum_{j \in N} C_j \leq k * BW_{allocated\ ,i}$$

where N is the set of end users connected to the service provider at a given time and C_j is the user *j´*s current data rate. As the geographical coverage area assumed to be small, all base stations are visible to all users, so they can select the service the provider that maximizes their utility

As can be seen the service providers are chosen according to their price, the latter being the sole criterion by which users chose them. Such interaction creates a first layer of competition among the service providers, who try to lure as many users as they can. As the providers lower their price, more users would seek to connect to this network, but if they raise the price, they might raise more revenue from their connected users. Therefore, there must be an optimal price that maximizes the service provider's revenue; it not only depends on the number of users and the amount of bandwidth allocated to the service provider, but also on the prices the others service providers´ prices.

Because choosing a proper pricing method is critical for the WSP, we propose a traffic-based pricing method instead of using flat rate pricing; besides flat rate pricing encourages social waste [25]. Now, since traffic-based tariffs can be specified in an infinite number of different pricing models, we have opted for a dynamic pricing model, whereby the traffic unit price varies with time and the available bandwidth. As a consequence the price increases as the service provider becomes more congested; in fact, pricing now resembles more of a congestion control tool. The price is obtained as follows:

$$p = (Price_{\min} - Price_{max}) * \left(\frac{freeCapacity}{maxCapacity}\right)^{\tau} + Price_{max}$$

The price will fluctuate between $Price_{\min}$ (the minimum price), at full provider´s capacity, and $Price_{max}$ (the maximum price), when there is capacity has been exhausted (Figure 11). Hereby the price is set by defining the three parameters: $Price_{max}$, $Price_{max}$ and τ.

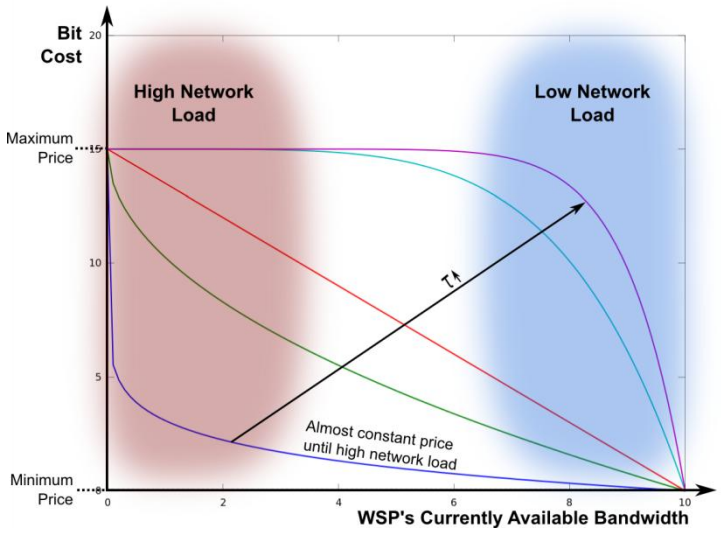


Figure 11: Price per transmitted bit as a function of the available bandwidth

Service providers also compete for bandwidth, as they will obtain different amounts of bandwidth depending on the others' bandwidth requests. Therefore, their strategy is built upon two parameters: the quantity of bandwidth they request, and the price per bit transferred. In an agent-based simulation model it is important to define the agents' goals, as their strategies must be aligned with them. In this case it is clear that the agent's goal is to maximize its own revenue, so that the bandwidth quantity and price must be chosen to achieve such goal.

The revenue of the WSP is determined by the incomes obtained from the end users and the cost of the bandwidth purchased for the Spectrum Broker:

$$Revenue_{cycle} = incomes_{cycle} - BW_{cost}$$

The revenue obtained in a cycle is the received incomes less the cost of the allocated bandwidth .Incomes will depend on the price as well as the allocated bandwidth: the higher the price, the higher revenues. On the other hand, the lower the price, the more users willing to join the WSP. Regarding the available bandwidth, the higher amount of bandwidth, the higher spectrum opportunities that can be offered to the end users. The bandwidth cost will depend on the bandwidth allocated as well as the others' bandwidth requests.

3.1.3 End User

End users create market opportunities for the service providers, and their presence demands a more efficient management of the spectrum, justifying the spectrum broker´s role. An end

user needs to perform a connection to enable him to communicate with someone else; the latter includes either calling another user, holding a videoconference, checking e-mail and so on. We refer to any such call as a session. Every session is characterized by a data rate and a time duration. A session data rate is the amount of bandwidth on terms of bits per second that it needs to be satisfactorily performed. For instance, the data rate needed to hold a videoconference is way larger than the data rate of a voice call.

In addition, we need to capture the value, the utility that has for a user to be able to perform such kind of communication, as this utility will determine the price the user is willing to pay to the WSP for the service provided. In order to measure a user's valuation, we introduce the concept of bit value, which is the amount of money he/she is willing to pay for every transferred bit. This bit value is fixed per session, assuming all session bits have the same value. Therefore, when a user decides to initiate a connection, he/she will choose the WSP with the lowest price per transferred bit, and if such price is higher than the user's bit value, the session won't take place, as it is not worth it for the user.

Besides the bit value and the data rate, we now explain the dynamics of a session generation. When defining the session generation we have to define two parameters: (1) the time at which the session is created, or equivalently, the session starting time and (2) its duration, that relates to the finish time.

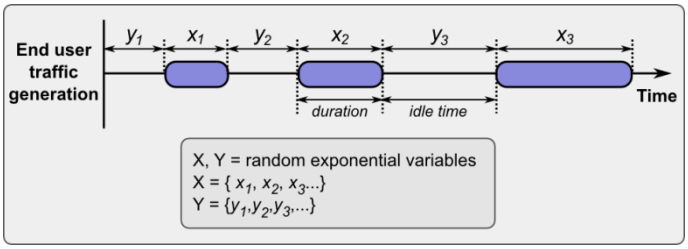


Figure 12: Traffic generation

An alternating random process generates variations for these two variables (Figure 12). From the first value we set the duration of the session. Once the session finishes, we get the time the connection remains idle from the second value, that is, the time until the next session starts. Afterwards, in a repeated fashion, we obtain the session's duration from the first variable and so on. Both variables are exponentially randomly generated.

Whenever a session is generated, the end user performs a service request and receives a set of prices as answer, one price from each service provider. As the end user can choose freely among the service providers, he/she will choose the one that maximizes his/her own benefits, that is, the lowest priced offer. As said before, the user will only proceed to connect to the selected WSP if the price offer is lower than its valuation, obtaining profit from the transaction. The benefit obtained by end user j when selecting a service provider i to serve session n can be calculated as:

$$P_j(p_i, d_n, c_n) = (p_i - b_{j,n}) * d_n * c_n$$

Where p_i is the price set by the service provider *i*, $b_{j,n}$ is the bit value for session *n* for end user *i*, and d_n, c_n are the duration and data rate of the session respectively.

3.1.4 Service Broker

The last agent of the model is introduced to simplify the programming of the simulation model. The service broker is used as a bridge between the end users and service providers before the service providers' selection by the end users. The end users obtain a list of all the available service providers by consulting the service broker, it is like the common channel where all the service providers can announce the price for their services and whether they have enough bandwidth to perform a communication or not. Therefore the service broker is more a logic agent than a real one, and this is why it does not obtain revenue from the transactions it performs

3.2 Two-level market

Spectrum demand varies as a function of the network load, hence, not only the spectrum management must be considered, but also the number of the end users' transmissions. The latter defines a two- level market: on the first level service providers get a hold of spectrum access rights; on the second level, spectrum access is used to provide services to end users.

3.2.1 Spectrum Trading

As explained before the spectrum rights sold by the spectrum broker last a DSA cycle, so at this level the transactions happen at the beginning of every cycle in a periodic way. The whole process can be described step by step as follows:

1. At the beginning of a new cycle, the service providers make a prevision of their spectrum needs for the next cycle. They communicate the spectrum broker the required quantity of bandwidth and wait for its decision.
2. Once the spectrum broker receives all the spectrum requirements from the service providers, it performs the allocation algorithm. This allocation algorithm will determine the spectrum share of every service provider.
3. The spectrum broker retransmits its decisions to the service providers. This decision will include the spectrum share of the service provider as well as the price it has to pay for it.
4. This allocation is valid till the end of the DSA cycle; afterwards all the steps are repeated.

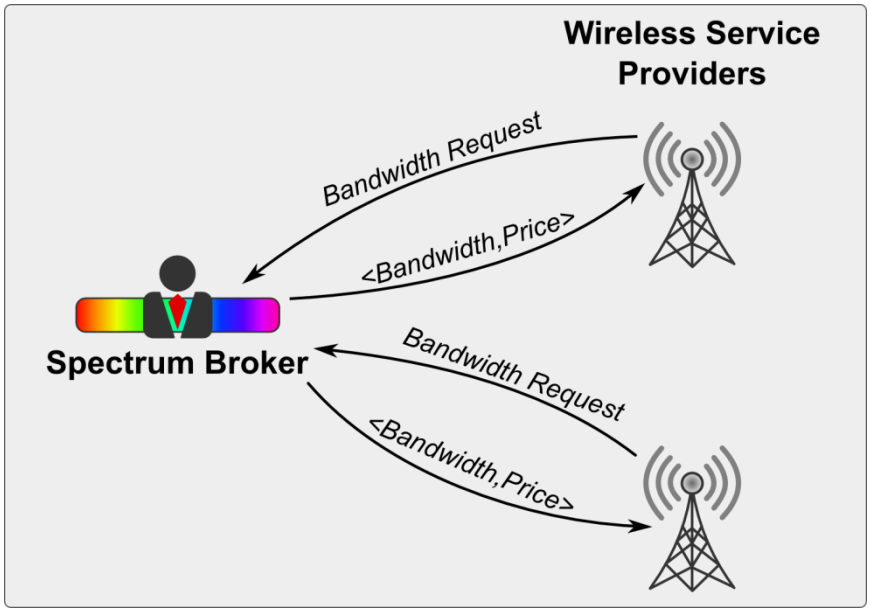


Figure 13: Spectrum Allocation workflow

3.2.2 Service provisioning

Service provisioning refers to all the necessary steps since an end user generates the need to communicate (generates a session), until it is served. This process is initiated by a stochastic event, which is the session generation, however all the further steps are completely deterministic. Service provisioning can be divided in two phases: firstly, the end user needs to obtain the list of service providers that can serve its session and the price they are going to charge for such service. Secondly, the session is served by a selected service provider. Step by step the whole process is:

1. The end user generates a session, which is defined by the necessary data rate and its duration.
2. The end user creates a service request where it specifies the data rate needed. This service request is sent to the service broker.
3. The service broker broadcasts the service request to all the service providers and waits for their offers.
4. Every service provider checks if there is enough bandwidth to provide the data rate needed by the request received. It creates an offer based on the free bandwidth at the moment, establishing the price per bit transferred. This offer is sent back to the service broker as answer.
5. The service broker packs all the offers received and forward them to the end user.
6. The end user selects the best service provider based on the price.

If this price is higher than the price the end user is willing to pay, then the session will be cancelled. Otherwise:

7. The end user initiates the connection with the selected service provider.
8. The session is served.

9. After the session finishes, the service provider sends the bill to the end user based on the price offered, the duration and the required data rate of the session.

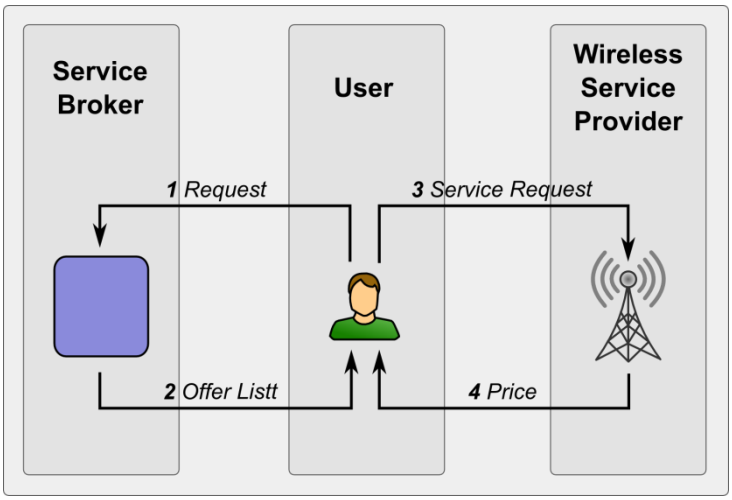


Figure 14: Service Request and provisioning simplified workflow

4 A Model with Learning Agents

The role played by the model's decision mechanism that every agent uses is very important for market modelling. This mechanism provides reality and accuracy to the model, as it aims to describe via an algorithm how the agent actually behaves, a situation that is not always obvious. When defining end-users' behaviour, their choice is either to accept or not a list of offers delivered by the service providers, and then select one of them. A selfish user will select, in a logical way, the offer that provides him with the best revenue; such is the mechanism we have implemented. However for the service provider, decisions do not have such an immediate outcome; in other words, it is not all clear which strategies the service provider will follow. The decision must align itself with maximizing the revenue obtained by the service provider, but knowing what is the best decision beforehand is a difficult task.

There are at least two basic ways of maximizing an outcome. The first way implies knowing how the decisions affect the outcome, then estimating the outcome produced by every strategy, and finally selecting the strategy that yields the best outcome. If such knowledge is not available, we can use a learning algorithm to infer what the outcomes will be for every strategy. We expect that, as time goes by, the service provider will be able to make more accurate decisions.

We have chosen the second approach here as it provides us with a fundamental advantage: when we define a learning algorithm where the implied knowledge of every considered variable is null -i.e. it is not known how the variable affects the revenue, if it should take low or high values, etc.-, it is much easier to modify the variables used by the service providers to make decisions, as we can add new variables without re-writing the algorithm. However when using this approach we have to be aware that the simulations may be inaccurate during the first cycles of the simulation. Therefore longer simulations are needed, to allow the service provider to learn its correct behaviour.

Our goal is to define a learning algorithm capable of maximizing the provider's utility over a subset of values of chosen decision variables. The algorithm has to look for the optimal vector of values of each variable that maximizes the objective function. The major challenge when defining the algorithm is that the outcome not only depends on the values a service provider selects, but also on the other providers' strategies, which are unknown to the provider.

4.1.1 What is a learning agent?

A learning agent is an agent capable of learning from previous experience. *The idea behind learning is that percepts should be used not only for acting, but also for improving the agent's ability to act in the future. Learning takes place as the agent observes its interactions with the*

world and its own decision-making processes [26]. Therefore a learning agent can choose the strategy that is supposed to generate the highest revenue.

Different methods to model learning are listed below:

1. ***Evolutionary*** approaches are based on a representation of the strategies as individuals, testing them and allowing the survival of strategies with higher revenues. This approach supposes no prior information; the fitness of every strategy is calculated after observing the outcome obtained when it is chosen.
2. ***Reinforcement learning*** defines the attraction of a strategy as the likelihood that it will be chosen as the one played in the next iteration. This attraction is updated every time the strategy is played based on the outcome produced in previous iterations, rewarding profitable strategies with higher attractions, and punishing the undesired strategies. An additional feature that can be implemented on this strategy is that strategies can "spill over" to similar ones (they can update the attractions of the similar strategies). Again in reinforcement algorithms, the available information for the players is their own revenue.
3. In ***belief learning***, the player is supposed to know the strategies chosen by the other players in previous cycles. With this information the agent can estimate the probability of every opponent's strategy, based on the frequency each strategy was chosen in the past. Once the strategy others will play is estimated, the player selects the strategy that produces higher payoff more frequently.
4. ***Experience-weight attractions (EWA) learning*** updates the attractions based not only on the actual payoff received, but also on the forgone payoff, that is, the payoff that the agent would have had obtained if it had chosen another strategy. EWA learning is a mix between belief learning and reinforcement learning; as a consequence the information needed is the same as in the previous one.
5. In ***anticipatory learning*** or ***sophistication***, players use information about the others payoffs to predict which will be the adversaries' strategies in the future. Consequently the players must have information about the whole payoff table.
6. Other possibility, when players know the others' revenues, is to ***imitate*** the most successful player. This is the strategy followed on imitation learning.
7. ***Learning direction theory*** is based on redirecting the current strategy to the optimal one, calculated a posteriori. Thus, the agent must know about which is its best response afterwards.
8. The last one is based on ***rules***. The players have rules to choose the best strategies, and they end up learning which rules they have to follow to obtain the highest payoff. In this case, information about the influence of the strategies is needed to be able to define the rules that will lead the algorithm. [27]

Therefore we have to consider which information is more likely to be available to the players to select the most convenient learning algorithm. The information that is most likely to be available for a service provider is the revenue obtained after selecting a concrete strategy, once it is already played. However it probably will not know which strategy or revenue have obtained the other service providers. Also, due to the randomness of the situation, it will

neither know the forgone revenue, that is, the revenue that it could have obtained in case of playing a different strategy.

Thus, the most sophisticated methods which take into account more information will not be applicable, like EWA learning, belief or sophisticated learning. Neither imitation is a valid approach, as the others strategies are not known. Finally, to boost reusability, we have decided not using rule-based learning, as it requires defining a set of rules for each parameter, depending on their influence on the environment, neither directional learning, that imposes an order over the variables. As a result, the two available algorithms that are finally selected are reinforcement learning and evolutionary learning.

These learning algorithms can be reinterpreted as some key characteristics of our model do not exactly fit their assumptions:

1. Once a player selects a strategy, their revenue is not deterministic, but a random value, as it depends on the users' traffic necessities, which is randomly generated.
2. There is no finite number of strategies but a infinite number (range) of valid values for every variable the learning is based on.

4.1.2 Genetic Algorithm

A genetic algorithm is a concrete implementation of the evolutionary approach. It is based on reproducing the natural evolution, where the strategies are elements of the population, which compete to survive. The strategies are defined by its genes, which in this case are the set of variables the service provider can decide on. A genetic algorithm is defined by the following cycle: evaluation, selection, reproduction and mutation. This cycle is repeated every learning cycle, evaluating the population elements through an attribute called *fitness* which measures its likelihood to survive: the higher the *fitness* of a strategy, the more likely it is to survive to the next cycle. As during the cycles, only the best strategies survive, after some time, the learning algorithm, will have reached to a point where the population is comprised only by a set of strategies that lead to the better outcomes.

Next subsections explain each phase in the genetic cycle.

4.1.3 Evaluation

Every strategy must be evaluated to calculate its *fitness*. In this case the *fitness* is represented by the revenue obtained by every strategy. Thus, each strategy must be evaluated to set its *fitness*, i.e. the strategy is played during a market cycle. This approach has two main drawbacks:

1. The revenue obtained by a strategy is not fixed, as it depends on the others strategies, as well as, the end user session generation process.

2. All the strategies must be tested. It is a time consuming phase during which the service provider will be testing undesired strategies; this may lead to a learning process slow down.

To alleviate the first drawback every strategy could be tested multiple times, setting the *fitness* as the obtained average revenue. However this solution reinforces the second drawback, increasing the number of strategies that must be tested. Our solution tries to leverage both, by averaging the *fitness* of a strategy if it survives with the *fitness* obtained in previous cycles:

$$fitness(i) = \alpha * fitness(i-1) + (1-\alpha) * revenue(i)$$

Where $fitness(i)$ is the new fitness and $fitness(i-1)$ is the previous fitness; $revenue(i)$ is the revenue obtained in the current cycle and α is the fading parameter.

This way, the fitness will tend to the average revenue. Unfortunately, because of the genetic process, the individuals end up mutating and reproducing, generating new individuals with unknown fitness.

4.1.4 Selection

In the selection phase the strategies that will survive are chosen. The selection criterion is based on each individual's *fitness*, i.e. individuals with higher *fitness* are more likely to survive. The probability of surviving is proportional to the *fitness* parameter; like a roulette wheel with elements with different probabilities, it selects N new elements for generating the next population. The number of strategies selected is the size of the population, and the same strategy may be selected multiple times, so strategies more fit are selected more times that strategies that are not that desirable. The strategies that are not selected do not survive, and are erased.

4.1.5 Crossover

After the selection process the selected individuals are paired up and combined on the crossover phase. On the crossover, the genes of the 2 individuals are interchanged from a splitting point selected randomly. That is, the crossover mechanism will interexchange some variables between the two strategies selected, e.g. if there are two variables involved in the learning process, bandwidth requested to the spectrum broker and price charged to the end user, a possible new strategy will be choosing the price of one good strategy and the bandwidth of another good one. This phase is not a strict step in the learning cycle, but only performed occasionally, based on the parameter *crossover probability* of the algorithm.

4.1.6 Mutation

In the same way, depending on the *mutation probability*, the individuals will mutate. When there is a mutation, one gene is chosen randomly and it is mutated. On our project, as the

genes are variables with a range of possible values, the mutation consists on smalls variations on the value of the selected variable:

$$v' = v * \delta * range * uniform(-1,1)$$

Where v' is the mutated variable value, v the value previously to the mutation phase, δ is the proportion of the range the variable is allowed to vary and $range$ is that range. $uniform(-1,1)$ generates a random number distributed uniformly between -1 and 1.

Once a new population is generated by these four steps, the process starts again, evaluating the just generated population and on. As the time goes by, the only survival strategies will be the ones that produce more revenues; therefore the strategies played by the service provider would be wise choices.

4.1.7 Reinforcement

Classical reinforcement algorithms are based on a finite set of strategies, each of which with an attractiveness value, which defines its likelihood to be chosen as the next strategy to play. To be able to implement a reinforcement algorithm, we have to discretize the range of values a concrete variable can take, so that a finite set of strategies is defined. Thus, each variable is divided in a number of chunks, which is established by the *precision* parameter of the algorithm programmed. As a consequence $precision^{number\ of\ variables}$ strategies are generated.

Evidently, these strategies have unknown expected payoffs, so the first step in this learning process is to evaluate each to initialise its *attraction*.

Then at every new cycle a strategy is chosen randomly, but the probability to choose each strategy is proportional to its *attraction*. Thus, we have defined the *attraction* of each strategy as follows:

$$A = a * mean\_payoff + b * max\_payoff$$

Where *a* and *b* are the coefficients of the mean and maximum payoff obtained by the strategy.

That way when a strategy is selected, its *attraction* is updated through the mean: if the payoff is better than expected then the *attraction* increases, otherwise, it decreases.

5 Simulation Environment

The following section describes the model programmed to simulate the market model described in section 0. The main goal of this implementation model is not only to represent faithfully the market model, but to be easily modified and customised to be adapted to new market structures, relationships and mechanisms.

As implementation platform, we have opted for OMNeT++, which is described below. Next subsections detail each element in the simulation model and its configurable parameters.

5.1.1 OMNeT++

OMNeT++ is an extensible, modular, component-based C++ simulation library and framework. Even though OMNeT++ is mainly used for the simulation of networks, its flexibility makes it possible to use it for our project. It is based on modules, channels and messages. Modules represent the entities of the system, which can be subdivided in submodules to cluster their functionalities and simplify the programming and design. To communicate among them, modules make use of the messages. Messages can be defined as desired to contain all the parameters needed for each type of communication. To deliver such messages, modules can send them directly to their destination module or use the channels that connect modules among them.

Each module is defined using a “ned” file, which describes the set of parameters this module needs to work correctly. Later, these parameters can be specified in a configuration file, allowing it to modify them for every simulation, without the need of code recompilation, simplifying the different analysis of the simulation. Moreover, the value of a parameter can be not only a concrete value, but a stochastic function that will make the parameter change during the simulation, every time it is obtained.

The main reasons for choosing OMNeT++ as implementation framework are its modularity, which provides the model with the desired reusability; its configurability obtained through a simple configuration file, with plenty of possibilities; and last, but not least, the prior knowledge of the tool, which eases the learning phase, speeding up the implementation phase of the project.

5.1.2 Module Description

The simulation model must represent the market model proposed. Furthermore it must make it possible to modify any model assumption easily. OMNeT++ gives us two powerful concepts

to achieve this: the modules and the messages. Every agent on the market model will be represented by an OMNeT++ module, making it easy to modify them separately, and also replicate them as desired. Besides generic communications protocols will be defined to make easier redefine market mechanisms. The global model defined is:

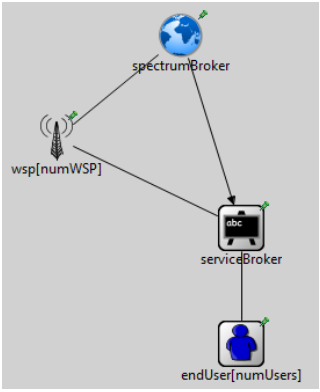


Figure 15: Simulation Model

The number of Wireless Service Providers and End Users are set through the simulation configuration file. Being amongst the most complex modules, we have decided to subdivide the WSP module and the EndUser module in submodules.

5.1.3 SpectrumBroker module

The SpectrumBroker module that represents the Spectrum Broker on the market model has three main tasks to perform: controlling the timing on the simulation, spectrum allocation and, finally, setting the price for the spectrum allocated.

5.1.3.1 *Simulation time*

The spectrum allocation happens periodically in cycles, therefore the spectrum broker as well as the service providers must be aware of the simulation time to know when a cycle starts and finishes. To avoid having all this modules being aware of the time, we give to the SpectrumBroker the responsibility of taking care of it. Moreover the length of the simulation is measured in the number of cycles the simulation runs, so the SpectrumBroker is also in charge of signaling to the EndUsers about the end of the simulation. EndUsers must be told when the simulation finishes so they can stop generating new sessions.

Cycle generation is performed by self scheduled messages, indicating the end of every cycle; thus, the SpectrumBroker informs the wireless service providers about the beginning of a new cycle, and the new allocation is performed.

When all the cycles of the simulation are generated, the simulation finishes, the SpectrumBroker sends a message to the ServiceBroker, who will forward this message to the EndUsers.

There are two NED parameters related with the simulation time that are:

- *numCycles*: Determines the number of cycles the simulation will run.
- *marketLifeCycle*: Determines the length of every cycle.

5.1.3.2 *Bandwidth Allocation*

In regard to the allocation algorithm, as explained in chapter 3, two different allocation algorithms have been considered:

- Proportional algorithm, which allocates the bandwidth proportionally to the requested bandwidth.
- Fairness algorithm, which allocates the bandwidth in a fairer way, trying to avoid service providers request for more bandwidth than required.

The allocation algorithm is selected by a NED parameter ("allocationAlgorithm"), to allow new algorithms to be implemented and integrated easily on the model.

5.1.3.3 *Pricing*

Finally, when the spectrum share of all the service providers has been decided the price is calculated by the function calculateUnitCost, following the formula explained in chapter 3:

$$c(\boldsymbol{B}) = x + y\left(\sum_j b_j\right)^{\tau}$$

Where x and y are non-negative constants and $\tau \geq$ and $\mathbf{B}=[b_1,b_2\ldots]$ is the vector of allocated spectrum quantities. Every of these constants are set by a NED parameter: x, y and tau.

5.1.4 WSP module

The WSP stands for Wireless Service Provider; this module represents the service provider agent in the market model. On this module is critic the decision taking, which is managed by a learning agent as explained in chapter 4. The strategies followed will influence on the revenue obtained by the agent; therefore they must maximize the agent revenue to ensure the agent is an accurate representation of the market agent. Apart from the learning module we have the controller module, which manages the resources and communicates with the other modules.

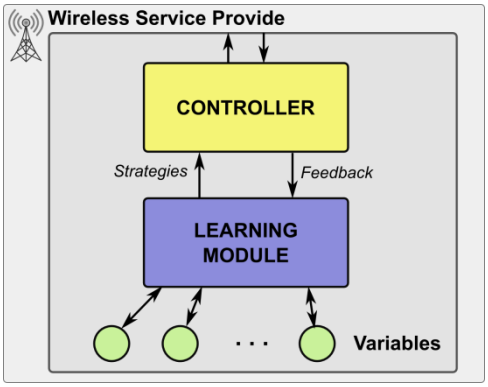


Figure 16: WSP module composition

5.1.4.1 *WSPController*

The WSPController is the WSP main submodule. It manages the communication with the other modules, and determines whether a new user can be served or not. Also it is the controller of the learning module and stores the statistics of the module, such as the utility obtained every cycle, the bandwidth utilization and so on.

The communication with the SpectrumBroker determines the bandwidth available in every cycle, as well as the initial expense for purchasing the bandwidth. Every DSA cycle is also considered a learning cycle, so the controller must provide the corresponding feedback to the learning module. This feedback is informing about the utility obtained in the previous cycle.

As regards to the ServiceBroker, it sends the service requests to the WSP. For every request the WSP receives, the controller checks if it has enough bandwidth to serve it or not. In case it has enough bandwidth a positive offer is created by the function createOffer. This offer will have information of the cost, which is calculated by the function getCost, using the function proposed on chapter 3:

$$p = (Price_{\min} - Price_{max}) * \left(\frac{freeCapacity}{maxCapacity}\right)^{\tau} + Price_{max}$$

The parameters of this function are the NED parameters: minCost, maxCost and tau.

Finally, when the offer is accepted by the EndUser, he/she will initiate a connection with the WSP. When the WSP receives the message indicating the beginning of a connection, it has to update the quantity of available bandwidth by subtracting the quantity that will be used by this connection. A message will be self scheduled, so when the connection finishes the bandwidth is restored. Also when the connection finishes the WSPController generates a message to the EndUser informing about the final cost of the connection.

The connections with the EndUsers make the available bandwidth change; these fluctuations are stored for later analysis.

5.1.4.2 *WSPLearning*

The WSPLearning is an abstract module that is redefined by two concrete modules, the Genetic module and the Reinforcement module. Each of these modules provides the same functionality but each use a different learning algorithm. The learning module works as follows, every market cycle a new strategy is chosen by the learning module, based on the feedback obtained on the previous cycles. A strategy is characterized by concrete values for every variable that is included in the learning module; at the moment the only variable is the 'requested bandwidth'. The feedback received is the utility obtained by the strategy, which is recorded at the end of every cycle, allowing the module to learn through the experience. The algorithms used are explained with further detail on chapter 4.

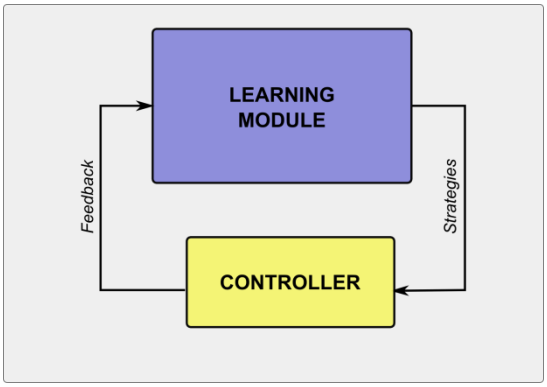


Figure 17: Learning cycle

5.1.5 ServiceBroker module

The ServiceBroker is the representation of the Service Broker from the market model. This is the simplest module, as its main function is just forwarding the service request from the end users to the WSPs. To forward the message it uses the function sendAvailable, to allow to future implementations the possibility to restrict the WSP that are available to a certain user. It also has to forward the signaling of the end of simulation sent by the SpectrumBroker to the end users, so no new sessions will be generated.

5.1.6 EndUser module

The EndUser module is the module in the simulation model designed to capture the behavior of the end-user. The module's main function is traffic generation. The EndUser module can be more easily understood divided into its three different submodules: (1) the SessionGenerator, (2) the EndUserController and (3) the Scoreboard. The SessionGenerator is responsible for the generation of sessions; a session is every communication the end-user wants to carry out. The EndUserController performs the service requests and selects the best WSP for each session. Finally the Scoreboard stores the data of the utility obtained by each session. Figure 18 shows the internal composition of the module.

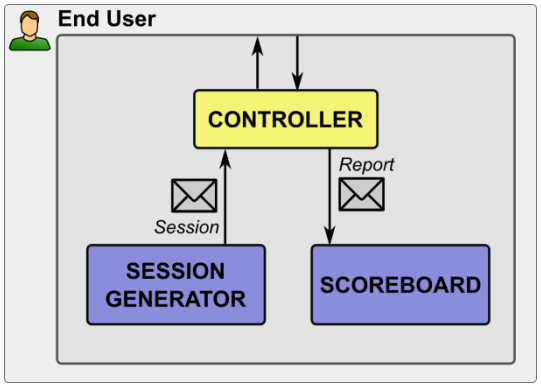


Figure 18: EndUser module composition

For the internal communications between the submodules, two types of messages have been defined:

- *SessionParameters*: contains the information about every session generated by the sessionGenerator; i.e. the data rate needed and its duration. This type of message is created by the SessionGenerator and sent to the EndUserController.
- *TransmitterReport*: stores relevant details of every session that it is finally served, like the session parameters and the cost of the communication. It is created by the EndUserController and sent to the Scoreboard.

5.1.6.1 *SessionGenerator*

The SessionGenerator is who generates the EndUser sessions. There are three important parameters in relation with the session generation: when the session is created and the two parameters that define the session: its datarate and its duration. Sessions are modeled as models sessions as alternating processes. That means that there are two different random variable which values are obtained in an alternate way, as shown in Figure 19, which define the status of the end user: active (transmitting data) or idle there is no data to transmit). The time spent on each state is defined by a random variable: duration for the active state and idle time for the idle state. We consider that the process that defines the duration of every session is not likely to change in the whole simulation. However, it is very likely that the time between sessions changes depending on the moment of the day, generating more sessions in the peak hours than during night, for instance. To allow this variation on the traffic intensity, the time spent on the idle status must be defined not only by a single random variable, but it must be possible to redefine it during the simulation. That is why we define a stretching factor that will affect the initial variable defined to characterize the process. This stretching factor is applied for slots of time of a defined duration (dT), and different factors affect to every slot of the day.

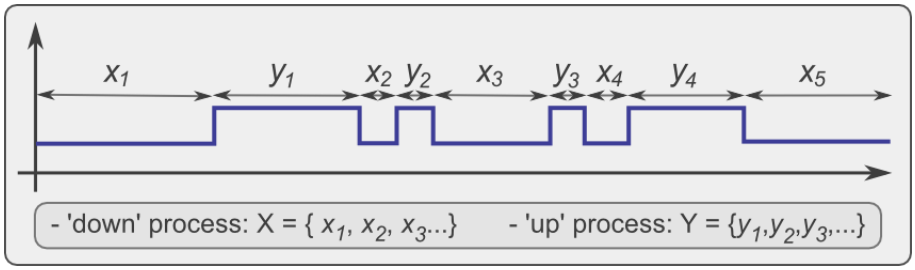
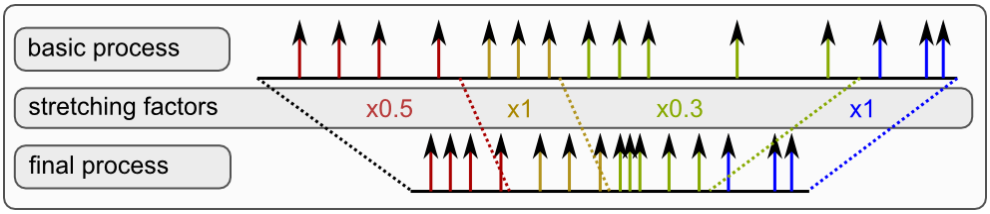


Figure 19: Alternating process



For instance, suppose the day is divided in 4 time slots, each of 6 hours of duration. The first hours (0.00 to 6.00) of the day will have low traffic, while the late ones will have much more traffic (18.00 to 0.00); as a result the coefficient of the first hours will be bigger than the coefficient of the late hours (more time between sessions will produce less sessions).

Stretching factor 1 (from 00.00 to 06.00)	Stretching factor 2 (from 06.00 to 12.00)	Stretching factor 3 (from 12.00 to 18.00)	Stretching factor 4 (from 18.00 to 24.00)
100	7	4	1.7

Table 2 : Stretching factor table

If we choose as the 'up' process an exponential variable with mean 6 minutes, and for the 'down' process another exponential variable with mean 10 minutes, that will be multiplied later by the stretching factors defined on Table 2, the traffic generated by the user would be something like Figure 20.

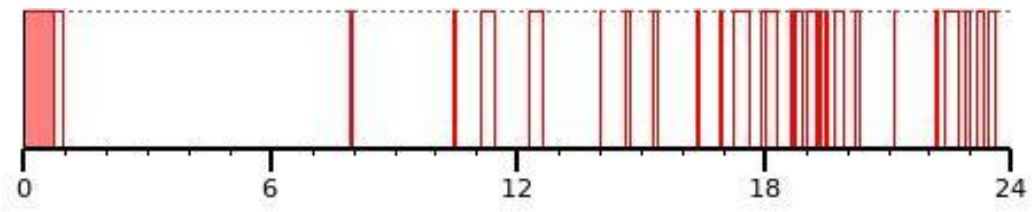


Figure 20 : Example of traffic generated by an user.

When a session is generated, its parameters are encapsulated in a message of type SessionParameters and it is sent to the EndUserController. A SessionParameters message contains information about the duration and datarate needed by the session just created.

5.1.6.2 *EndUserController*

The EndUserController is the responsible of the communication with the other modules; it also is the one that selects the WSP that will serve the session and whether a session will be served or rejected.

The selection of WSP is performed by the function selectWSP, where it selects the WSP with best score. The score is calculated by the function evaluateWSP, which mainly returns the

bitValue of the EndUser minus the price of the offer, but could easily be changed for something more complex. If the score is greater than 0, then the session is not rejected, as it generates utility. Then the EndUserController initiates the connection with the selected WSP and the session is served.

The last task of the controller is generating the TransmitterReport. The transmitterReport is a message that is sent to the Scoreboard every time a connection finishes. It contains information about the session that has just finished, like the cost, bitValue, etc… that way the Scoreboard can store statistics about the utility obtained.

5.1.6.3 *Scoreboard*

The initial purpose of this module was to store information about the connections carried out in the past, so this information could be used for future WSP selection. However its functionality has been simplified to basic data collection, for further analysis in the model. Concretely it records data about the utility achieved in each transaction as well as the total utility obtained by the end of the simulation.

5.1.7 Transaction Protocols

The transactions defined on the model are the spectrum allocation, the service requesting and the service provisioning.

5.1.8 Spectrum Allocation

The transaction protocol defined to characterize the spectrum allocation process has been defined as a three way handshake, when just two messages would be enough. The reason of choosing a tree way protocol is that it can be adapted to different transaction mechanisms. The basic mechanism is the depicted in Figure 21: every new cycle the SpectrumBroker notifies the WSP, then the WSP request a certain amount of bandwidth and with all the requests the SpectrumBroker realizes the allocation and sends its decision to the WSP. To make the protocol more robust, the SpectrumBroker programs a temporizer that will trigger the allocation algorithm regardless not all the bandwidth requests are received. This prevents the SpectrumBroker to wait indefinitely if any bandwidth request is lost. The trigger is a special message indicating the time to live (TTL) of the waiting period has finished. Apart from sending the new cycle alerts to the WSPs and programming the TTL, every new cycle the SpectrumBroker must schedule the next cycle. The whole protocol is shown in 21.

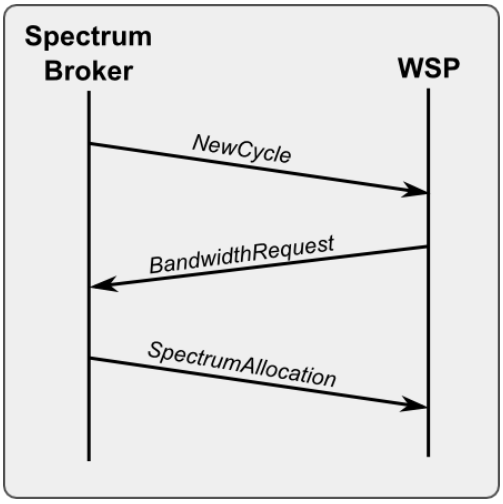


Figure 21 : Tree way protocol

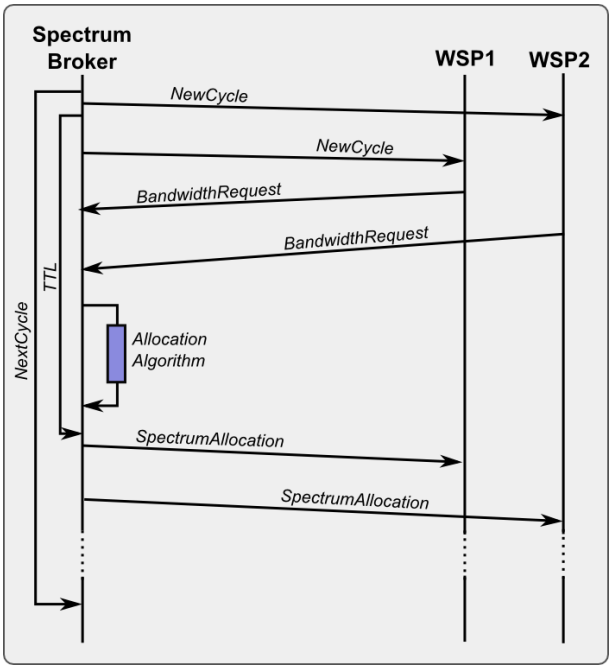


Figure 22 : Whole Spectrum Allocation Transaction

Two different types of messages have been defined for this transaction:

- *WSPBid*: is the message with the bandwidth request.
- *SpectrumBrokerAllocation*: is the message with the final decision about the spectrum allocation. It contains the size of spectrum share for a concrete WSP and its price.

As the TTL and the new cycle message contain no additional information, no more message definitions are necessary.

5.1.9 Service Request

The service provisioning is divided in two stages; on the first one the end user obtains information about the available service providers. The second stage only takes part if the user

is satisfied with the offer of one of the available service providers. On the second stage the end user connects to the selected service provider and its session is served.

Service request refers to the first stage. This process starts with a new session generated by the EndUser. The EndUser creates a service request with the required data rate and forwards it to the ServiceBroker. The ServiceBroker is the responsible of forwarding this request to all the WSPs available for the EndUser and waits for their offers. When the WSPs receive the request, have to check whether they have enough bandwidth to serve the session or not. They send an offer to the ServiceBroker indicating if they can serve the session or not, and with the price information in affirmative case. The ServiceBroker packs all the affirmative offers (the offers from the WSP that have enough bandwidth) and forwards them to the EndUser. With all the offers the EndUser can take the decision whether or not it is worth it to serve the session. As in the allocation process, the ServiceBroker programs a TTL to ensure at least the offers received till that moment are sent regardless some WSPs' offers are lost.

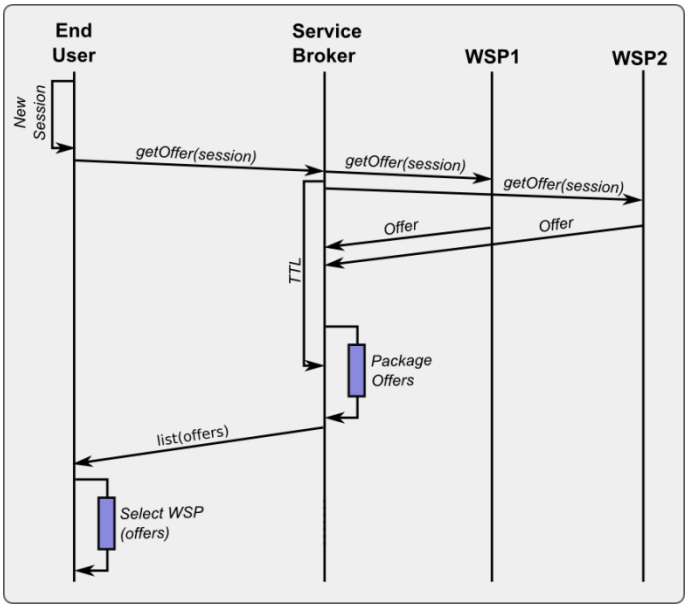


Figure 23: Service Request workflow

5.1.10 Service Provisioning

Service provisioning refers to the process where the session is finally served by the service provider. When a user connects to a service provider, it implies the end user has some channel allocated to perform the communication. This channel can be time slots, frequency slots or code slots, but serving the end user supposes a reduction of the effective bandwidth that the service provider has available for other new users.

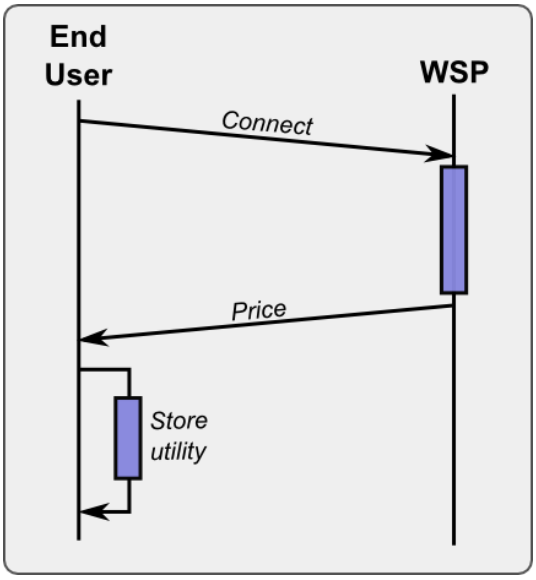


Figure 24: Service provisioning workflow

6 Simulation Scenarios

On this section, we will describe the scenarios that will be analysed using the simulation model described in the previous chapters. These scenarios are based in 9 basic scenarios that differ among them in the number of agents that play a part in the simulation. These 9 scenarios will be used to break down and analyse the effects of different parameters depending on the market characteristics.

6.1.1 Basic scenarios

The 9 basic scenarios configure the market characteristics in two aspects: the competition among service providers and the demand level. As regards to the competition among service providers, we have chosen three options: (1) no competition, with only one WSP, i.e. a monopoly; (2) a duopoly and (3) an oligopoly with 5 WSP where their actions influence in the others gains.

On the other hand, to modify the demand, we vary the number of users, so that their total bandwidth requirements are: (1) above the available bandwidth, (2) less than the available bandwidth, but creating a loaded network; and (3) a market with low demand, where the required bandwidth is way lower than the available. The combination of the former 3 options with these 3 ones make up to 9 basic scenarios:

	1 WSP	**2 WSP**	**5 WSP**	
10 users	Scenario A	Scenario B	Scenario C	Light load
100 users	Scenario D	Scenario E	Scenario F	Medium load
200 users	Scenario G	Scenario H	Scenario I	High load
	Monopoly	Duopoly	Oligopoly	

Table 3: Scenarios defined

To choose the correct number of users for each of the above situations, we have to calculate which will be the network load of every user and how it will influence the system. First of all we need to know which will be the total bandwidth available from the spectrum broker to lease to the WSPs. In our case we have chosen a bandwidth of 25 MHz, using as example the bandwidth unleashed by the FCC in May 2010 for mobile broadband service[1]. This bandwidth in terms of hertz will be transformed into the bits per second it can carry using the spectral

[1] News release: http://hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-298308A1.pdf

efficiency of the WSPs. Concretely, it will be in our model 2.88 as it is the efficiency of a transmission using HSDPA. Therefore, the global capacity of the system will be:

$$Available_{datarate} = k * Available_{bandwidt\ h} = 25Mhz * 2.88 = 72\ MBps$$

On the other hand we have the traffic generated by every end user. An end user will be active a certain percentage of the simulation time depending on its session duration and the idle time between sessions. We have decided for users active the 25% of the time in average:

$$Transmission_{percentage} = \frac{session_{duration}}{session_{duration} + idle_{time}} = \frac{10}{30 + 10} = 25\%$$

The traffic generated during this time depends on the data rate of the sessions, which are selected from a random uniform variable among 0 and 3.6 Mbps. 3.6 Mbps corresponds with the maximum data rate capacity provided with HSDPA category 6. Therefore the mean network load generated by a user will be:

$$UserTraffic_{load} = Transmission_{percentage} * datarate = 0.25 * \frac{3.6}{2} = 4.5\ Mbps$$

If we compare the traffic load per user with the global capacity of the system, it will provide us the number of users we need for each market situation. Concretely we have chosen 10, 100 and 200 users to recreate the 3 situations, which make a network occupancy when all the users are served of:

$$Network_{load} = \#users * Transmission_{percentage} * datarate = \begin{Bmatrix} 10 \\ 100 \\ 200 \end{Bmatrix} * 0.25 * \frac{3.6}{2} = \begin{matrix} 4.5\ Mbps \\ 45\ Mbps \\ 90\ Mbps \end{matrix}$$

$$Occupancy = \frac{Network_{load}}{Available_{datarate}} = \begin{matrix} 6{,}2\% \\ 62{,}5\% \\ 125\% \rightarrow 100\% \end{matrix}$$

6.1.2 Parameters Study

We will perform 4 different sets of parameters studies to analyse how they influence the system as well as to try to find their optimal values for each of the market conditions mentioned above. We will focus first on the parameters of the 2 learning algorithms, trying to find the optimal values that will provide higher profits to the WSPs. Afterwards, we will analyse the effects of the parameters of the two pricing functions: (1) the price charged to the WSP for a piece of spectrum and (2) the price charged to the users for their service.

6.1.3 Learning Algorithms Analysis

The learning algorithms programmed have different parameters that must be calibrated in order to obtain an optimal result. As the objective of the learning algorithms is to emulate the behaviour of the WSPs, maximizing their utility, their parameters must be set in order to obtain the same objective.

Each algorithm must be calibrated in a multidimensional space with as many dimensions as parameters take part. So in the first simulations we will sample this space in all the dimensions, to analyse which parameters have a higher impact on the WSP's profit. Later, we will sample each parameter in higher detail to obtain more accurate values. The order of this second step of the parameters calibration will be starting from the ones with more influence and finishing with the parameters with less influence.

6.1.3.1 *Genetic algorithm*

The parameters in the genetic algorithm are the following:

- *Population size*: it determines the number of strategies that are being considered simultaneously. Higher values suppose a slower learning rate as the learning cycles become longer (a learning cycle lasts as many market cycles as elements are in the population).
- *Mutation probability*: varies among 0 and 1. It represents the probability of a strategy modification. Higher values imply strategies that are constantly changing and it makes the learning process more unpredictable. Combined with a high value of the parameter increment, it may also slow down the learning rate, as it changes well-known strategies for new ones with unknown results.
- *Increment*: this parameter specifies how much the strategy can change when it is selected to mutate. It can vary among 0 and 1.
- *Fading*: as the results obtained from each strategy vary each time it is executed – due to the randomness of the simulation, and others elections-, their fitness is obtained through a faded mean:

  $$fitness(i) = fading * fitness(i-1) + (1 - fading) * revenue(i)$$

  Therefore, high values of fading suppose slow mean convergence, while high values imply a fast adapting averaged fitness. It can vary among 0 and 1.
- *Crossover probability*: in this particular case, where the learning algorithm works with only one gene, the crossover probability must be seen as the probability to reset the fitness of a strategy, as it does not modify the current strategy. It can vary among 0 and 1.
- *Minimum fitness*: Selects the value of fitness that will be assigned to the strategies with no profit or negative profit. Choosing a high value for this parameter will imply the reselection of the current strategy in future cycles despite it has not provided profit in the previous executions.

6.1.3.2 *Reinforcement algorithm*

The reinforcement algorithm only considers three parameters. They are:

- *Precision*: each variable of the learning algorithm is subdivided in as many slots as the parameter precision indicates. Higher values imply more precise values, but a more complex learning process.

- *Max* and *mean coefficients*: the attractiveness of each strategy will vary accordingly to the profit obtained by it. As the profit varies each time a strategy is chosen, due to the randomness of the simulation, each strategy has an average profit and a maximum profit. Each of these values has an importance in the attractiveness of the strategy, measured by the max and mean coefficients. If the max coefficient is way larger than the mean one, it implies that strategies with high maximum profit are favoured instead of others with higher average profit but lower maximum profit.

6.1.4 Service Pricing Analysis

The charge for each service is determined by the service pricing function. This price is a function of the available bandwidth of the service provider, and it is shaped with a set of parameters. On this analysis we will analyse, how do this parameters influence on the benefits of the different agents of the model, as well as in terms of bandwidth efficiency. The parameters in question are:

- *Maximum Price*: establishes the maximum price, which will be charged to the end user when there is almost no available bandwidth.
- *Minimum Price*: establishes the minimum price, which will be charged to the end user when all the bandwidth is available.
- *Tau*: establishes the curvature of the function. Higher values of tau imply that the price rapidly increases to the maximum price. Low values of tau (less than 1) imply that the price remains almost constant until the network is saturated, when it rises till the maximum price.

6.1.5 Bandwidth pricing function

Similarly, the spectrum broker sets the price of the bandwidth based on the global bandwidth demand, using a function. On this case we will analyse the effects on the spectrum broker and WSPs' benefits, as well as the global efficiency of the bandwidth. The parameters that calibrate the price are:

- *x*: establishes the fixed cost per unit of bandwidth.
- *y*: establishes the dependency of the cost with the global demand.
- *tau*: defines the shape of the function.

7 Results

Following section presents the results obtained for the four set of parameters studies realised and the last subsection presents a basic analysis of the different market characteristics of each scenario described in section 6.1.1.

7.1 Genetic Algorithm

This subsection presents summarization of the results after running several simulations varying the different parameters from the genetic algorithm, the whole analysis can be found in Appendix A. The different parameters are analysed first depending on the scenario. Afterwards some conclusions are stated.

7.1.1 Scenario A

The influence of the different parameters on scenario A is shown in Figure 25.

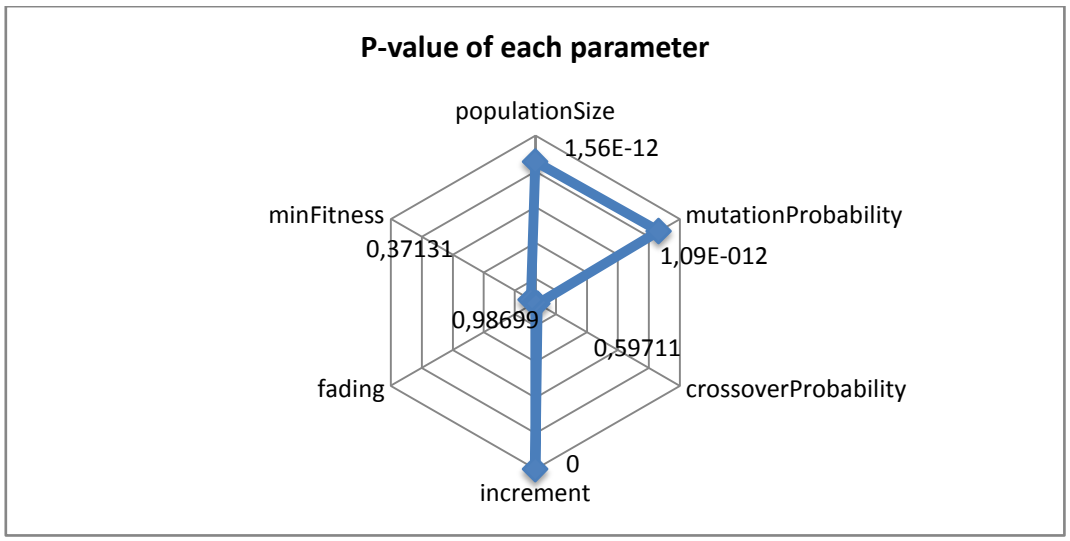


Figure 25: P-value of each parameter on scenario A

After analysing the most important values, we discovered that both the **increment** and the **mutation probability** should take **low** values for optimal results. Also the population size should be relatively small (around 20) and the minimum fitness should be below the 1000. The rest of parameters do not present much influence in the results.

7.1.2 Scenario B

Following figure shows the relevance of the parameters for scenario B.

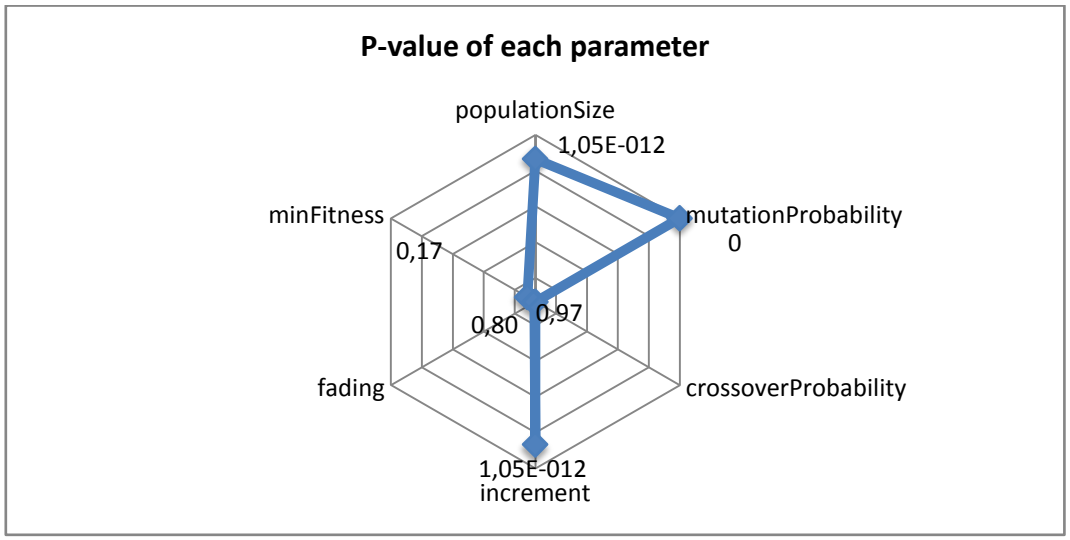


Figure 26: P-value of each parameter on scenario B

In this case, it is WSP's interest to set the **increment** and **mutation probability** to **high** values. The population size is also bigger in this scenario (around 70), but the rest of parameters are the same than scenario A.

7.1.3 Genetic C

Again, the ANOVA analysis shows similar importance in the parameters, and the algorithm behaves quite randomly, with the **increment** and **mutation probability** taking **high** values.

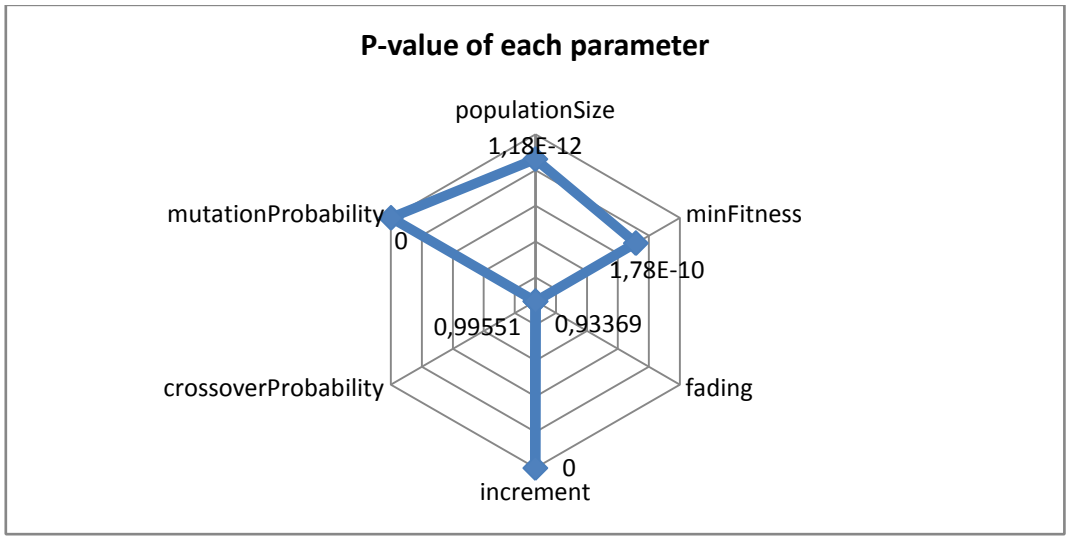


Figure 27: P-value of each parameter on scenario C

7.1.4 Scenario D

The ANOVA analysis of the scenario D shows that the most relevant parameters are the population size, the increment and the mutation probability; while the other 3 parameters are almost irrelevant in the revenue of the WSP.

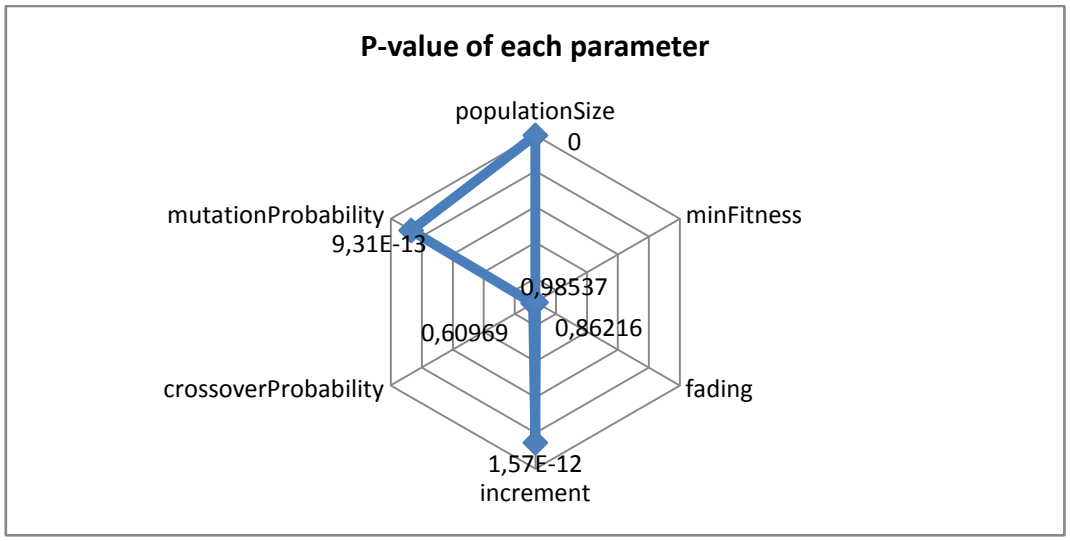


Figure 28: P-value of each parameter on scenario D

In this case, the **increment** takes **low** values, but the **mutation probability** is **high**, making the strategies to vary often, but little. The population size is considerably low in this scenario, as it is only 8 elements. The rest of parameters follow the same trend than previous scenarios.

7.1.5 Scenario E

Scenario E is only influence by the mutation probability and increment parameters. After calibrating the **increment** as a really **low** value, the second one loses its impact, though.

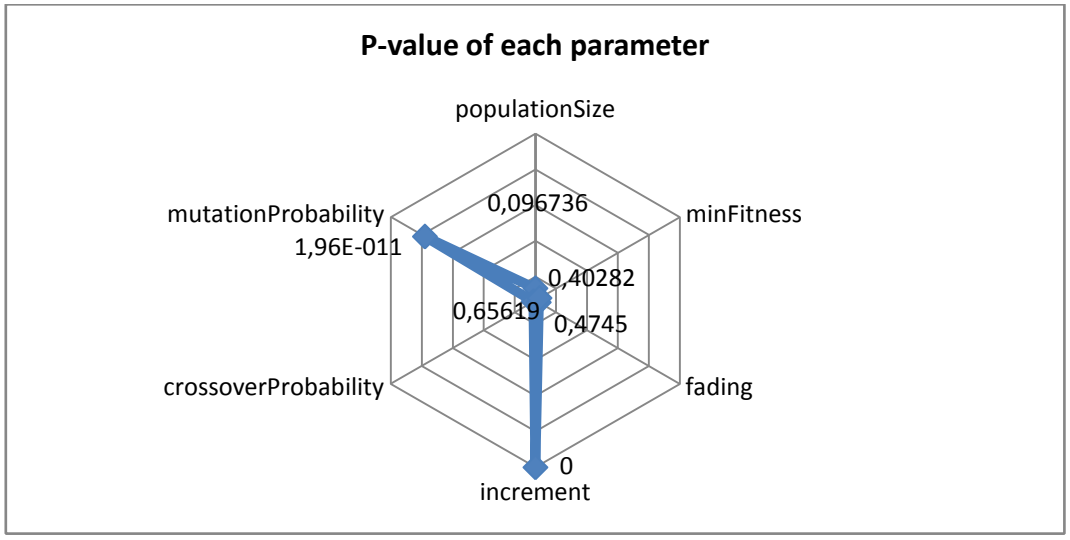


Figure 29: P-value of each parameter on scenario E

7.1.6 Scenario F

Figure 30 shows that the **increment** is the more relevant parameter in scenario F and it should take **high** values for maximizing the WSP's utility.

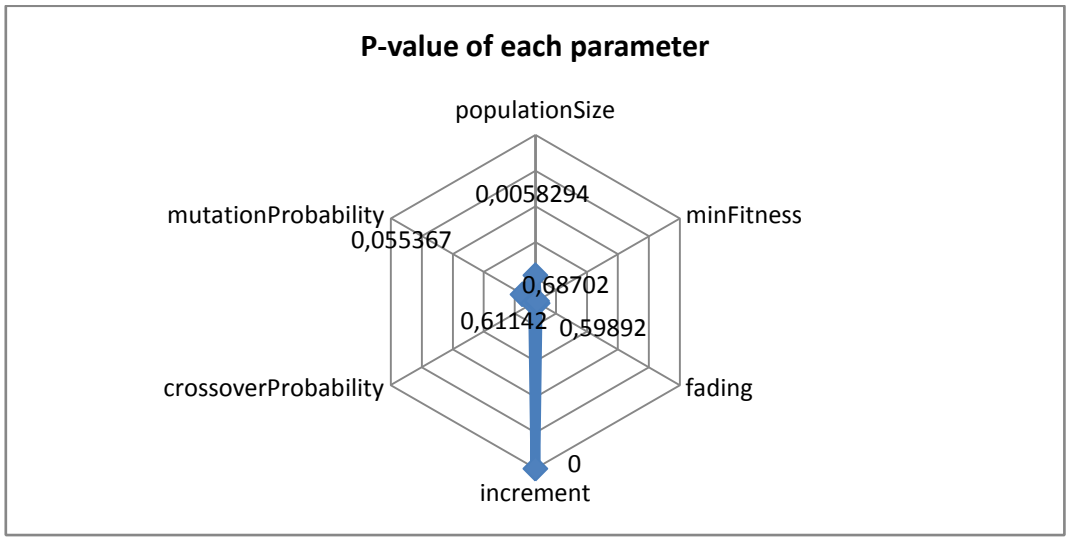


Figure 30: P-value of each parameter on scenario F

7.1.7 Genetic G

On scenario G, the increment is the most relevant parameter followed by the mutation probability and the population size. As the other scenarios representing the monopoly, it is the WSP's interest to assign a **low** value to the **increment** parameter, so that the selected strategies do not vary too much.

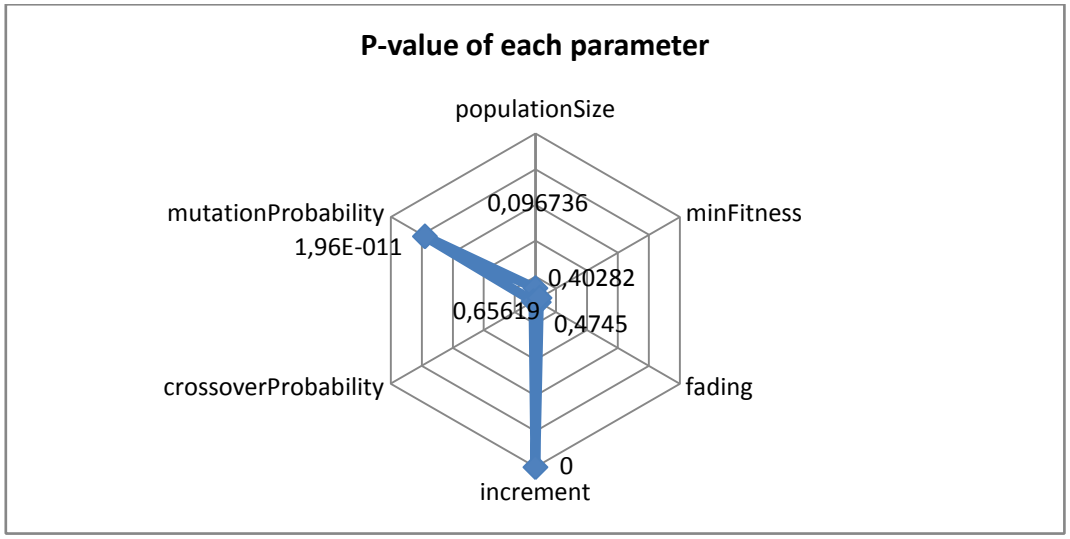


Figure 31: P-value of each parameter on scenario G

7.1.8 Scenario H

Figure 32 represents the relevance of the different parameters on scenario H. In this case the optimal value for the **increment** is also **low**, but combined with a high mutation probability, so that even if the strategies mutate often, this variation is small.

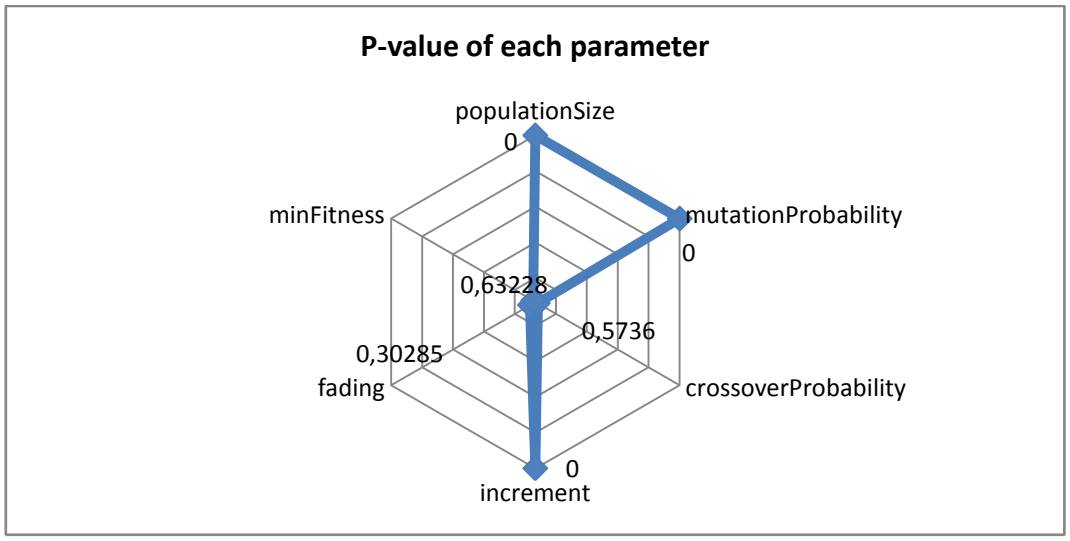


Figure 32: P-value of each parameter on scenario H

7.1.9 Scenario I

Scenario I's outcomes are only affected slightly by the learning parameters. Being practically irrelevant the values set for them in the WSP's utility.

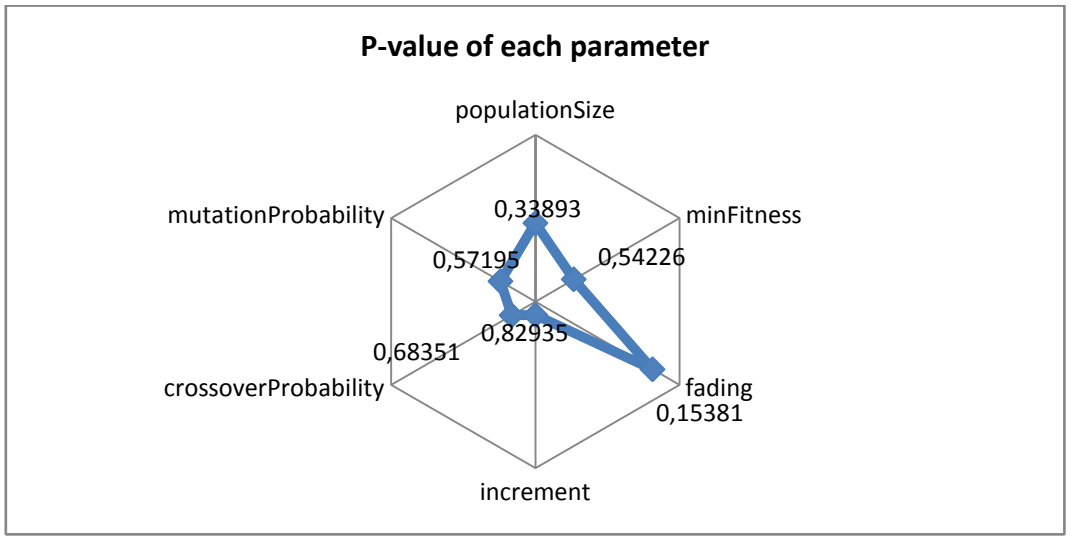


Figure 33: P-value of each parameter on scenario I

7.1.10 Conclusions

Considering a single variable in the learning algorithm, the most important variables are the mutation probability and the increment. On the other hand the crossover probability and the fading are irrelevant in every scenario. Regarding the population size, most of the scenarios preferred a low value, allowing the simulation to complete more learning cycles (as a learning cycle last as many market cycles as elements in the population). And the minimum fitness usually should be lower to the average revenue obtained by the service provider.

Therefore if we consider the most relevant parameters, the mutation probability and the increment, they define the variability of the strategies, i.e. if the strategies of one cycle will be similar o completely different to the previous cycle. As we can see, situations without competition (monopoly) or low risk (high-medium network load) prefer low variability in the strategies, leading to simulations that converge to a concrete set of strategies. On the other hand, when the competition is higher, the service provider prefers playing unexpected strategies, selecting high values for the increment and mutation probability parameters.

7.2 Reinforcement Algorithm

As previous section, here a summarization of the results obtained after the reinforcement algorithm are presented. The complete analysis can be found in Appendix B.

7.2.1 Scenario A

The most relevant parameter in scenario A is the **precision**, which produces better outcomes when it is **low** (around 3).

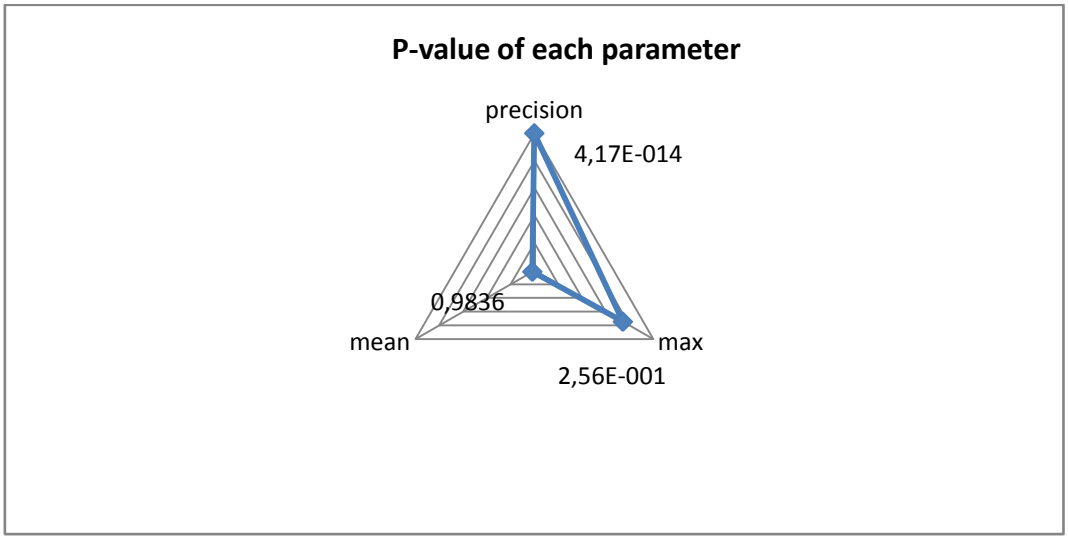


Figure 34: P-value of each parameter on scenario A

7.2.2 Scenario B

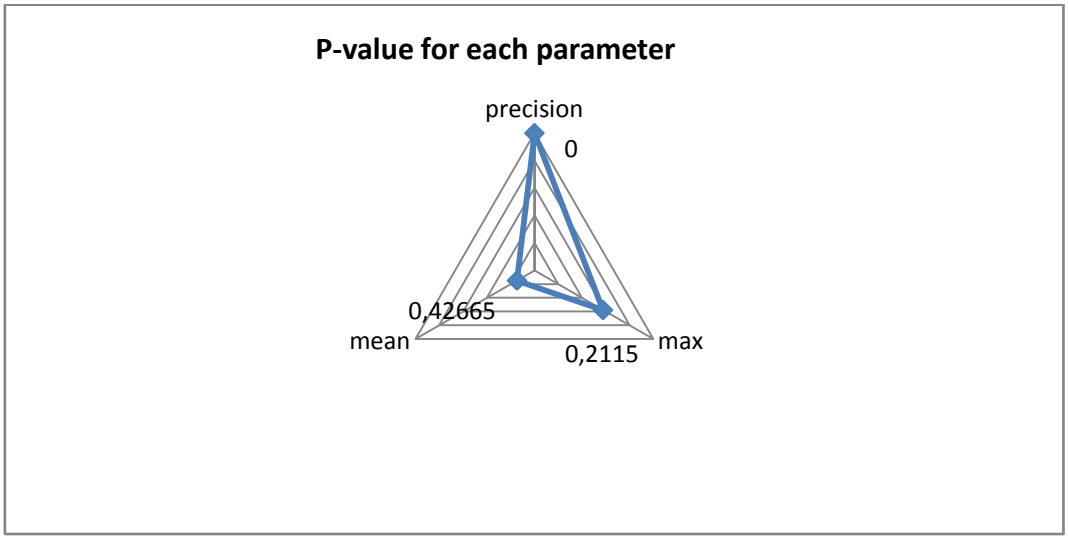


Figure 35: P-value of each parameter on scenario B

Like in scenario A, **precision** has the biggest influence, but in this case is better for **higher** values; concretely it should take values above 50.

7.2.3 Scenario C

As in scenario B, the precision in scenario C should be high, in this case above 70. The other parameters are irrelevant for the WSP's utility.

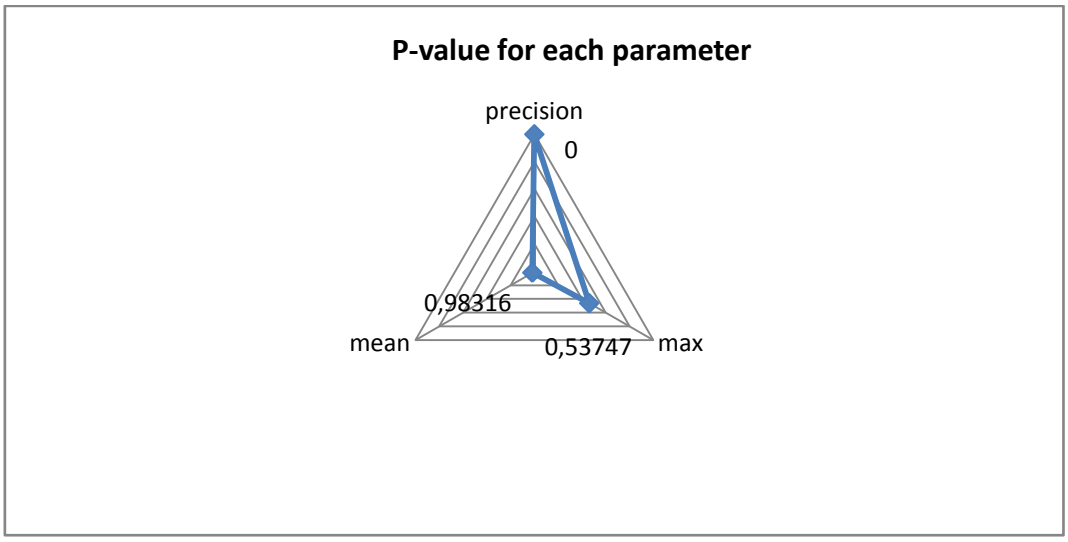


Figure 36: P-value of each parameter on scenario C

7.2.4 Scenario D

We obtain the same results in the parameters importance order in the scenario D. And as in the case of monopoly in A, the **precision** should take **low** values.

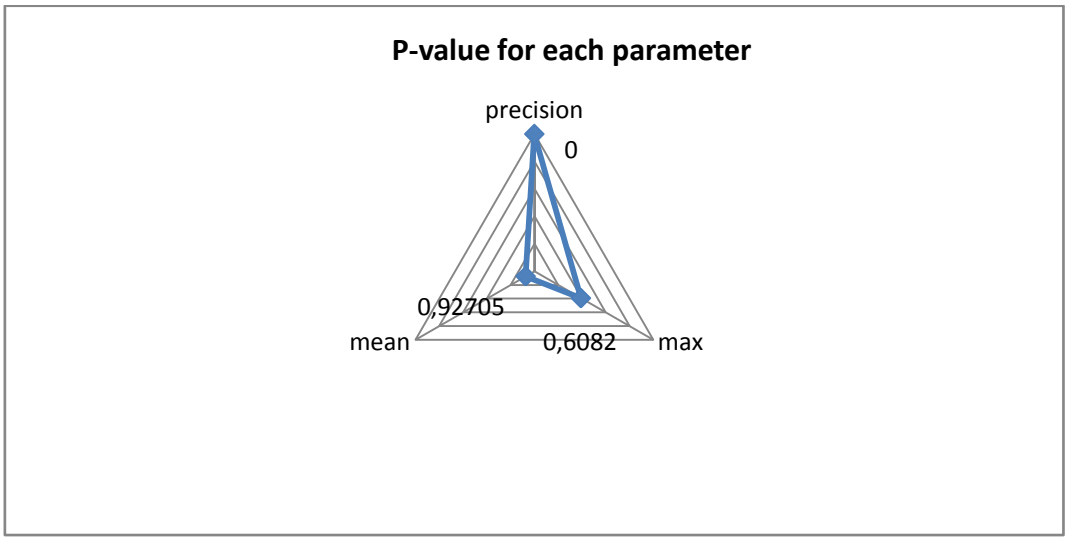


Figure 37: P-value of each parameter on scenario D

7.2.5 Scenario E

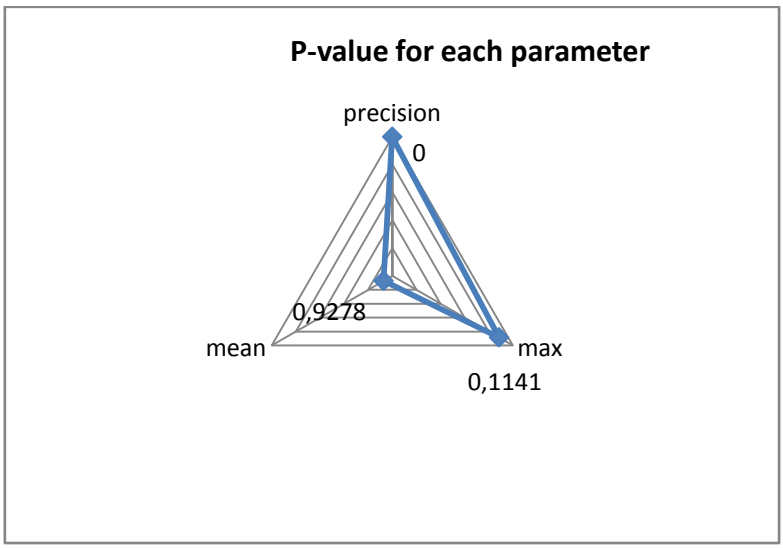


Figure 38: P-value of each parameter on scenario E

Scenario E is also characterised for preferring **low** values for the **precision**, and not being influenced by the mean and max coefficient.

Reinforcement F

This is the first scenario where the parameters importance seems different, as shown by the ANOVA analysis:

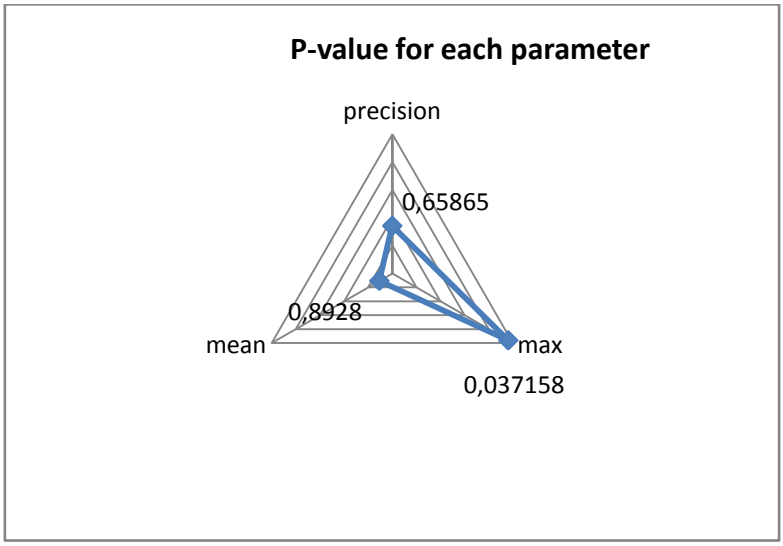


Figure 39: P-value of each parameter on scenario F

After the separate analysis for each parameter, it has been notice, that none present much influence in the outcomes for scenario F.

Scenario G

Scenario G is similar to the other monopoly situations, where the most influence is caused by the **precision**, and it should take **low** values preferably.

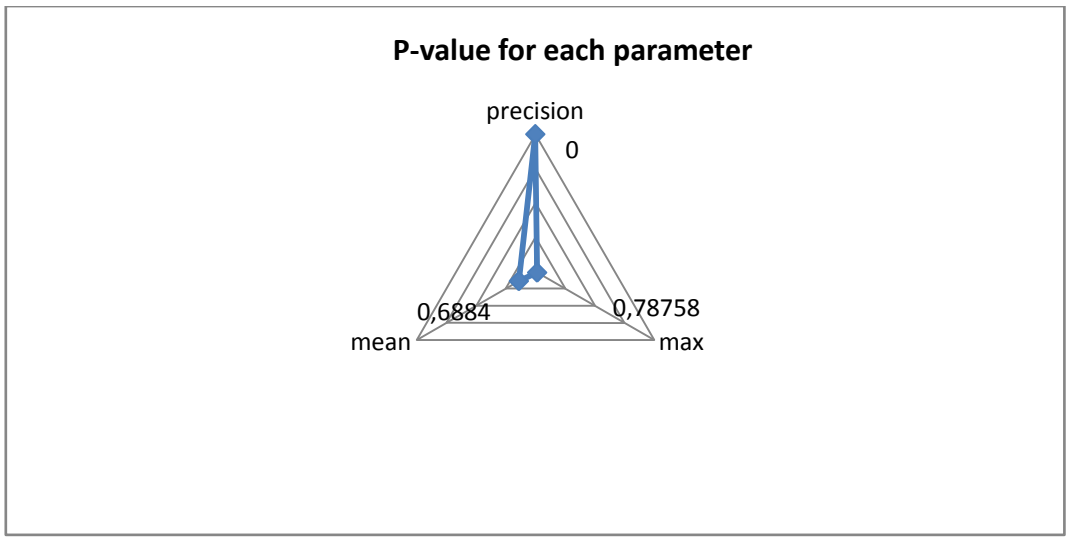


Figure 40: P-value of each parameter on scenario G

Scenario H

Scenario H is similar to scenario G. **Low** values for **precision**, which is the most relevant parameter.

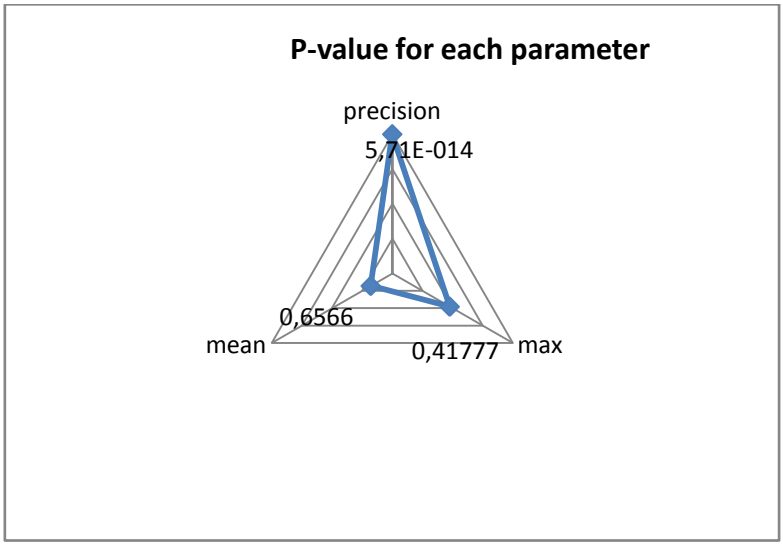


Figure 41: P-value of each parameter on scenario H

Scenario I

Finally, scenario I is **very little influenced** by the parameters selected for the learning algorithm.

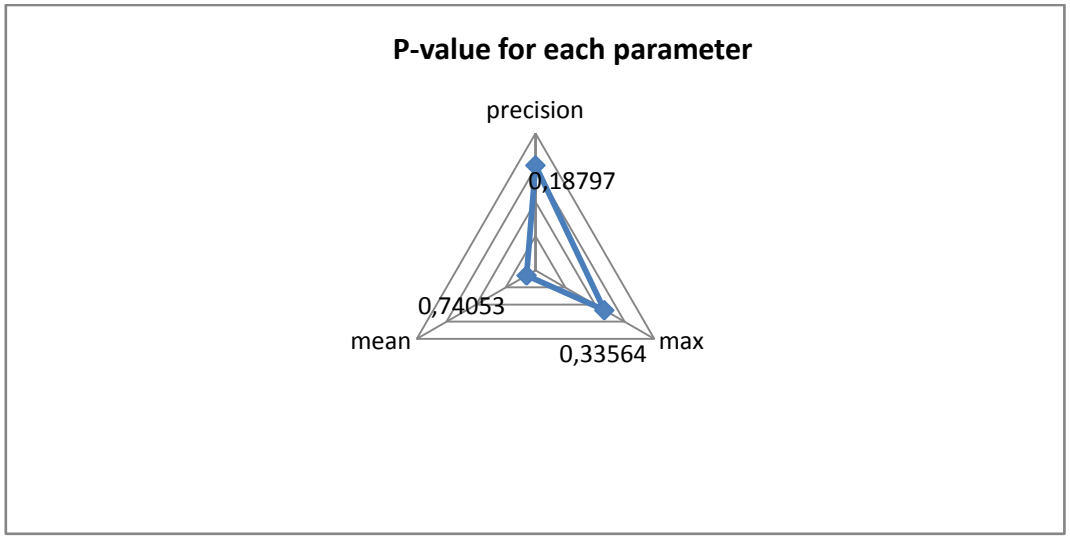


Figure 42: P-value of each parameter on scenario I

7.2.6 Conclusions

The most important parameter in the reinforcement learning is the precision or number of chunks each variable will be divided into. In cases with low competition, the service provider prefers low values of precision, which will allow the algorithm to be learn faster, as there are less elements to evaluate. With competition and low network load, however, it is preferred to have higher values for the precision.

Regarding the max and mean coefficient, they are not very important, but usually is better to give more relevance to the average revenue (mean coefficient = 1) than to the maximal revenue obtained by a strategy.

7.3 Analysis of the learning algorithms

The efficiency of the learning algorithms has been analysed using a dumb learning module, which selects randomly a value for each variable. The following table shows the average revenue for each scenario gained using each algorithm:

	Random	Genetic	Reinforcement
A	579127,969	780510	755660
B	132474,289	153140	134490
C	3182,72822	8853,3	3222,4
D	1,27E+007	15401000	14032000
E	7,39E+006	7562600	7541700
F	2,74E+006	2781500	2,75E+006
G	2,18E+007	29878000	2,51E+007
H	1,48E+007	16112000	15677000
I	6,36E+006	6393100	6375800

Table 4: WSP's utility for each learning algorithm

	% Genetic	% Reinforcement
A	34,77	30,48
B	15,60	1,52
C	178,17	1,25
D	21,20	10,43
E	2,37	2,09
F	1,42	0,20
G	37,16	15,35
H	9,05	6,11
I	0,59	0,32

Table 5: Enhancement compared to random strategies by using a learning algorithm

Table 5 presents the percentage of increase that the WSP's utility gains if it uses a learning algorithm instead of choosing randomly a strategy. As can be seen, the scenarios representing a monopoly are the most favoured by the fact of using a learning algorithm, instead of just randomly selecting a strategy. However for the oligopoly, the advantages of using one of the algorithms proposed is not appreciable, which means that there is no such learning in the algorithms. This is due to the fact that more complex learning algorithms are necessary, maybe considering multiple cycles, instead of just selecting the next cycle value for each variable.

7.4 WSP pricing function

Next, the effects of each parameter from the service provider pricing function will be analysed, first separately for each scenario; and then, obtaining some conclusions considering the whole picture.

7.4.1 Scenario A

The following picture shows the influence of the parameters $Price_{min}$ and $Price_{max}$:

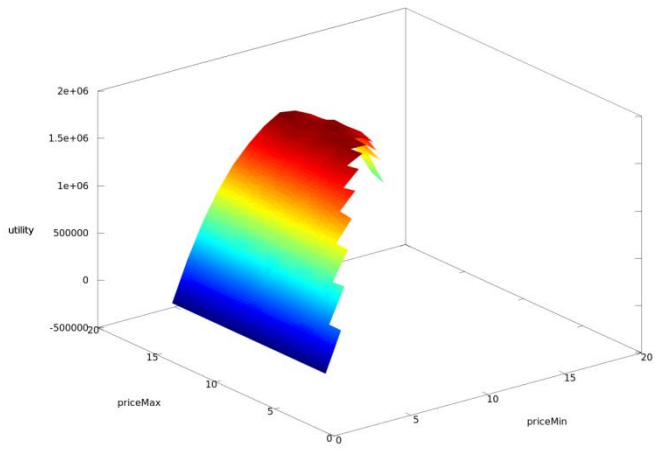


Figure 43: WSP's utility varying parameters $Price_{min}$ and $Price_{max}$

Therefore the optimal value for $Price_{min}$ is 8 and 15 for the $Price_{max}$. Using these values for each parameter, an additional experiment was run to analyse the effects of parameter τ:

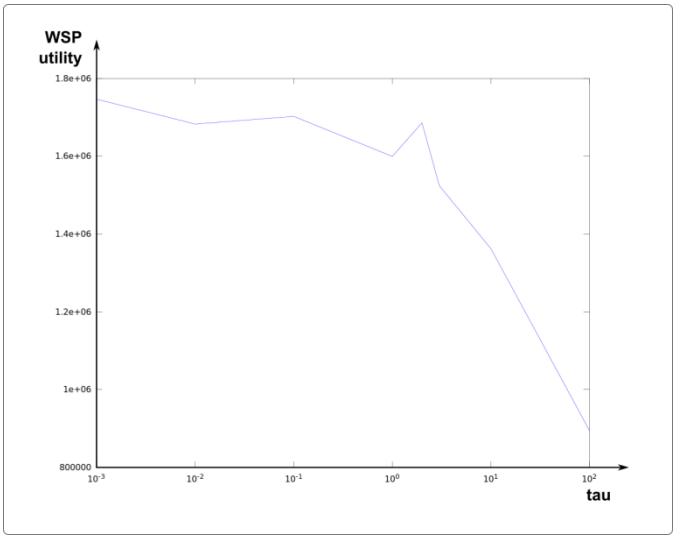


Figure 44: WSP's utility varying the τ parameter

So the optimal value is 0.001, which results in a pricing function like the following:

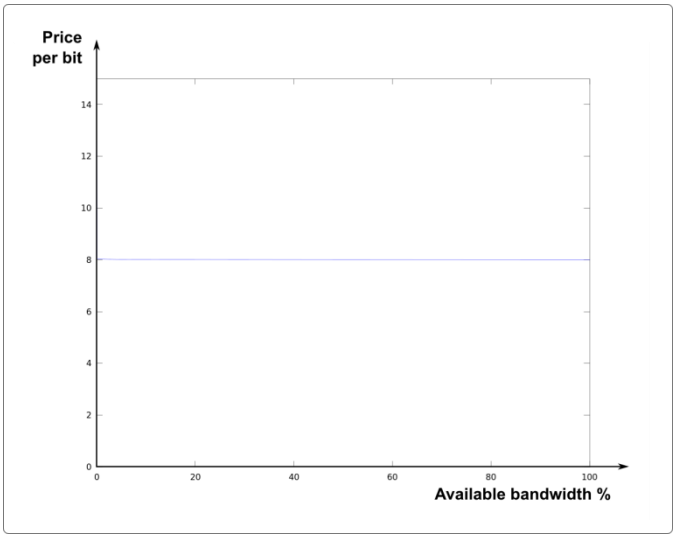


Figure 45: Optimal calibration of the pricing function for Scenario A

As the τ parameter takes a low value, the price is almost constant until high network occupancy.

7.4.2 Scenario B

On scenario B, the influence of the price range is not so clear as can be seen in Figure 46:

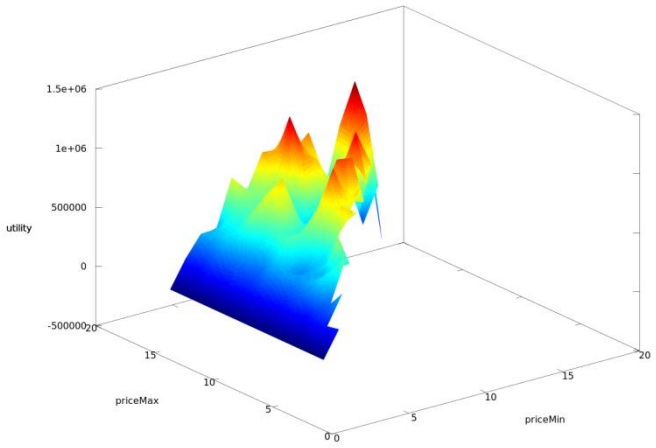


Figure 46: WSP's utility varying $Price_{min}$ and $Price_{max}$

On the other hand, the effect of τ parameter is optimal when its value is 2:

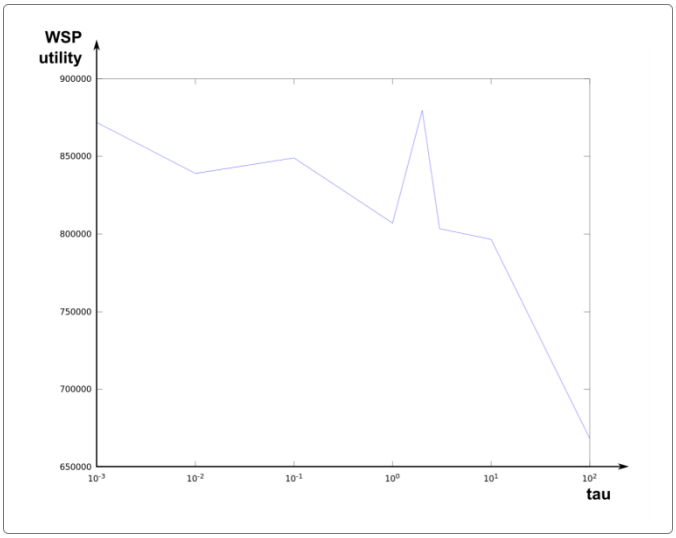


Figure 47: WSP's utility when varying τ

So in this case the pricing function will be concave:

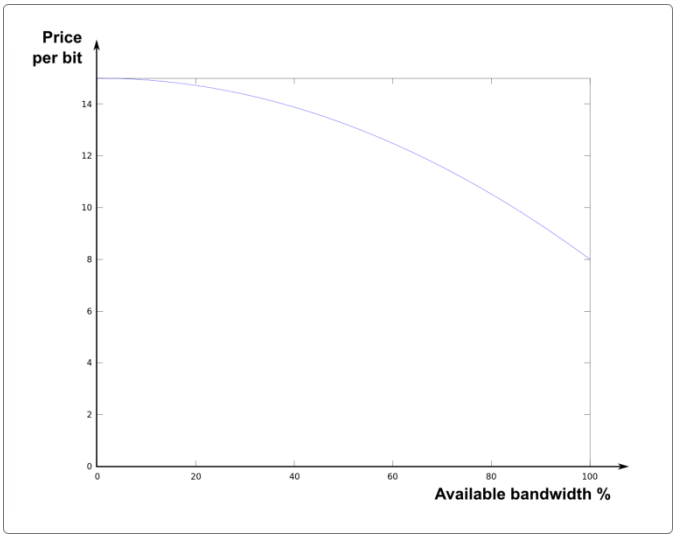


Figure 48: Optimal calibration of the pricing function for Scenario B

7.4.3 Scenario C

The optimal price range in scenario C is also between 8 and 15.

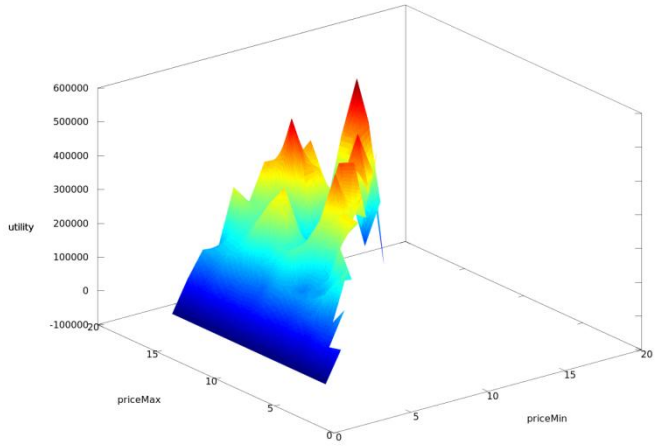


Figure 49: WSP's utility varying $Price_{min}$ and $Price_{max}$

As in scenario A, the optimal value for parameter τ is 0.001, so the pricing function would be the same.

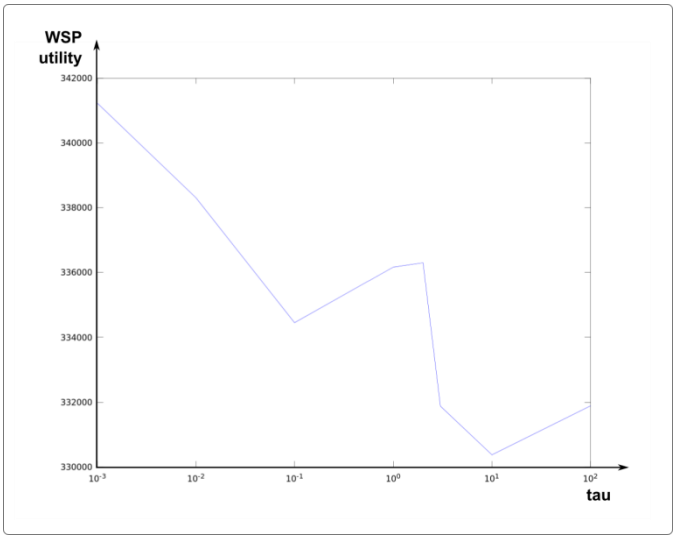


Figure 50: WSP's utility varying τ parameter

7.4.4 Scenario D

The optimal price range for scenario D is between 7 and 8:

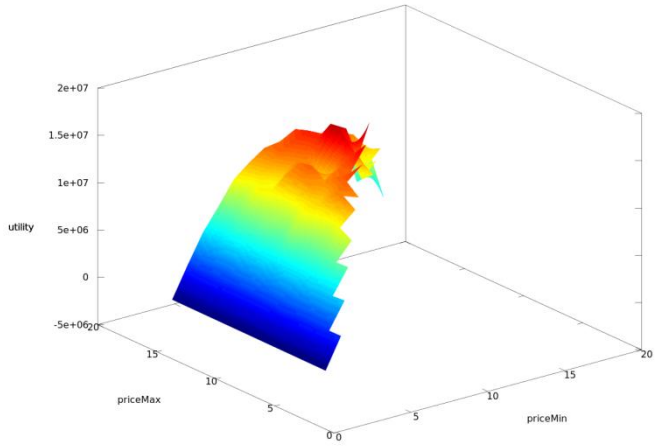


Figure 51: WSP's utility varying the $Price_{min}$ and $Price_{max}$

Regarding the τ parameter, its optimal value is 3:

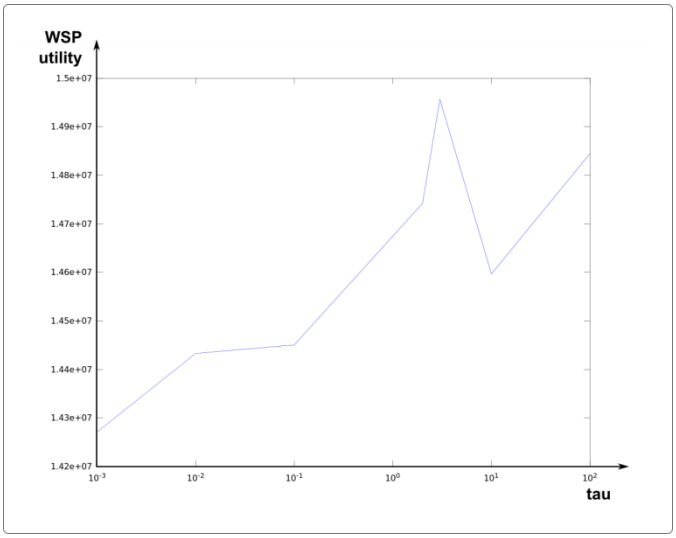


Figure 52: WSP's utility varying τ parameter

So the resulting price is practically 8 all the time:

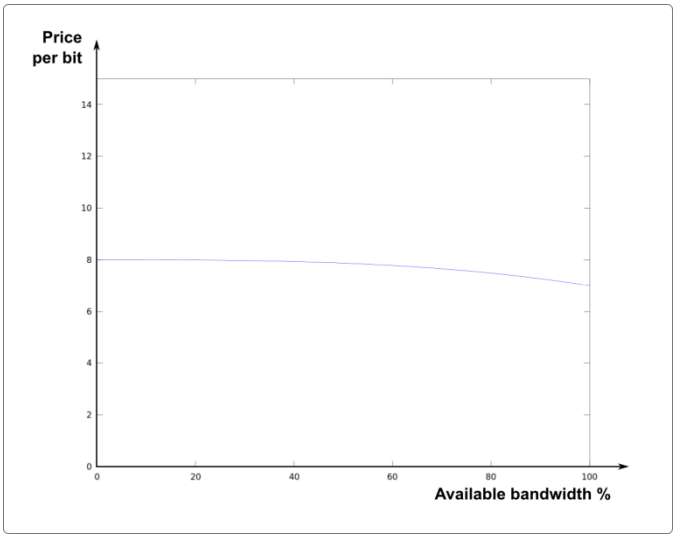


Figure 53: Optimal calibration of the pricing function for Scenario D

7.4.5 Scenario E

Scenario E results are pretty similar to scenario D, but in this case the price range is from 7 to 8.

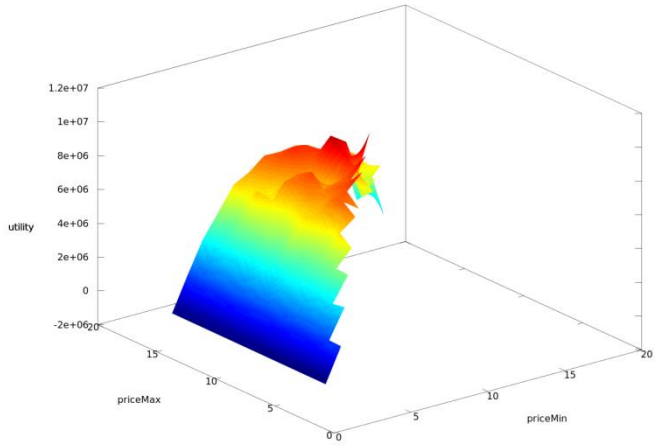


Figure 54: WSP's utility varying the $Price_{min}$ and $Price_{max}$ parameters

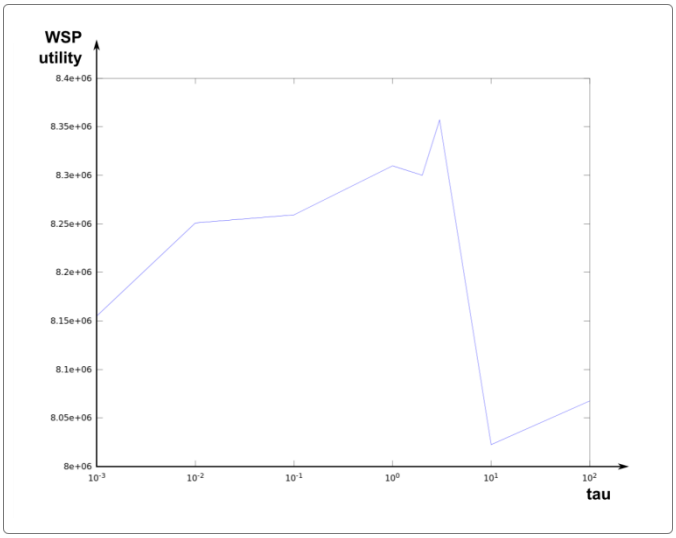


Figure 55: WSP's utility varying the τ parameter

So in this case the pricing function is almost constant at value 9.

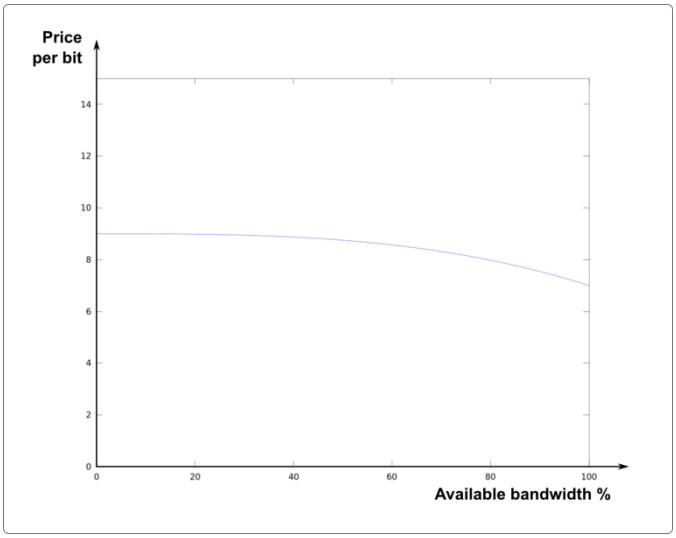


Figure 56: Optimal calibration of the pricing function for Scenario E

7.4.6 Scenario F

Scenario F results are the same to scenario D.

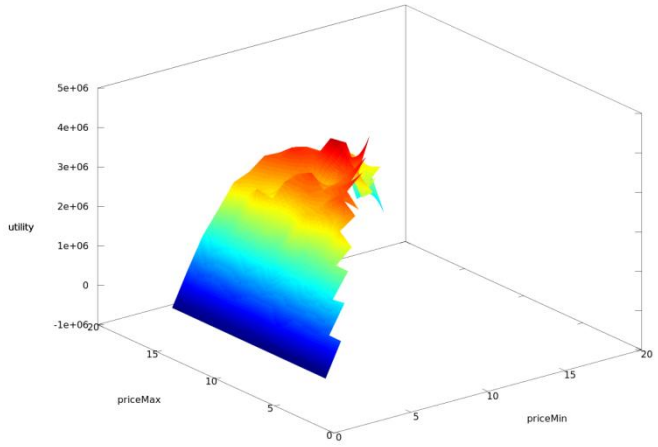


Figure 57: WSP's utility varying $Price_{min}$ and $Price_{max}$ parameters

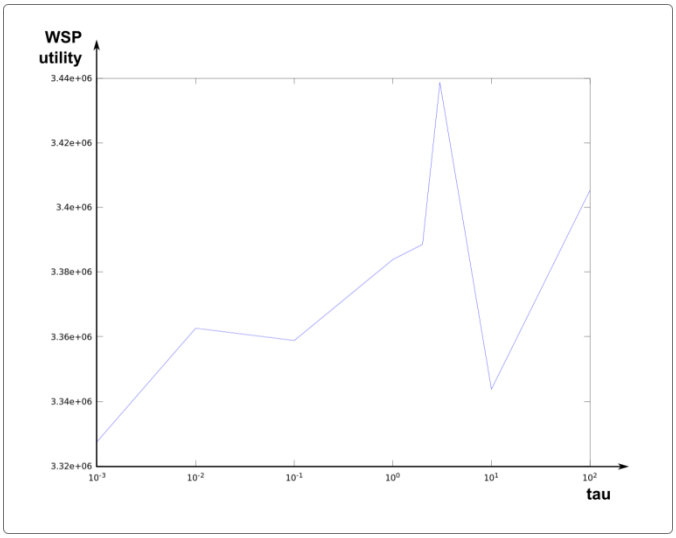


Figure 58: WSP's utility varying the τ parameter

7.4.7 Scenario G

The increase on the network load makes the price range to rise, which in scenario G its optimal value is between 10 and 12.

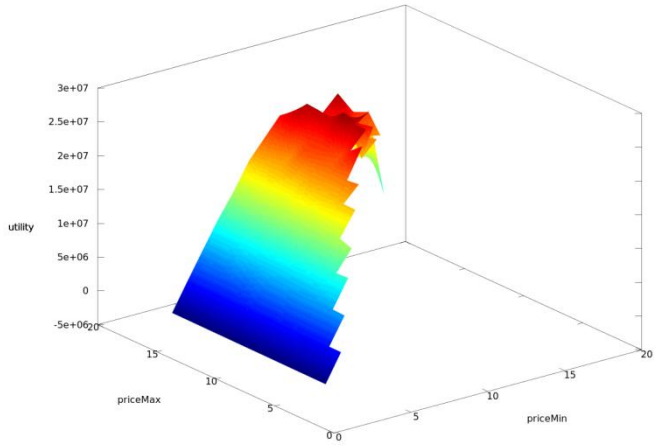


Figure 59: WSP's utility varying $Price_{min}$ and $Price_{max}$ parameters

On the other hand, the low value of the parameter τ makes that the price is almost constant at value 10.

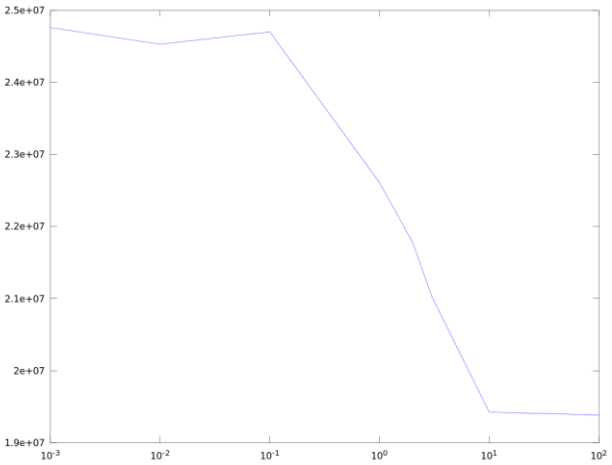


Figure 60: WSP's utility varying the τ parameter

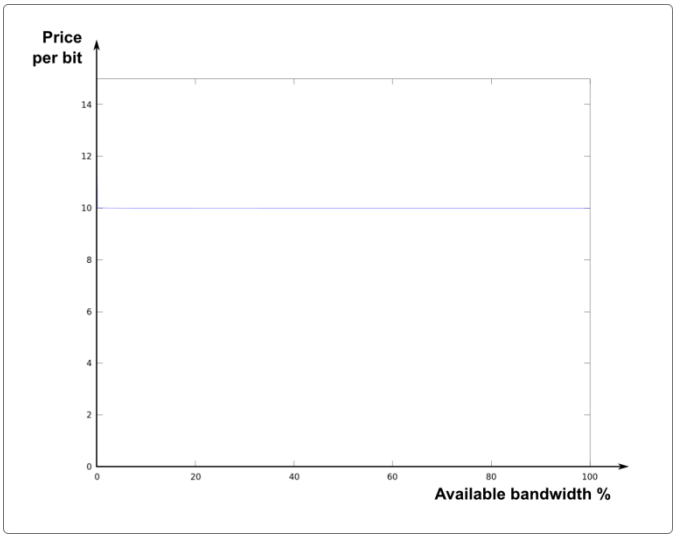


Figure 61: Optimal calibration of the pricing function for Scenario G

7.4.8 Scenario H

In this case, the minimum price lowers a little bit (9), as there is competition between two WSPs. The rest of the parameters remain the same.

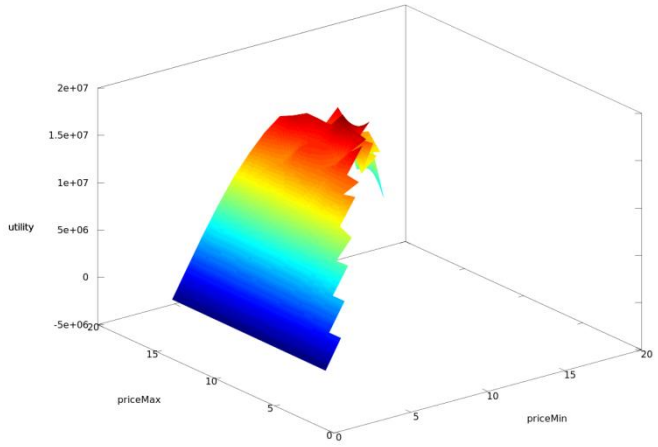


Figure 62: WSP's utility varying the $Price_{min}$ and $Price_{max}$ parameters

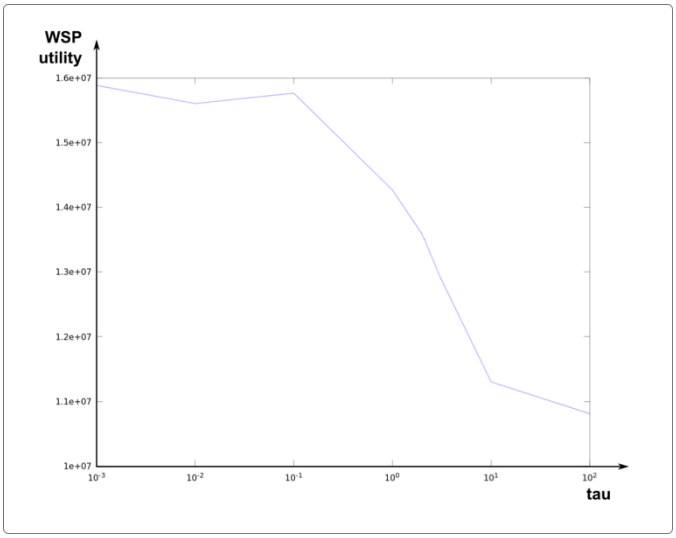


Figure 63: WSP's utility when varying the τ parameter

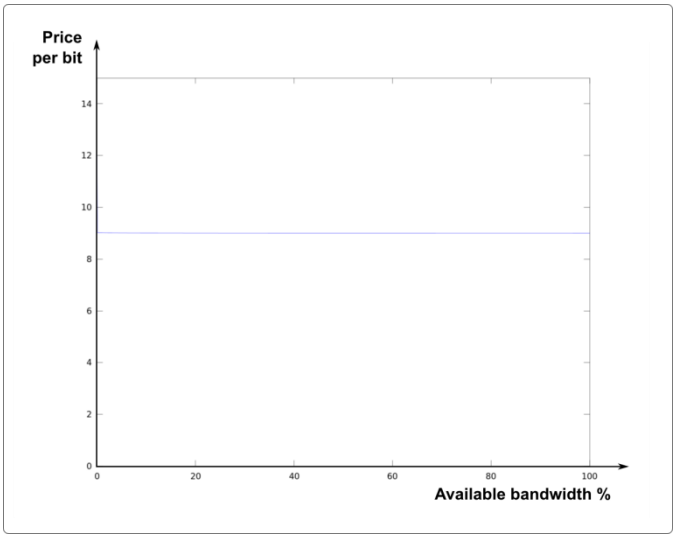


Figure 64: Optimal calibration of the pricing function for Scenario H

7.4.9 Scenario I

Again, due to the increase in the competition, the optimal price range is lower than in the two previous scenarios, in this case from 8 to 11.

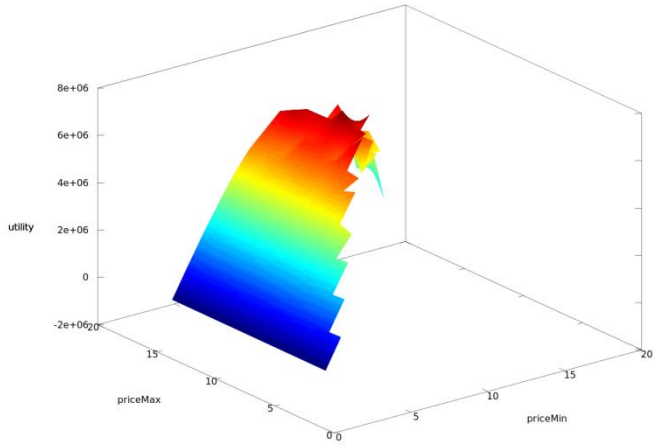


Figure 65: WSP's utility varying $Price_{min}$ and $Price_{max}$ parameters

On this case, however, the optimal τ value is 0.1:

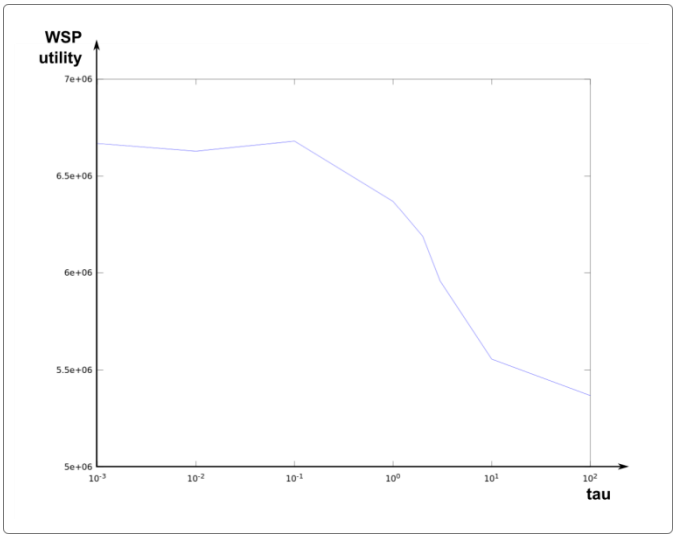


Figure 66: WSP's utility varying the τ parameter

So the result pricing function is the following:

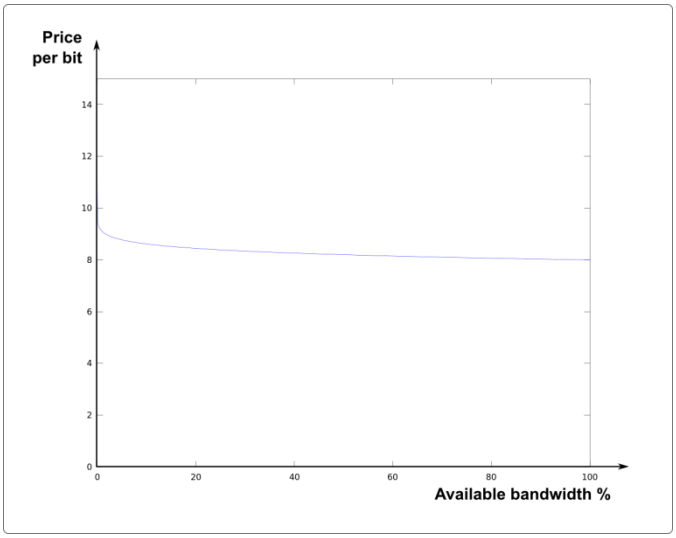


Figure 67: Optimal calibration of the pricing function for Scenario I

7.4.10 Conclusions

The following table approximately the mean value of the price per bit transferred depending on the scenario:

	Monopoly	Duopoly	Oligopoly
Low network load	8	12.7	8
Medium network load	8.5	8.5	7.75
High network load	10	9	8.25

Table 6: Average price per bit transferred

So as we can see, except for the anomaly of scenario B, prices raise when the network load is higher, and they drop as there is more competition.

7.5 Spectrum Broker pricing function

The function of the Spectrum Broker to decide the price that will b e charged to the different WSP for each unit of bandwidth allocated is defined in chapter 0. In this section we will analyse the effects of its parameters to know how to calibrate them correctly when running simulations scenarios.

In this case there are three parameters: x, y and τ, and after analysing each scenario separately we have realised that the effects of each parameter do not depend on the scenario. Following figures show the effects of each parameter on the global utility for each role in the market in scenario A:

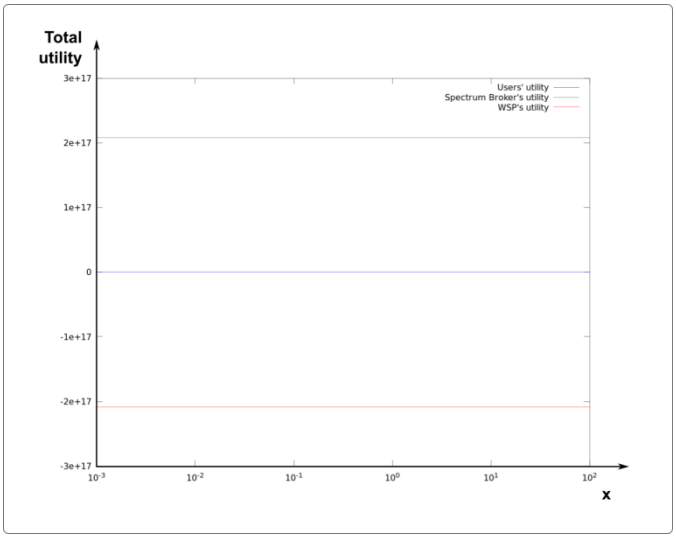


Figure 68: Total utility for each role varying the x parameter

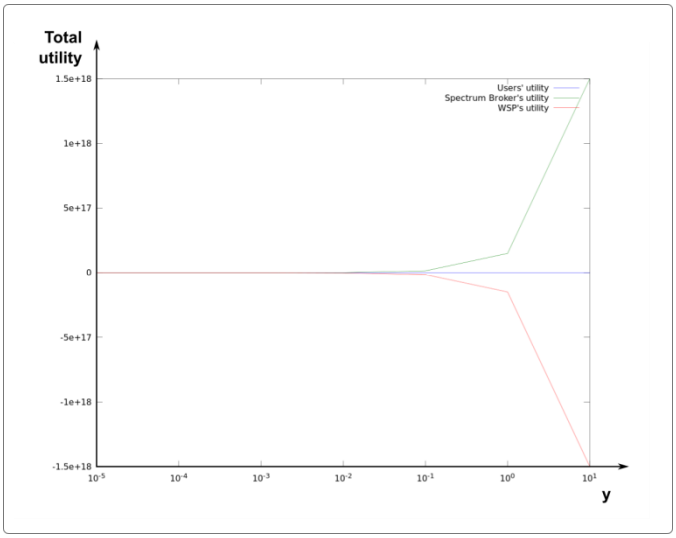


Figure 69: Total utility of each role varying the y parameter

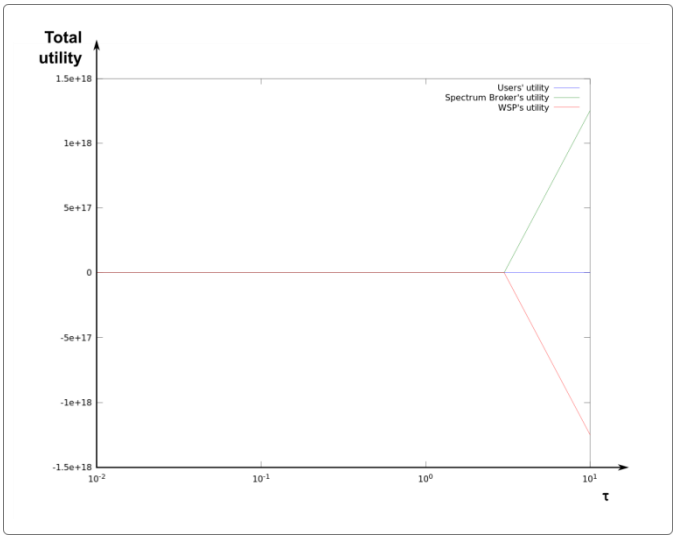


Figure 70: Total utility of each role varying the τ parameter

So as we can see, we must set both the y and the τ below a value, so that the WSPs utility is positive (and therefore the scenario makes sense). On the other hand, parameter x seems not to have much influence in the results, as it is a constant. Next picture shows the effect considering the y and τ dimensions:

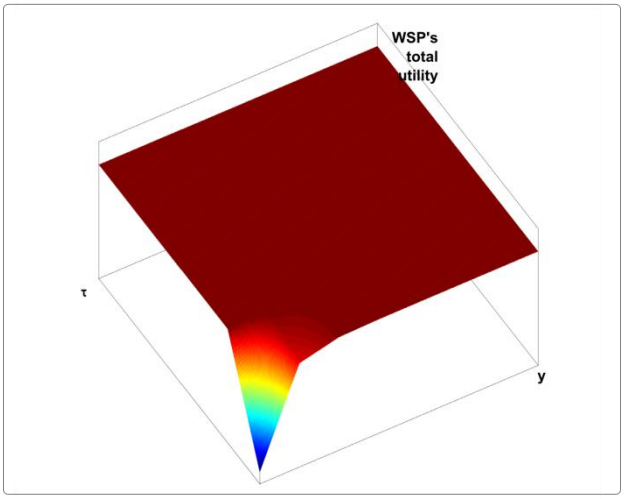


Figure 71: WSP's utility as function of parameters y and τ

Appendix C shows the utility obtained by the WSP depending on these two parameters to help on the correct parameter calibration.

7.6 Scenario Analysis

On this last section we analyse the utility of the different roles on the market depending on the scenarios run.

As the results show, the market structure creates competition both between the end users and the service providers, so the more users there are, their utility decreases. On the other hand, service providers compete among them, so the more WSPs, the less utility they obtain, going it to the end users. Next figures show the total utility of all the users and service providers depending on the scenario run.

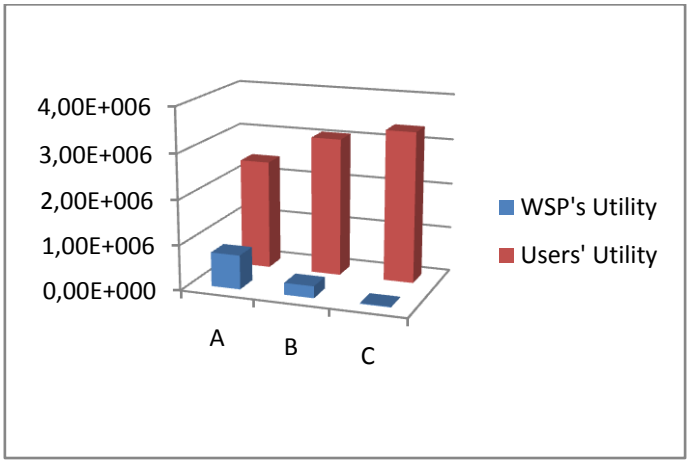


Figure 72: Utilities for the low network load (10 users)

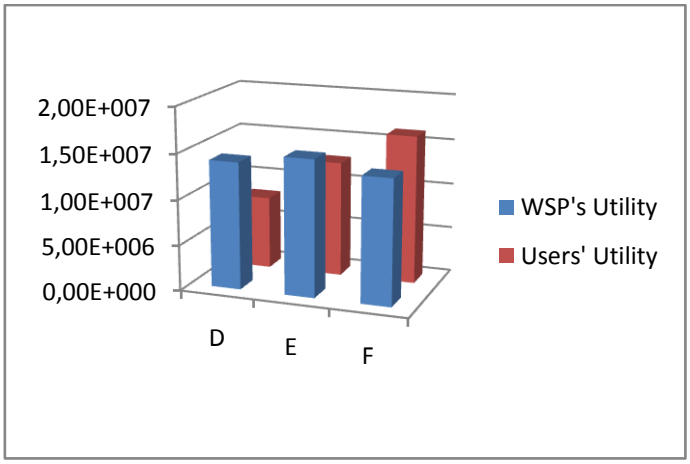


Figure 73: Utilities for medium network load (100 users)

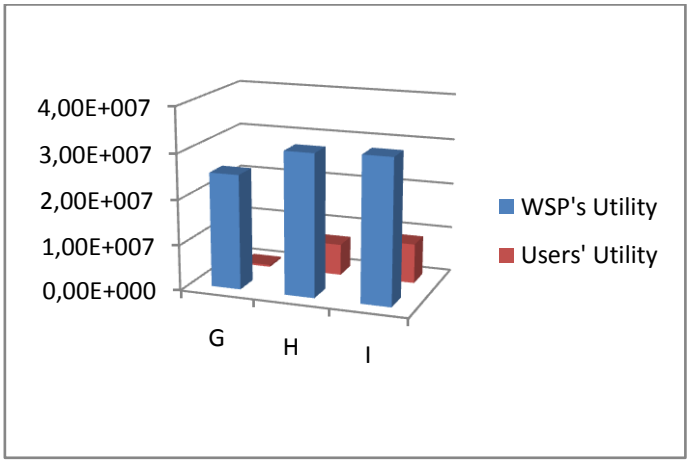


Figure 74: Utilities for high network load (200 users)

The next figure shows the percentage of connections that are not served on each scenario:

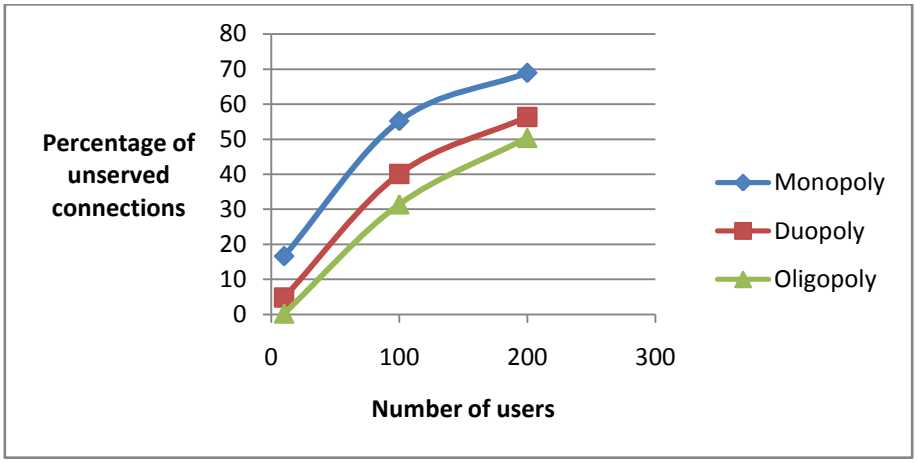


Figure 75: Percentage of non-served connections for each scenario

So as we can see, the block probability does not only depend on the number of users, but also on the number of WSPs, as the more WSPs, the lower they are willing to set the price for connection.

8 Conclusions and Future Work

We have presented a possible market model to represent the new tendencies towards a more dynamic use of the spectrum. It is based on a centralised figure, the Spectrum Broker, who is in charge of managing the spectrum. Then we have proposed a novel approach to face and analyse the situations that appear under these circumstances: an agent-based simulation model. And so we have defined and implemented a simulation model, representing the market model described using OMNeT++ as a framework. Having reusability and customization as main goals for the simulation model, we have decided to provide it with artificial intelligence, so the service providers are capable of making their own decisions, independently of the new modifications of the simulation model. This is achieved through two different learning mechanisms: (1) a genetic algorithm and (2) a reinforcement algorithm. However, as has been proof in section 7.3, these algorithms are not powerful enough in complex situations where there are multiple WSPs in the market. A possible enhancement would be considering multiple plays on the learning algorithm instead of a single one. This thesis concludes with the analysis of a first set of simulations, pointing out the effects of the different parameters of the simulation model. As proof of the easiness of simulation model's modification, annex B presents two different extensions of the model and how to achieve them.

Future work is two-folded. On one hand many different scenarios can be defined, simulated and analysed, using the current simulation model, and they can lead to interesting analysis over the problem stated. On the other hand, the simulation model can be extended and enhanced to represent more accurately the new tendencies of the market and the underlying technology. As regards to the new economic approaches it may be interesting analysing different allocation and pricing mechanisms and see the difference between them, both in terms of bandwidth efficiency and benefit distribution. Regarding the technological point of view, it will be interesting using some of the already deployed wireless network simulators for OMNeT++ and extend and adapt it so that the market model can include more detailed technical issues and some QoS parameters can be taken into account.

References

[1] M. A. McHenry, D. McCloskey and G. Lane-Roberts, "New york city spectrum occupancy measurements september 2004," 2004.

[2] L. Berlemann and S. Mangold, *Cognitive Radio and Dynamic Spectrum Access.* Chichester, UK : John Wiley & Sons, 2009, 2009.

[3] M. M. Buddhikot, "Understanding dynamic spectrum access: Models,taxonomy and challenges," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on,* 2007, pp. 649-663.

[4] FCC, "Notice of proposed rule making and order," 2003.

[5] Spectrum Policy Task Force (SPFT). (2002, Nov. 2002). *Report ET docet 02-135*. FCC.

[6] F. H. P. Fitzek and M. D. Katz, *Cognitive Wireless Networks :Concepts, Methodologies and Visions Inspiring the Age of Enlightenment of Wireless Communications.* Berlin: Springer, 2007.

[7] J. Chapin and W. Lehr, "COGNITIVE RADIOS FOR DYNAMIC SPECTRUM ACCESS - The Path to Market Success for Dynamic Spectrum Access Technology " *IEEE Communications Magazine,* vol. 45, pp. 96 <last_page> 103, 2007.

[8] D. Niyato and E. Hossain, "A game theoretic analysis of service competition and pricing in heterogeneous wireless access networks," *IEEE Transactions on Wireless Communications,* vol. 7, pp. 5150-5155, 2008.

[9] D. Niyato and E. Hossain, "Equilibrium and disequilibrium pricing for spectrum trading in cognitive radio: A control-theoretic approach," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE,* 2007, pp. 4852-4856.

[10] D. Niyato and E. Hossain, "Spectrum trading in cognitive radio networks: A market-equilibrium-based approach," *Wireless Communications, IEEE,* vol. 15, pp. 71-80, 2008.

[11] D. Niyato and E. Hossain, "A cooperative game framework for bandwidth allocation in 4G heterogeneous wireless networks," in *Communications, 2006. ICC '06. IEEE International Conference on,* 2006, pp. 4357-4362.

[12] D. Niyato and E. Hossain, "Competitive pricing for spectrum sharing in cognitive radio networks: Dynamic game, inefficiency of nash equilibrium, and collusion," *IEEE J. Select. Areas Commun.,* vol. 26, pp. 192-202, 2008.

[13] D. Niyato and E. Hossain, "Market-equilibrium, competitive, and cooperative pricing for spectrum sharing in cognitive radio networks: Analysis and comparison," *IEEE Transactions on Wireless Communications,* vol. 7, pp. 4273-4283, 2008.

[14] V. Rodriguez, K. Moessner and R. Tafazolli, "Auction driven dynamic spectrum allocation: Optimal bidding, pricing and service priorities for multi-rate, multi-class CDMA," in *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2005, September 11, 2005 - September 14,* 2005, pp. 1850-1854.

[15] D. Niyato and E. Hossain, "A game-theoretic approach to competitive spectrum sharing in cognitive radio networks," in *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE,* 2007, pp. 16-20.

[16] D. Niyato and E. Hossain, "Competitive spectrum sharing in cognitive radio networks: a dynamic game approach," *Wireless Communications, IEEE Transactions on,* vol. 7, pp. 2651-2660, 2008.

[17] Yongle Wu, Beibei Wang, K. J. R. Liu and T. C. Clancy, "Collusion-resistant multi-winner spectrum auction for cognitive radio networks," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE,* 2008, pp. 1-5.

[18] A. P. Subramanian and H. Gupta, "Fast spectrum allocation in coordinated dynamic spectrum access based cellular networks," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on,* 2007, pp. 320-330.

[19] D. Niyato, E. Hossain and Zhu Han, "Dynamics of Multiple-Seller and Multiple-Buyer Spectrum Trading in Cognitive Radio Networks: A Game-Theoretic Modeling Approach," *Mobile Computing, IEEE Transactions on,* vol. 8, pp. 1009-1022, 2009.

[20] Zhu Ji and K. J. R. Liu, "Multi-Stage Pricing Game for Collusion-Resistant Dynamic Spectrum Allocation," *Selected Areas in Communications, IEEE Journal on,* vol. 26, pp. 182-191, 2008.

[21] Shufang Lin and Xuming Fang, "Two-level game based spectrum lease framework in cognitive radio networks," in *Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on,* 2008, pp. 104-108.

[22] D. Niyato, E. Hossain and Long Le, "Competitive spectrum sharing and pricing in cognitive wireless mesh networks," in *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE,* 2008, pp. 1431-1435.

[23] D. Niyato, E. Hossain and Zhu Han, "Dynamic spectrum access in IEEE 802.22- based cognitive wireless networks: a game theoretic model for competitive spectrum bidding and pricing," *Wireless Communications, IEEE,* vol. 16, pp. 16-23, 2009.

[24] O. Ileri, D. Samardzija and N. B. Mandayam, "Dynamic property rights spectrum access: Flexible ownership based spectrum management," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on,* 2007, pp. 254-265.

[25] C. Courcoubetis and R. Weber, *Pricing Communication Networks :Economics, Technology and Modelling.* West Sussex: John Wiley & Sons, 2003.

[26] S. J. Russell, P. Norvig and J. F. Canny, *Artificial Intelligence :A Modern Approach.* Englewood Cliffs, N.J.: Prentice Hall, 2003.

[27] C. Camerer and Russell Sage Foundation, *Behavioral Game Theory :Experiments in Strategic Interaction.* New York N.Y.: Russell Sage Foundation, 2003.

Appendix A Parameter study of the Genetic Algorithm

The following appendix presents the results of the parameters study realised regarding the genetic algorithm. This analysis has been done separately for each scenario and some conclusions are presented at the end. The different parameters that have been analysed are the population size, the mutation and crossover probability, the minimum fitness, the fading and the increment. On each scenario, the relevance of each parameter is measured by means of an ANOVA test, and afterwards they are calibrated to maximize the WSP's utility

A.1 Scenario A

The first analysis shows the following ANOVA test values for each variable:

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	283.18	171.77	0.69	9786	0.09	1.04
Test statistic F	1.56E-012	1.09E-012	0.59	0	0.99	0.37

Table 7: ANOVA test values for the genetic algorithm on scenario A

So as we can observe, the biggest influence are the parameters increment, mutation probability and population size. Then it is the minimum fitness value and the crossover probability and finally the fading parameter. So we run a set of experiments for each parameter one by one. Following subsections present the results for the parameters that actually influence the WSP's revenue.

A.1.1 Increment

An experiment over the parameter increment ranging from 0 to 1 was performed and the results showed that the outcomes where better when the increment was around 0.1.

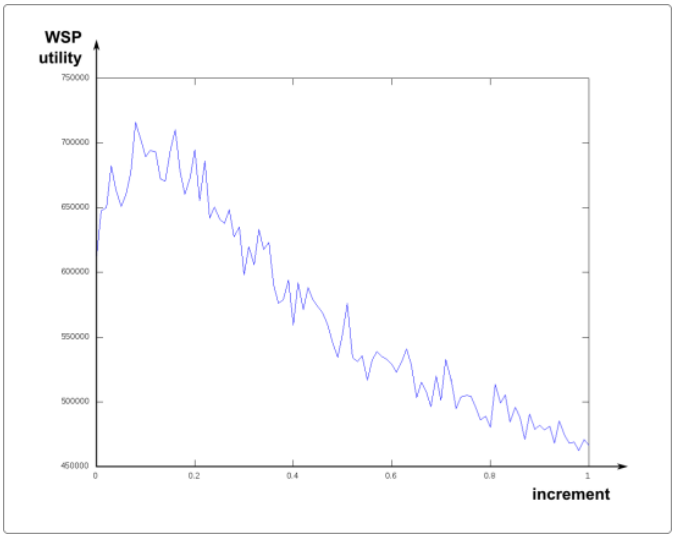


Figure 76: WSP's utility varying the increment parameter in scenario A

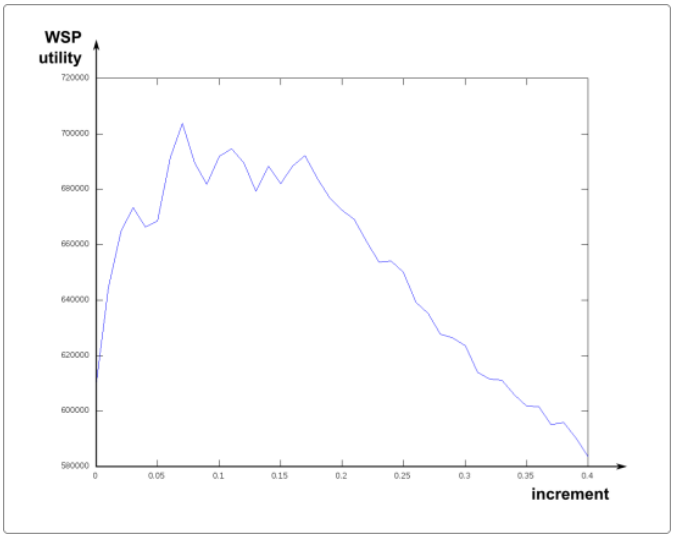


Figure 77: Close up of lower values of the increment parameter

A.1.2 Mutation Probability

The next parameter is the mutation probability. Its value also ranges from 0 to 1, but after selecting a low value for the increment (0.07), its effects are not that drastic:

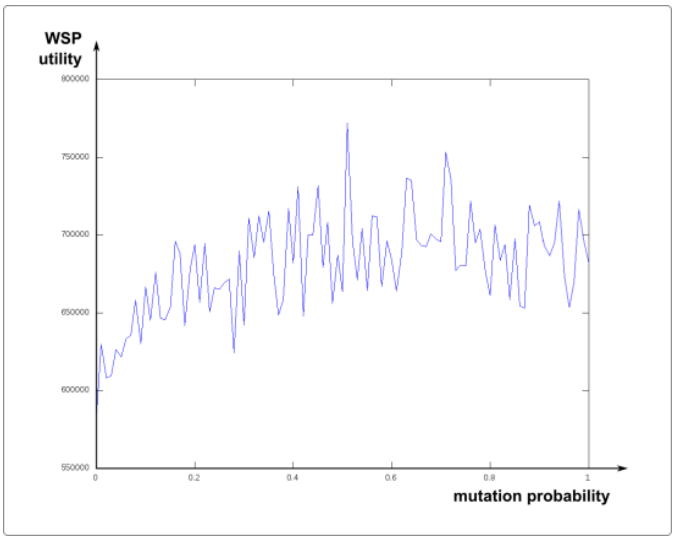


Figure 78: WSP's utility values varying the mutation probability on scenario A

We obtain a clearer vision when calibrating the rest of the parameters:

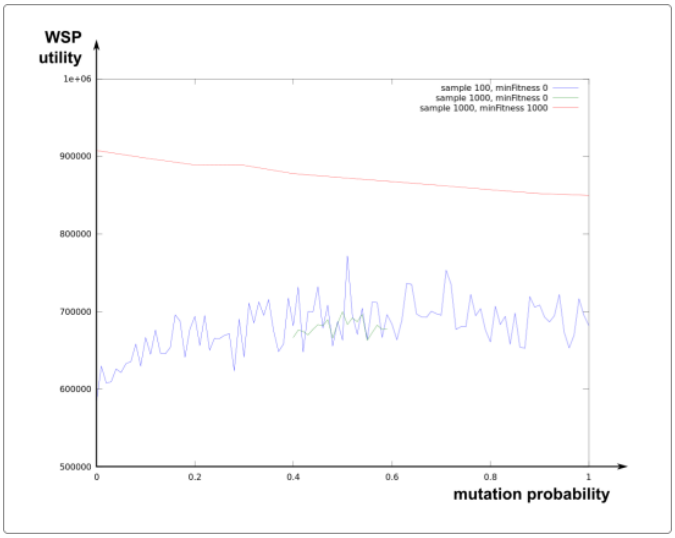


Figure 79: WSP's utility values varying the mutation probability after calibrating the minimum fitness.

A.1.3 Population Size

The optimal population size is around 20 as can be seen in the graphics generated with the results of multiple experiments ran:

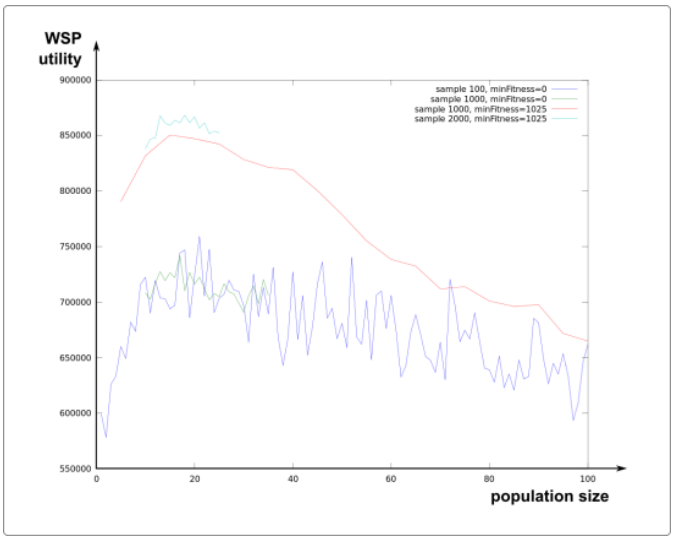


Figure 80: WSP's utility varying the population size in scenario A

A.1.4 Minimum Fitness

With regards to the minimum fitness, the experiment took values from 0 to 10^6, following a logarithmic scale. As we can see its optimal value is around 1000.

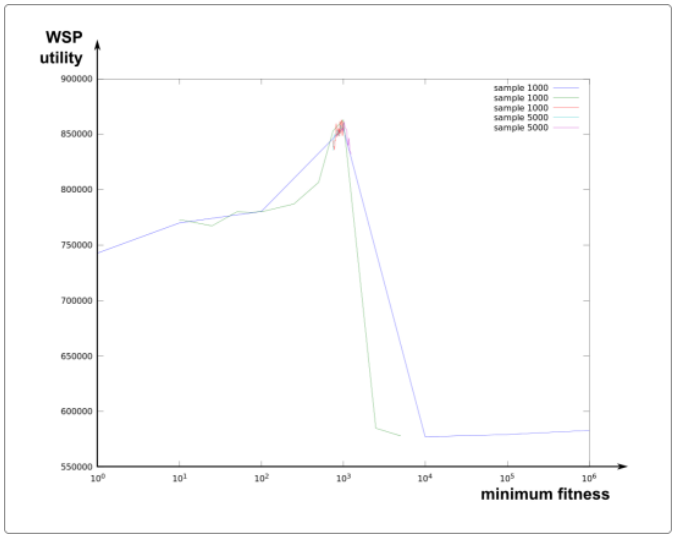


Figure 81: WSP's utility when varying the minimum fitness parameter on scenario A

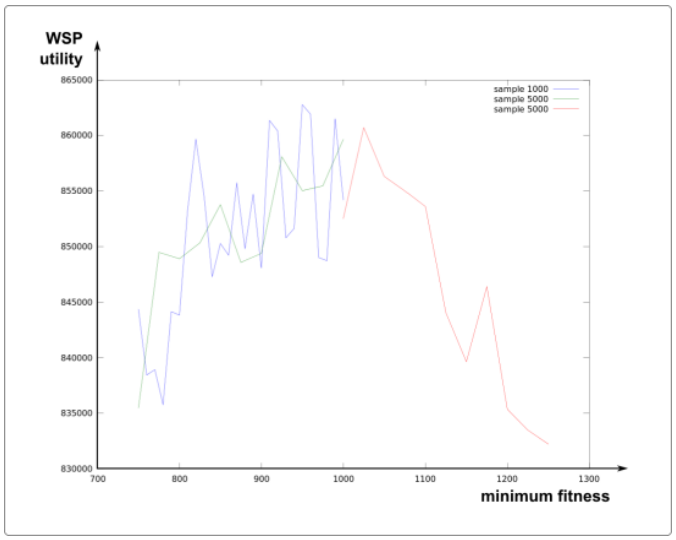


Figure 82: Close up of the range with optimal results

A.1.5 Summary

Parameter	Number of experiments	Optimal value
Increment	2	0.07 (around 0.1)
Mutation Probability	3	0
Population Size	4	18 (around 20)
Minimum Fitness	7	1025 (around 1000)
Crossover Probability	2	0.6 (irrelevant)
Fading	2	0.7 (irrelevant)

Table 8: Summary table of genetic algorithm parameters for scenario A

A.2 Scenario B

On scenario B, the parameters importance is quite similar to the ones in scenario A; first we have the mutation probability, the increment and the population size, followed by the minimum fitness, then the fading parameter and finally the crossover probability. As in

previous analysis, only the most relevant parameters are described in the next subsections, whereas the other ones do not influence in the WSP's utility.

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	1,05E-12	0	9,68E-01	1,05E-12	7,99E-01	1,70E-01
Test statistic F	239,5336	295,6341	0,1396	1975,9949	0,4133	1,6755

Table 9: ANOVA test values for the genetic algorithm on scenario B

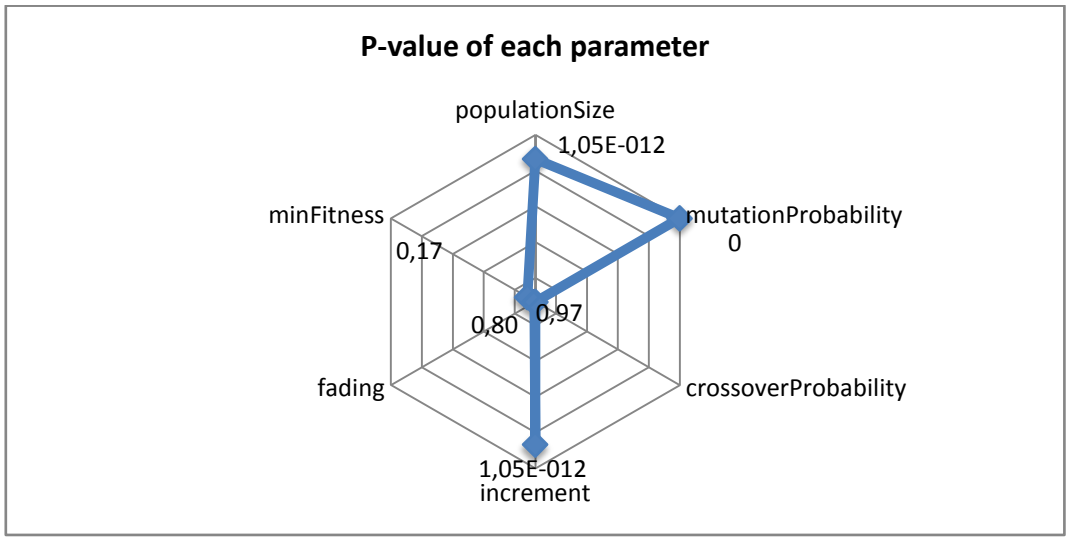


Figure 83: P-value of each parameter on scenario B

A.2.1 Mutation Probability

In this case, clearly, the greater is the mutation probability, the greater is the utility obtained by the WSP.

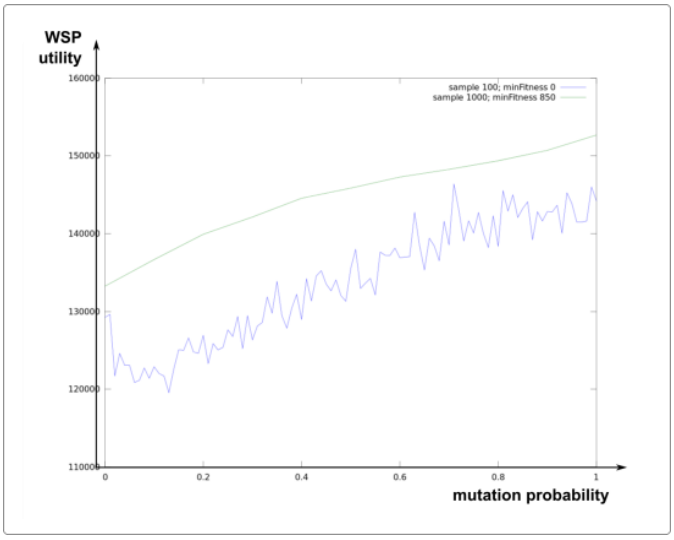


Figure 84: WSP's utility varying the mutation probability in scenario B

A.2.2 Increment

Similarly, the greater the increment, the greater is the utility obtained, which implies that the algorithm will behave quite randomly, as the mutation probability is high and also the increment parameter.

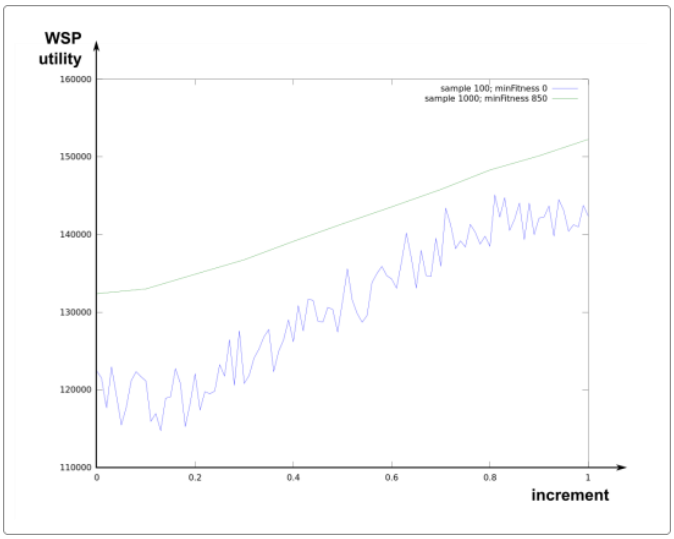


Figure 85: WSP's utility varying the increment parameter in scenario B

A.2.3 Population Size

Regarding the population size, the optimal value is between 50 and 100 as can be seen in the image below:

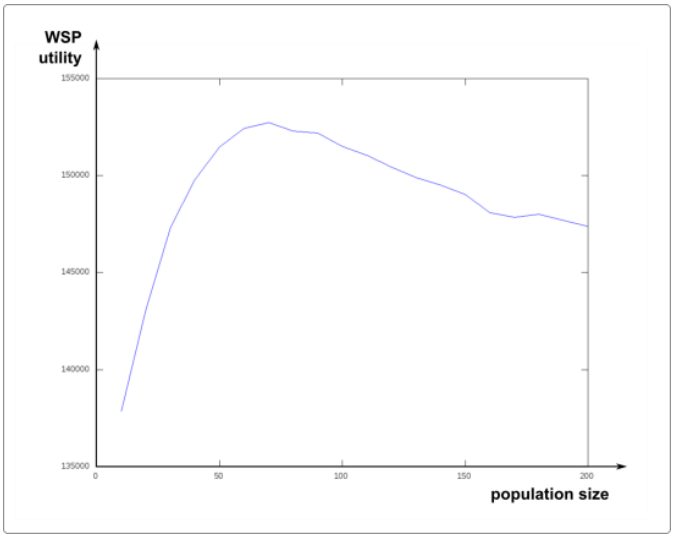


Figure 86: WSP's utility varying the population size in scenario B

A.2.4 Minimum Fitness

The minimum fitness is optimal around 1000, concretely 850 is the optimal value found during the simulations.

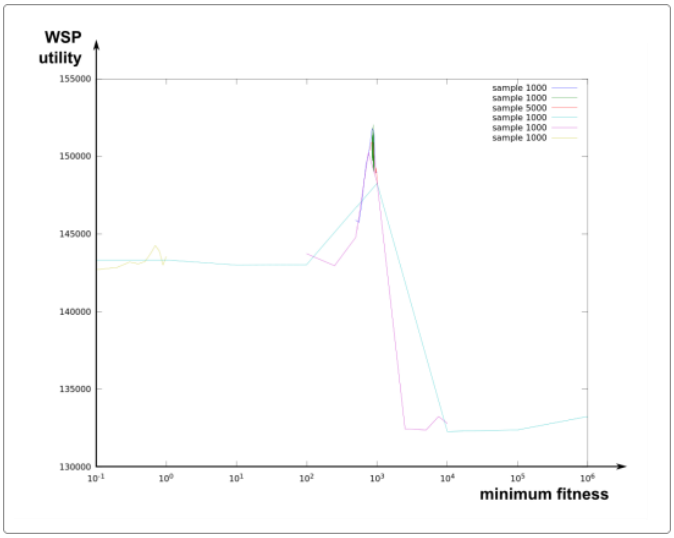


Figure 87: WSP's utility varying the minimum fitness in scenario B

A.2.5 Summary

Parameter	Number of experiments	Optimal value
Increment	2	1
Mutation Probability	2	1
Population Size	4	70 (between 50 and 100)
Minimum Fitness	8	850 (around 1000)
Crossover Probability	2	0.9 (irrelevant)
Fading	2	0.4 (irrelevant)

Table 10: Summary table of genetic algorithm parameters for scenario B

A.3 Genetic C

The first analysis shows the following ANOVA test values for each variable:

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	1.18E-12	0	0.99551	0	0.93369	1.78E-10
Test Statistic F	131.4255	4115.5522	0.0489	23858.95	0.2088	16.1211

Table 11: ANOVA test values for the genetic algorithm on scenario C

Therefore, in this scenario, both the increment and mutation probability are the most relevant parameters. Next, there are the population size and the minimum fitness. And again, neither the fading nor the crossover probability has too much influence in the results, so they have not been included in this appendix.

A.3.1 Increment

As can be seen in the graph below, higher values for the increment parameter grant higher revenues to the WSP.

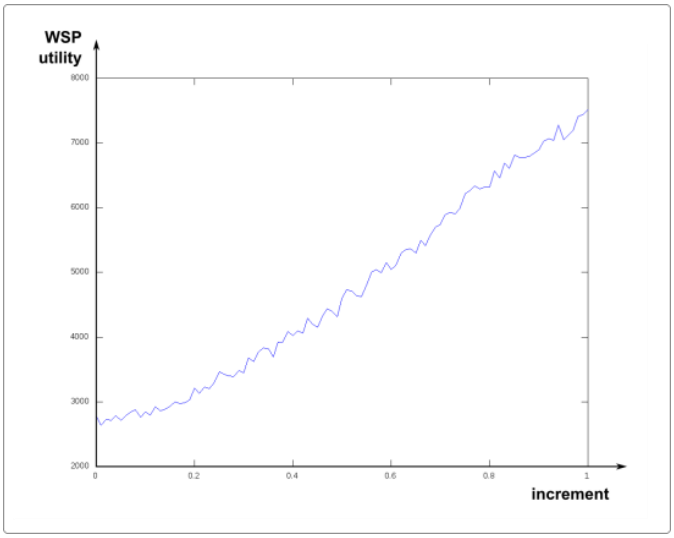


Figure 88: WSP's utility varying the increment

A.3.2 Mutation Probability

Similarly, higher probabilities of mutation, grant higher revenues.

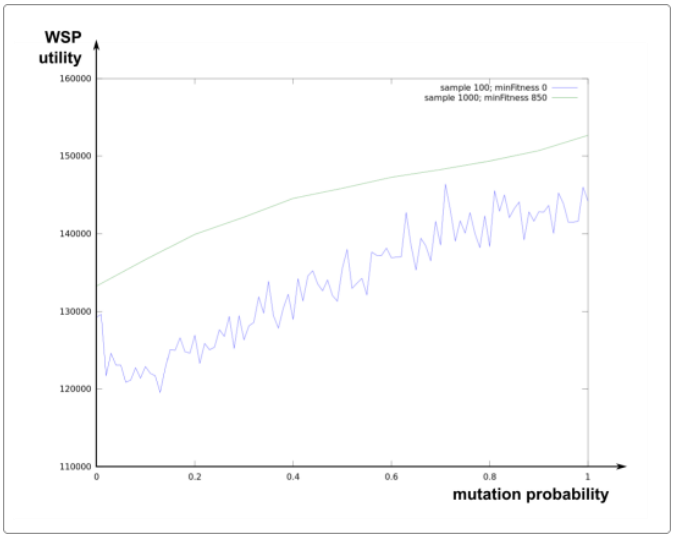


Figure 89: WSP's utility varying the mutation probability

A.3.3 Population Size

The optimal value of the population size is around 500, as can be seen in the graph below.

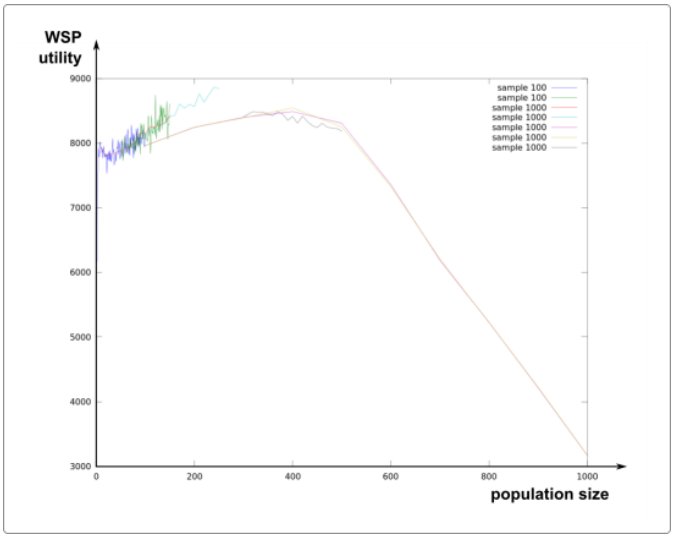


Figure 90: WSP's utility varying the population size

A.3.4 Minimum Fitness

Regarding the minimum value a strategy can have as fitness, we can observe that better revenues are obtained as long as it is lower than a threshold value around 100.

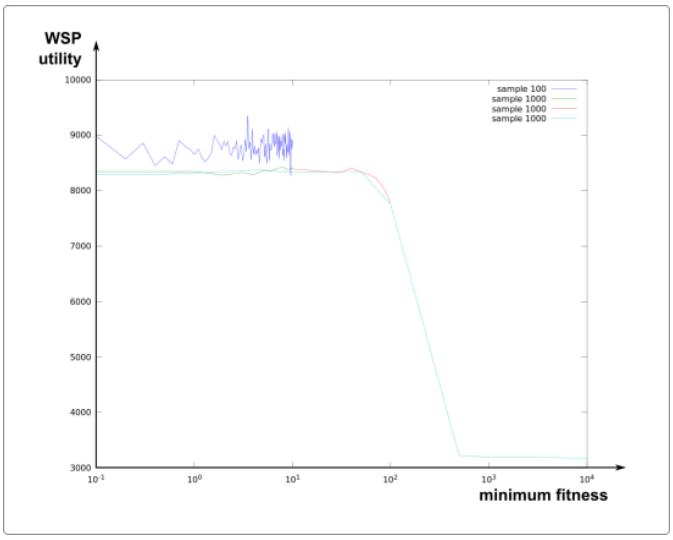


Figure 91: WSP's utility varying the minimum fitness

A.3.5 Summary

Parameter	Number of experiments	Optimal value
Increment	1	1
Mutation Probability	1	1
Population Size	7	400
Minimum Fitness	4	8 (lower than 40)
Crossover Probability	2	0.6 (irrelevant)
Fading	2	1 (irrelevant)

Table 12: Summary table of genetic algorithm parameters for scenario C

A.4 Scenario D

The ANOVA analysis of the scenario D shows that the most relevant parameters are the population size, the increment and the mutation probability; while the other 3 parameters are almost irrelevant in the revenue of the WSP.

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	0	9.31E-13	0.60969	1.57E-12	0.86216	0.98537
Test Statistic F	187.1698	111.8513	0.6743	3801.5702	0.3238	0.0497

Table 13: ANOVA test values for the genetic algorithm on scenario D

A.4.1 Population Size

As can be seen in the graph below, the optimal value for the population size in this scenario is around 10.

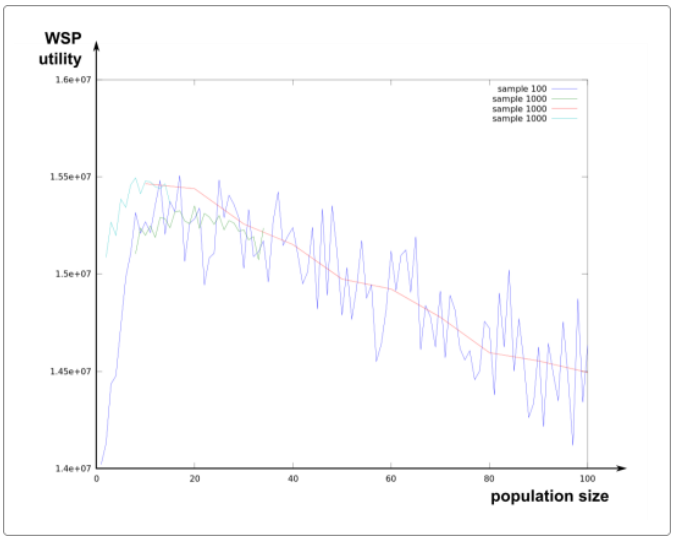


Figure 92: WSP's utility varying the population size

A.4.2 Mutation Probability

The mutation probability must be higher than 0.4 for obtaining optimal values of revenue.

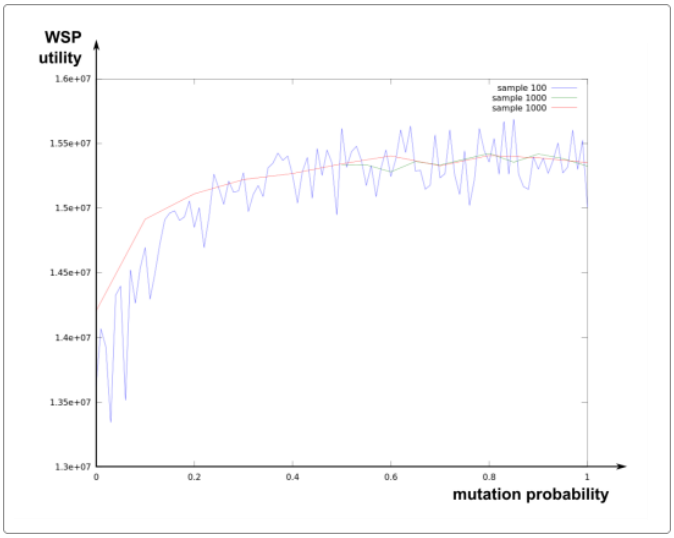


Figure 93: WSP's utility varying the mutation probability

A.4.3 Increment

However, even though the strategies may mutate quite often, the variations must be around the 10% to obtain better revenues as seen in Figure 94.

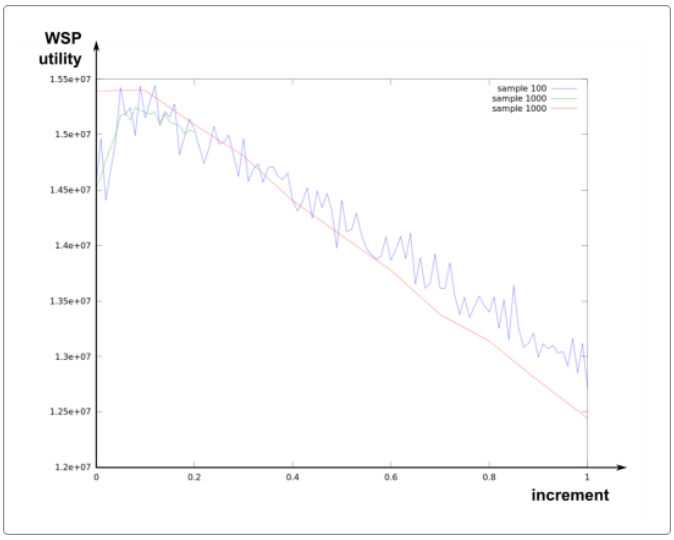


Figure 94: WSP's utility varying the increment

A.4.4 Minimum Fitness

Again in scenario D, the minimum fitness must be below a threshold (10^4 in this case) to obtain better revenues.

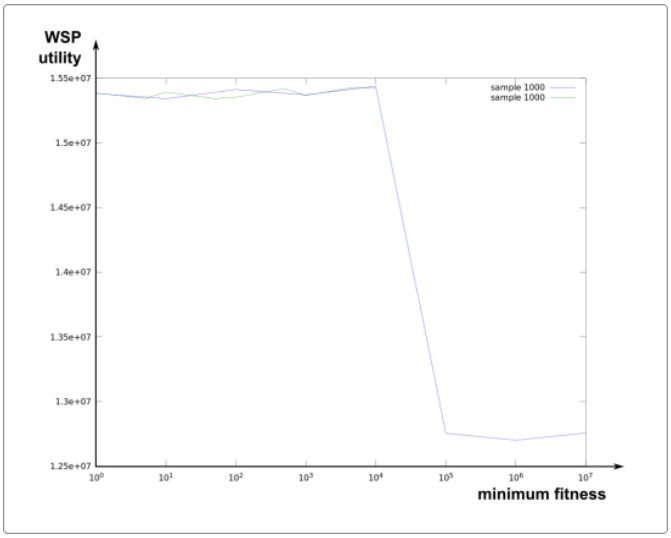


Figure 95: WSP's utility varying the minimum fitness

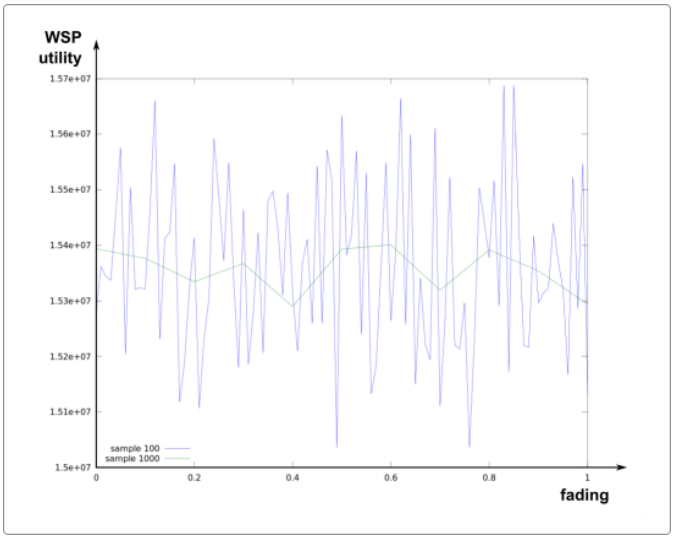


Figure 96: WSP's utility varying the fading

A.4.5 Summary

Parameter	Number of experiments	Optimal value
Increment	3	0.1
Mutation Probability	3	0.8 (greater than 0.4)
Population Size	4	8
Minimum Fitness	2	5000 (lower than 10^4)
Crossover Probability	2	0.6 (irrelevant)
Fading	2	0.6 (irrelevant)

Table 14: Summary table of genetic algorithm parameters for scenario D

A.5 Scenario E

The first ANOVA analysis of scenario E reveals that the two more decisive parameters are the increment and the mutation probability. Also the population size and the minimum fitness

value may have some relevance and the crossover probability and the fading parameter are the more irrelevant parameters.

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	0.096736	1.96E-11	0.65619	0	0.4745	0.40282
Test Statistic F	2.3358	14.0036	0.6089	162.535	0.8806	0.9761

Table 15: ANOVA test values for the genetic algorithm on scenario E

A.5.1 Increment

In this case, the revenues obtained for lower values of the increment are quite better than the revenues obtained when the increment parameter is higher.

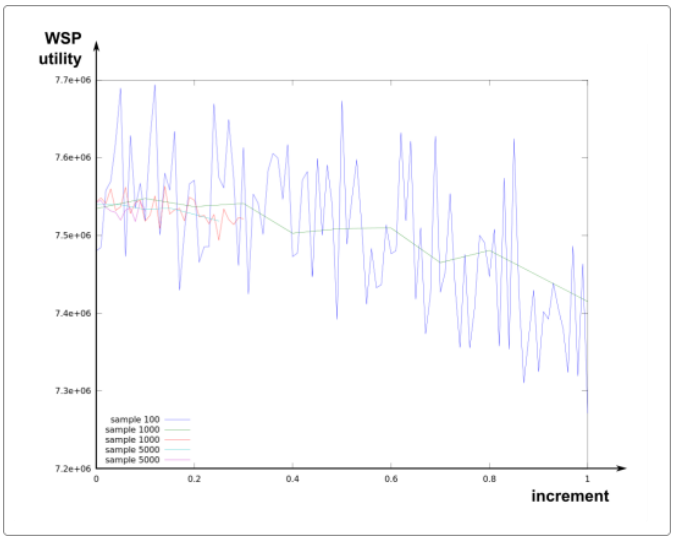


Figure 97: WSP's utility varying the increment

A.5.2 Mutation Probability

After selecting a low value for the increment (0.01), as we can observe the effect of the mutation probability is faded, so it does not have too much relevance in the WSP's revenue.

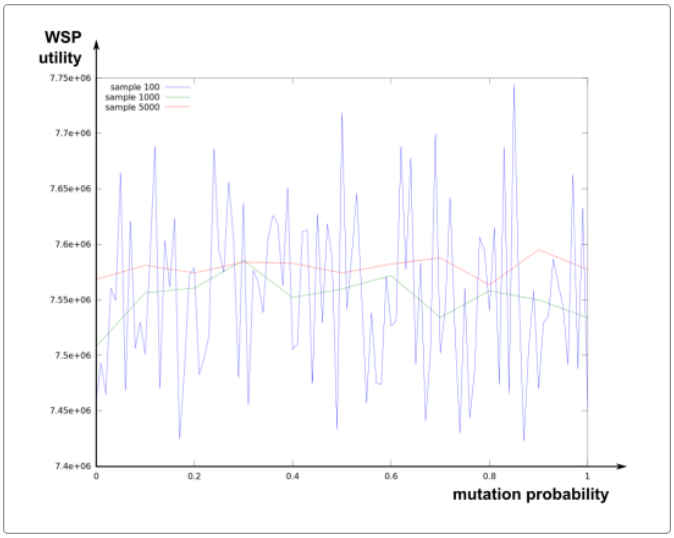


Figure 98: WSP's utility varying the mutation probability

A.5.3 Population Size

Regarding the population size, as can be seen in Figure 99, lower values grant better revenues.

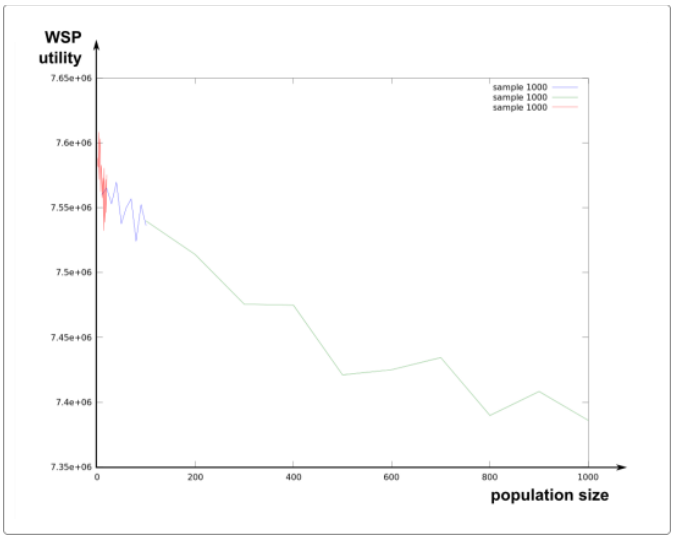


Figure 99: WSP's utility varying the population size

A.5.4 Minimum fitness

The minimum fitness value acts similarly in scenario E than in previous analysed scenarios: it must be below a threshold (around 1000) to grant better revenues.

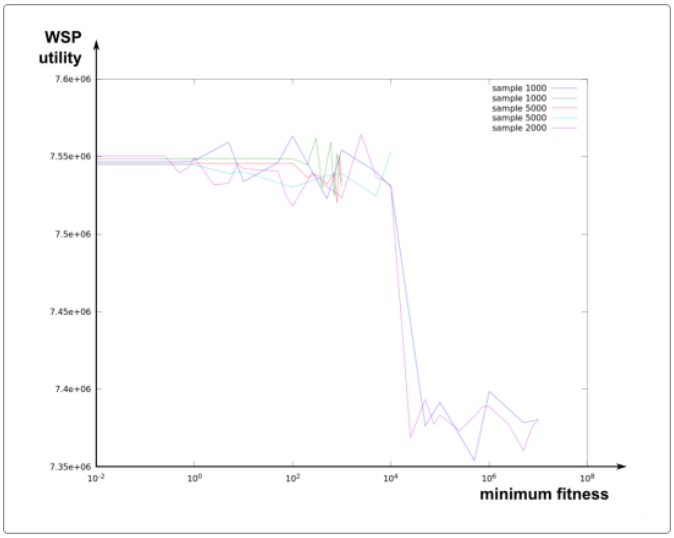


Figure 100: WSP's utility varying the minimum fitness

A.5.5 Summary

Parameter	Number of experiments	Optimal value
Increment	5	0.01 (low values)
Mutation Probability	3	0.9 (irrelevant when increment is low)
Population Size	4	4 (low values)
Minimum Fitness	6	900 (lower than 10^4)
Crossover Probability	2	0.3 (irrelevant)
Fading	2	0.3 (irrelevant)

Table 16: Summary table of genetic algorithm parameters for scenario E

A.6 Scenario F

Increment is the most relevant parameter in scenario F, followed by the population size. Then there is the mutation probability and the rest of the parameters do not seem to have much relevance in the utility of the service provider.

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	0.0058294	0.055367	0.61142	0	0.5989	0.68702
Test Statistic F	5.1452	2.3102	0.6719	25.8824	0.6898	0.4931

Table 17: ANOVA test values for the genetic algorithm on scenario F

A.6.1 Increment

The revenue of the service provider increases slightly when the increment parameter increases, as can be seen in the figure below.

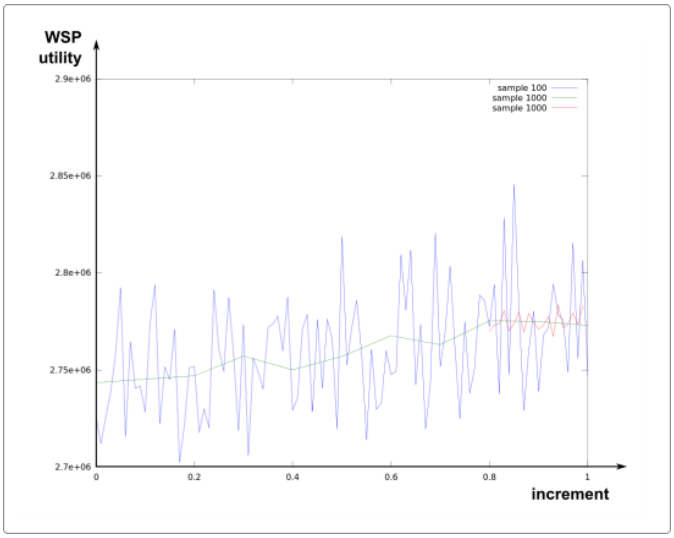


Figure 101: WSP's utility varying the increment

A.6.2 Population Size

On the other hand, lower population sizes seem to generate better revenues, as can be seen in the graph below.

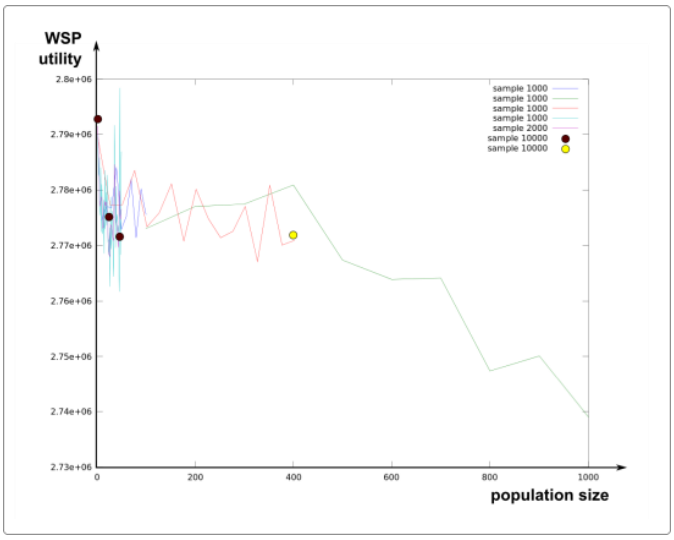


Figure 102: WSP's utility varying the population size

A.6.3 Minimum fitness

Again, we can see the same effect of the minimum fitness already explained on the previous scenarios, where is better to keep it under a threshold.

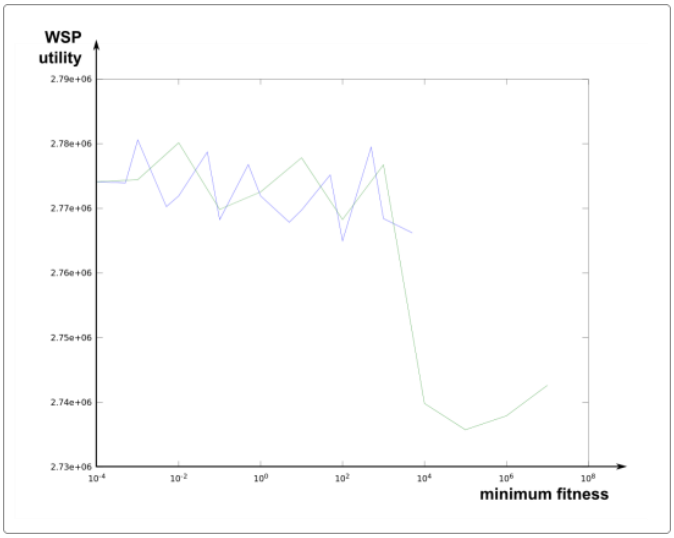


Figure 103: WSP's utility varying the minimum fitness

A.6.4 Summary

Parameter	Number of experiments	Optimal value
Increment	3	0.94 (high values)
Mutation Probability	2	0.8 (high values)
Population Size	8	2 (low values)
Minimum Fitness	2	0.001 (lower than 10^4)
Crossover Probability	2	0.3 (irrelevant)
Fading	2	0.3 (irrelevant)

Table 18: Summary table of genetic algorithm parameters for scenario F

A.7 Genetic G

On scenario G, the increment is the most relevant parameter followed by the mutation probability and the population size. The rest of the parameters do not seem too important in the first analysis conducted.

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	0.096736	1.96E-11	0.65619	0	0.4745	0.40282
Test Statistic F	2.3358	14.0036	0.6089	162.535	0.8806	0.9761

Table 19: ANOVA test values for the genetic algorithm on scenario G

A.7.1 Increment

Two different simulation experiments were run; the first one shows how the optimal value for the increment is around 0.1. In the second experiment we obtain a better close-up on values around 0.1.

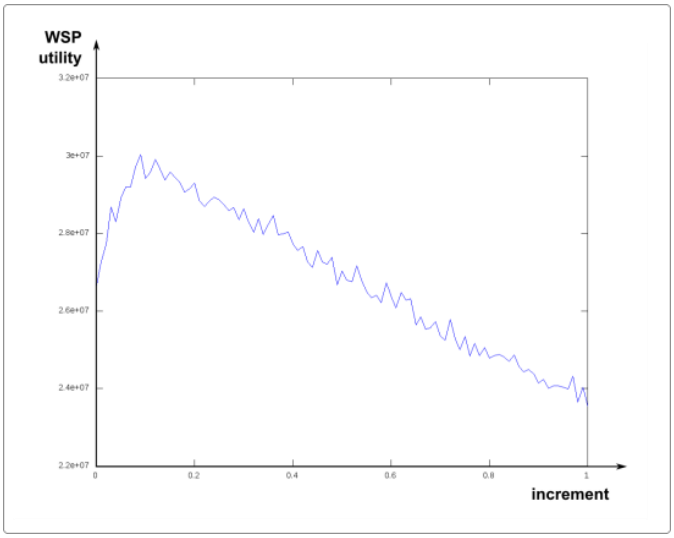


Figure 104: WSP's utility varying the increment

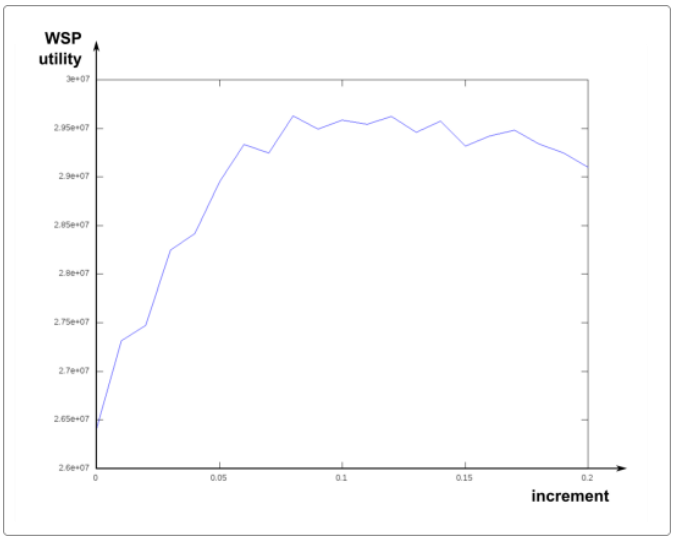


Figure 105: WSP's utility close up for low values of increment

A.7.2 Mutation Probability

As seen in the figure below, higher values of the mutation probability lead to higher WSP's utilities.

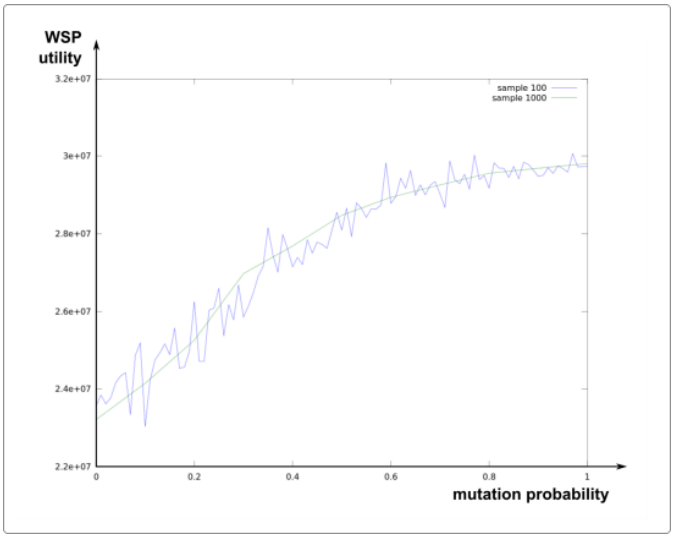


Figure 106: WSP's utility varying the mutation probability

A.7.3 Population Size

As regards to the population size, its optimal value is around 10.

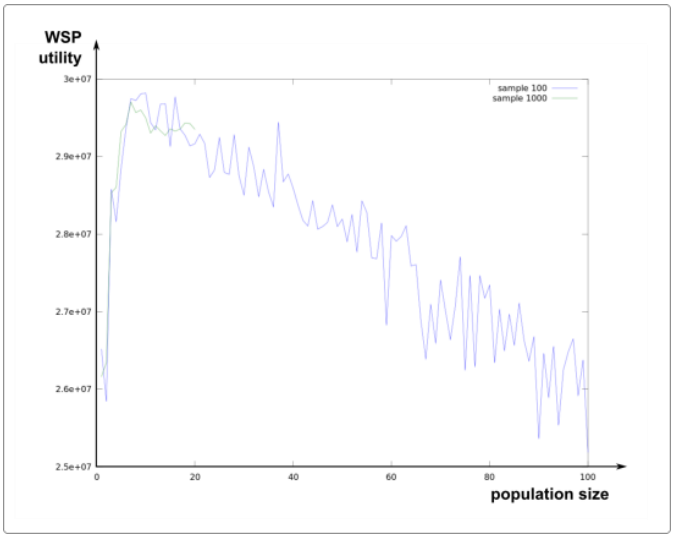


Figure 107: WSP's utility varying the population size

A.7.4 Minimum Fitness

The threshold for the minimum fitness value on scenario G is 10^4.

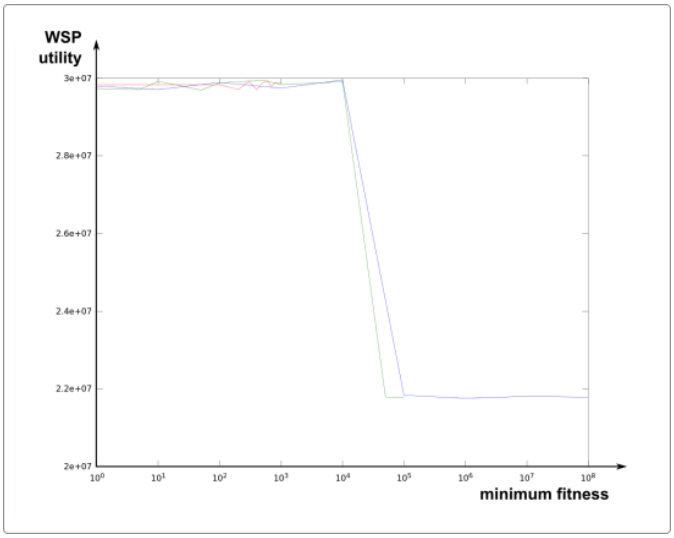


Figure 108: WSP's utility varying the minimum fitness

A.7.5 Summary

Parameter	Number of experiments	Optimal value
Increment	2	0.08
Mutation Probability	2	1
Population Size	2	7
Minimum Fitness	3	600 (lower than 10^4)
Crossover Probability	2	0.5 (irrelevant)
Fading	2	0.6 (irrelevant)

Table 20: Summary table of genetic algorithm parameters for scenario G

A.8 Scenario H

The first analysis shows the following ANOVA test values for each variable:

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	0	0	0,5736	0	0,30285	0,63228
Test statistic F	98,2783	49,7769	0,7266	646,5017	1,213	0,5737

Table 21: ANOVA test values for the genetic algorithm on scenario H

As we can see, parameters population size, mutation probability and increment are the more relevant ones in this scenario. Then there is the fading, followed by the crossover probability and minimum fitness is the last one.

A.8.1 Population Size

The service provider utility seems to increase quickly, obtaining the best results when the population size is around 10, and afterwards it decreases slowly for greater values of population size as can be seen in the following graph.

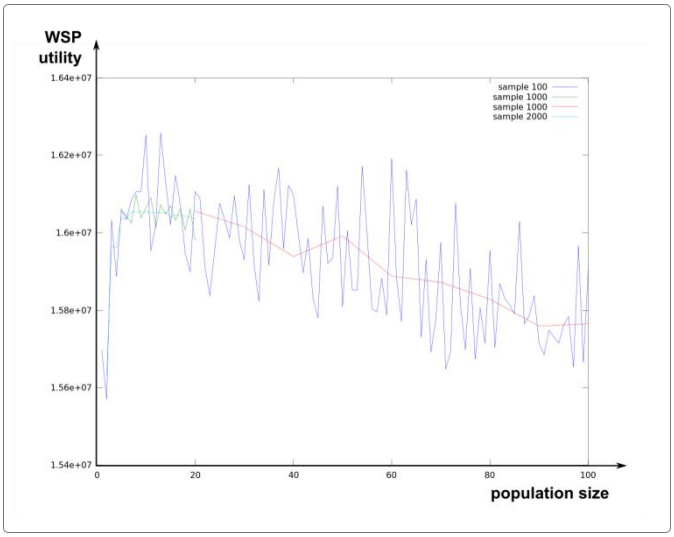


Figure 109: WSP's utility varying the population size in scenario H

A.8.2 Mutation Probability

On the other hand, the mutation probability grants better revenues when it is high, probably due to the increment parameter is set to a low value.

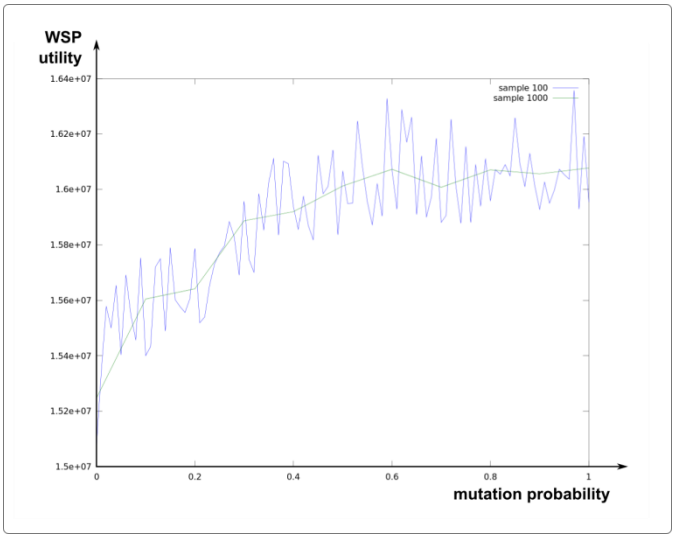


Figure 110: WSP's utility values varying the mutation probability on scenario H

A.8.3 Increment

Effectively, the best utilities are obtained when the increment is around 0.1.

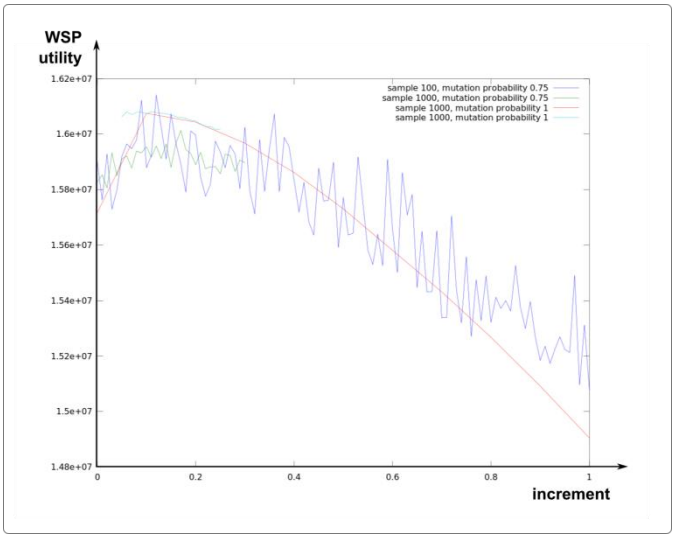


Figure 111: WSP's utility varying the increment parameter in scenario H, with two different values of the mutation probability

A.8.4 Minimum Fitness

The minimum fitness effects are the already commented in other scenarios. It must be set to a value lower than 10^4.

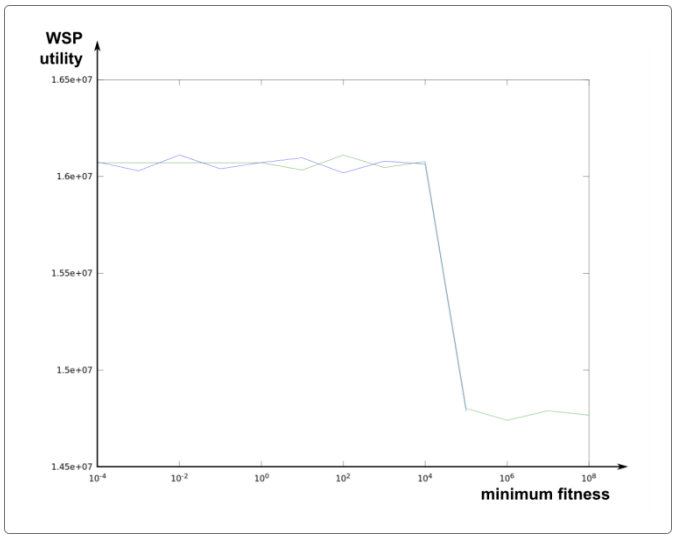


Figure 112: WSP's utility when varying the minimum fitness parameter on scenario H

A.8.5 Summary

Parameter	Number of experiments	Optimal value
Increment	4	0.11 (around 0.1)
Mutation Probability	2	1
Population Size	4	10
Minimum Fitness	2	0.01 (lower than 10^4)
Crossover Probability	3	0 (irrelevant)
Fading	2	0.3 (irrelevant)

Table 22: Summary table of genetic algorithm parameters for scenario H

A.9 Scenario I

The different parameters of the learning algorithm have little influence in the scenario I. The most significant one is the fading parameter followed by the population size. The rest of parameters seem insignificant in the first analysis done.

Parameter	Population Size	Mutation Probability	Crossover Probability	Increment	Fading	Minimum Fitness
P-value	0.33893	0.57195	0.68351	0.82935	0.15381	0.54226
Test Statistic F	1.082	0.729	0.5712	0.3712	1.67	0.716

Table 23: ANOVA test values for the genetic algorithm on scenario I

The analysis performed showed that only the population size and the minimum fitness are relevant on the WSP's revenue, as can be seen in the following subsections.

A.9.1 Population Size

After analysing the simulations run, seems that low values of population size lead to better revenues, however the results are not as clear as in other scenarios.

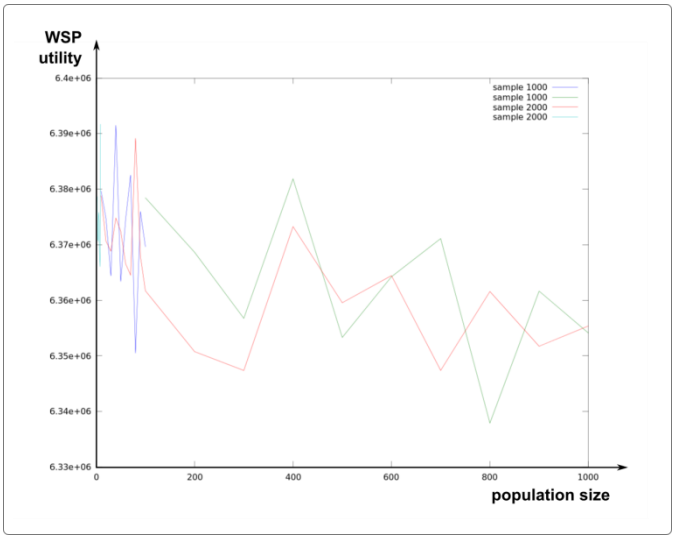


Figure 113: WSP's utility varying the population size

A.9.2 Minimum Fitness

Similarly, seems that the minimum fitness should be above 100, but the results are not as clear in this scenario.

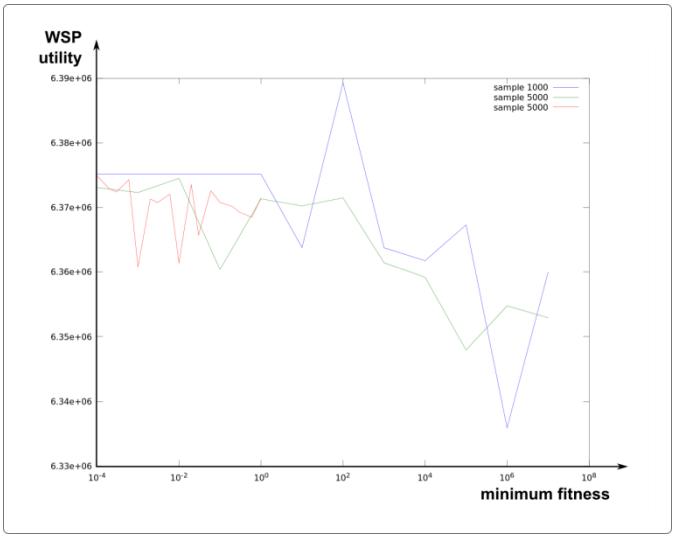


Figure 114: WSP's utility varying the minimum fitness

A.9.3 Summary

Parameter	Number of experiments	Optimal value
Increment	2	0.3 (irrelevant)
Mutation Probability	2	0.3 (irrelevant)
Population Size	5	9
Minimum Fitness	3	0.0001 (lower than 100)
Crossover Probability	2	0.3 (irrelevant)
Fading	2	0.3 (irrelevant)

Table 24: Summary table of genetic algorithm parameters for scenario I

A.9.4 Conclusions

Considering a single variable in the learning algorithm, the most important variables are the mutation probability and the increment. On the other hand the crossover probability and the fading are irrelevant in every scenario. Regarding the population size, most of the scenarios preferred a low value, allowing the simulation to complete more learning cycles (as a learning cycle last as many market cycles as elements in the population). And the minimum fitness usually should be lower to the average revenue obtained by the service provider.

Therefore if we consider the most relevant parameters, the mutation probability and the increment, they define the variability of the strategies, i.e. if the strategies of one cycle will be similar o completely different to the previous cycle. As we can see, situations without competition (monopoly) or low risk (high-medium network load) prefer low variability in the strategies, leading to simulations that converge to a concrete set of strategies. On the other hand, when the competition is higher, the service provider prefers playing unexpected strategies, selecting high values for the increment and mutation probability parameters.

Appendix B Parameter study of the Reinforcement Algorithm

Similarly, this appendix studies the effects of the reinforcement algorithm's parameters over the WSP's utility. Subsections follow the same structure than the Genetic algorithm analysis.

B.1 Scenario A

The first analysis shows the following ANOVA test values for each variable:

Parameter	Precision	Max coefficient	Mean coefficient
P-value	4.17E-14	2.56E-01	0.9836
Test Statistic F	178.8957	1.3311	0.0965

Table 25: ANOVA test values for the reinforcement algorithm on scenario A

Therefore the major influence is the variable precision, followed by the max coefficient and the mean coefficient. As the effects of the last two parameters were probed not relevant, only the precision results are shown.

B.1.1 Precision

We run a first experiment of 100 repetitions using values from 5 to 1000 with a gap of 5 units among them and we obtained the following results:

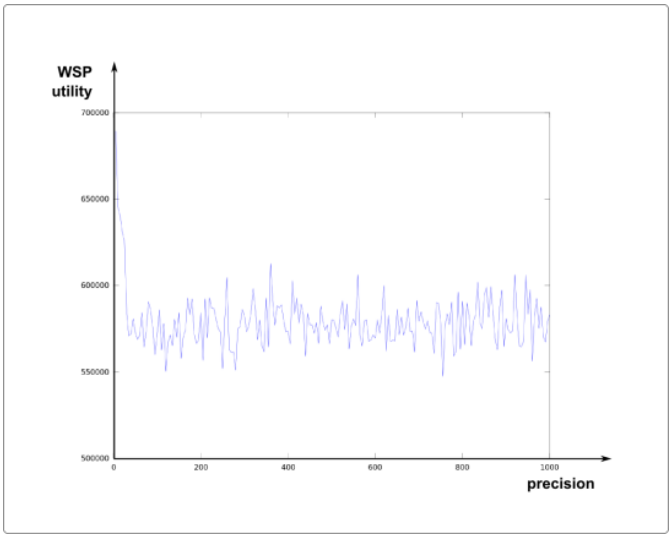


Figure 115: WSP's utility varying the precision

As we can see the major influence is with lower levels of precision, so we run a new set of simulations, this time from 1 to 35, each repeated 1000 times:

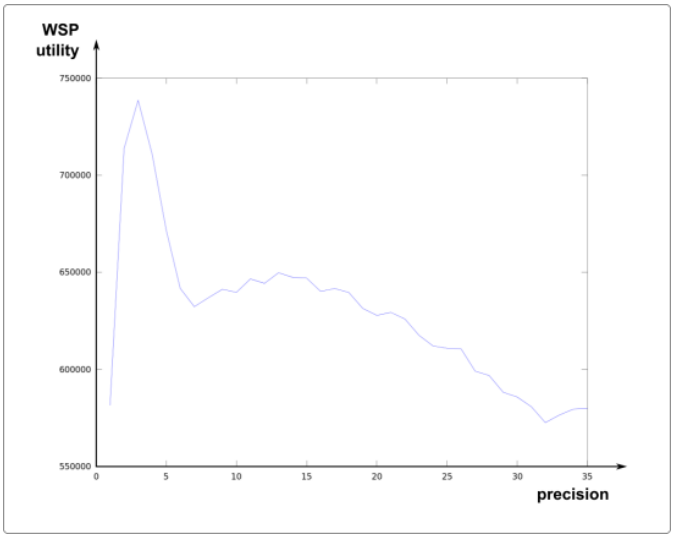


Figure 116: WSP's utility close up for low values of precision

In this case we obtained a clear optimal value for the precision which is 3.

B.1.2 Summary

Parameter	Number of experiments	Optimal value
precision	2	3 (low values)
max coefficient	6	0
mean coefficient	6	0.1 (bigger than 0)

Table 26: Summary table of reinforcement algorithm parameters for scenario A

B.2 Scenario B

As in scenario A, firstly we analyse the effects of each variable, by means of the ANOVA analysis:

Parameter	Precision	Max coefficient	Mean coefficient
P-value	2.5535E-14	0.2115	0.42665
Test Statistic F	159.3117	1.4605	0.9628

Table 27: ANOVA test values for the reinforcement algorithm on scenario B

Like in scenario A, precision has the biggest influence, followed by the max coefficient; and the mean coefficient parameter is the last one.

B.2.1 Precision

In the first approach to analyse the effects of the precision, we have run a simulation where it takes values from 5 to 1000. We can see there is a lower threshold to obtain high revenues:

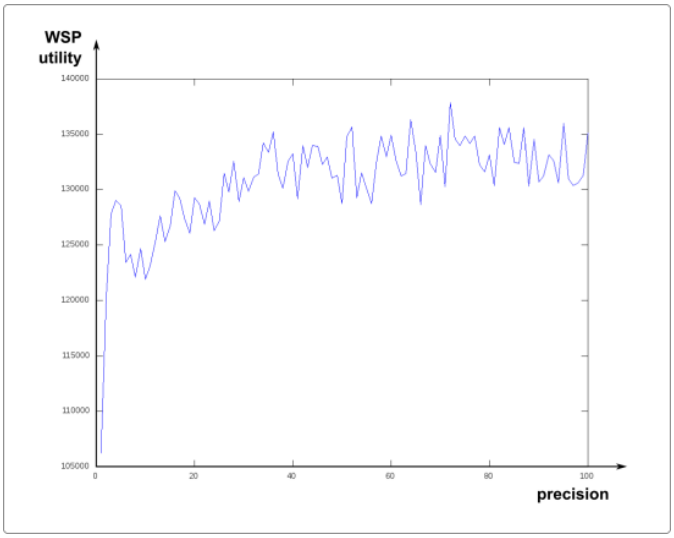


Figure 117: WSP's utility varying the precision

B.2.2 Summary

Parameter	Number of experiments	Optimal value
Precision	5	450 (greater than 50)
Max	6	1 (irrelevant)
Mean	3	0.67 (irrelevant)

Table 28: Summary table of reinforcement algorithm parameters for scenario B

B.3 Scenario C

The ANOVA analysis in scenario C sets the same importance than before over the parameters of the learning algorithm: (1) precision, (2) max coefficient and (3) mean coefficient.

Parameter	Precision	Max coefficient	Mean coefficient
P-value	0	0.53747	0.98316
Test Statistic F	1481.5155	0.7808	0.0979

Table 29: ANOVA test values for the reinforcement algorithm on scenario C

B.3.1 Precision

When analysing the effects of the parameter precision we obtain the following tendency:

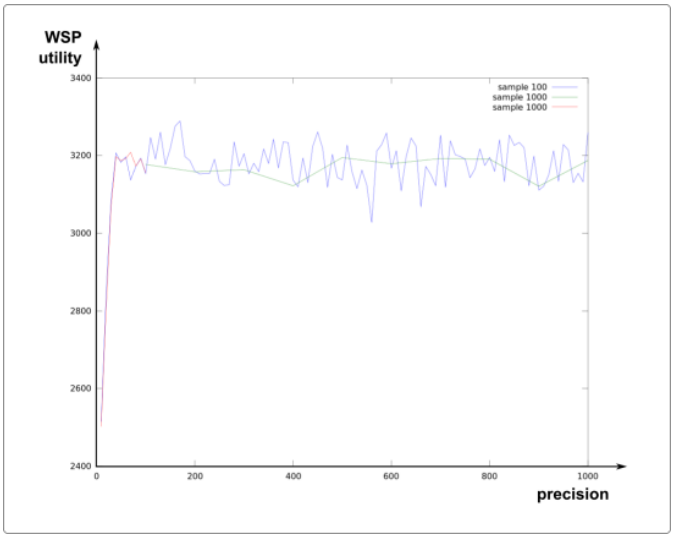


Figure 118: WSP's utility varying the precision

This clearly establishes a threshold value of the precision to obtain better utility values.

B.3.2 Summary

Parameter	Number of experiments	Optimal value
Precision	3	70 (greater than 50)
Max	2	0 (irrelevant)
Mean	2	0 (irrelevant)

Table 30: Summary table of reinforcement algorithm parameters for scenario C

B.4 Scenario D

We obtain the same results in the parameters importance order in the scenario D:

Parameter	Precision	Max coefficient	Mean coefficient
P-value	0	0.6082	0.92705
Test Statistic F	601.4618	0.6765	0.2206

Table 31: ANOVA test values for the reinforcement algorithm on scenario D

B.4.1 Precision

In a first analysis of precision parameter, we realise that lower values of precision obtain higher revenues:

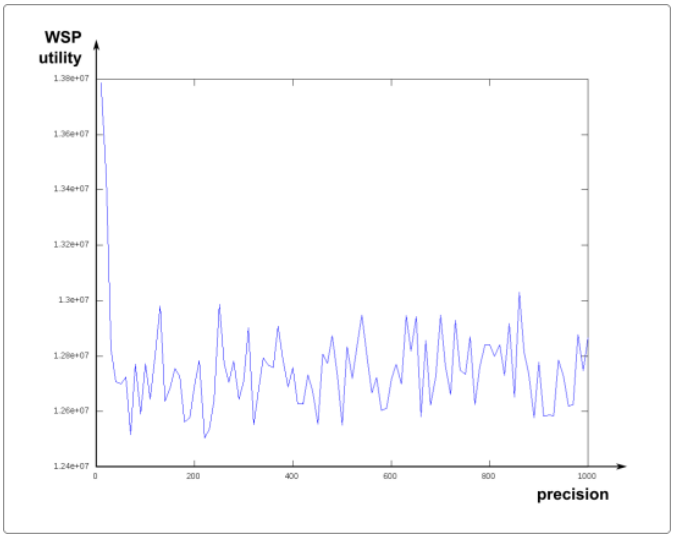


Figure 119: WSP's utility varying the precision

We obtain a better close-up in a second simulation:

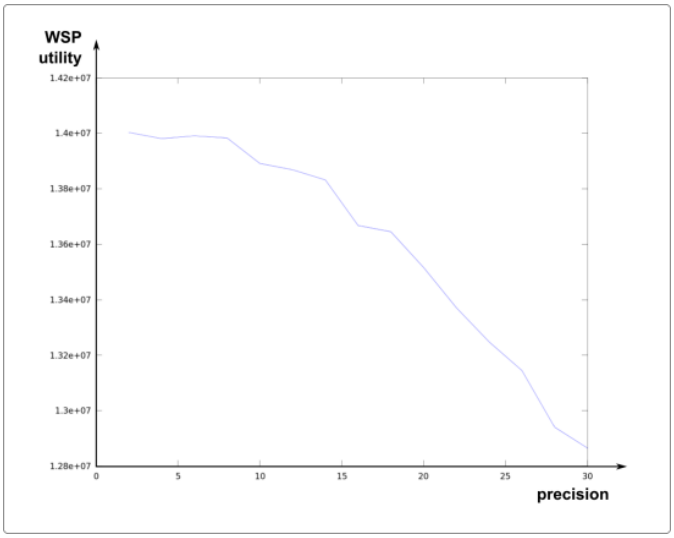


Figure 120: WSP's utility close up for low values of precision

Where we can see the optimal value is the lowest one (2).

B.4.2 Summary

Parameter	Number of experiments	Optimal value
Precision	2	2 (low values)
Max	2	0.6 (irrelevant)
Mean	2	0.6 (irrelevant)

Table 32: Summary table of reinforcement algorithm parameters for scenario D

We present the ANOVA analysis, to show the little influence of the parameters:

#Experiment	Max 1	Max 2	Mean 1	Mean 2
step	0,01	0,1	0,01	0,1
repeat	100	1000	100	1000
P-value	0,06105	0,83836	0,094259	0,87399

Table 33: ANOVA analysis for the max and mean coefficients' simulations run

B.5 Scenario E

The first analysis reveals the parameters importance which is first the precision, followed by the max and finally the mean coefficient.

Parameter	Precision	Max coefficient	Mean coefficient
P-value	0	0,1141	0,9278
Test Statistic F	21,3894	1,8629	0,2193

Table 34: ANOVA test values for the reinforcement algorithm on scenario E

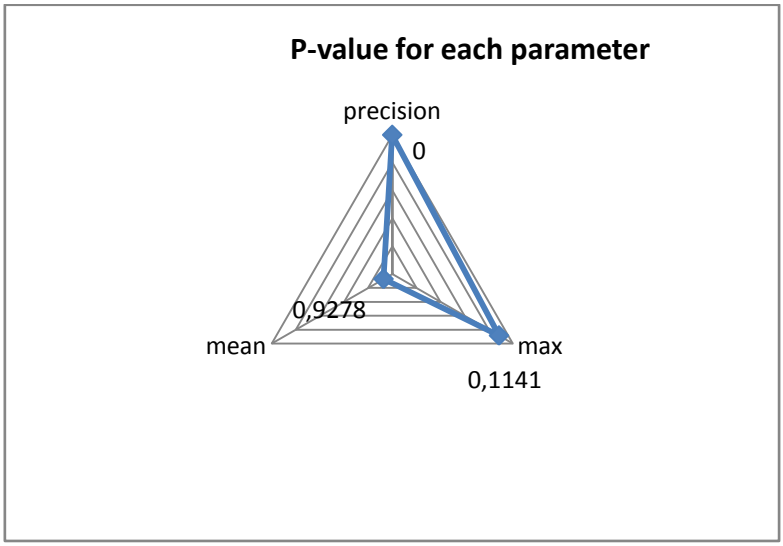


Figure 121: P-value of each parameter on scenario E

B.5.1 Precision

The optimal value for the precision is a low value, concretely 18, as can be seen in the graphs below.

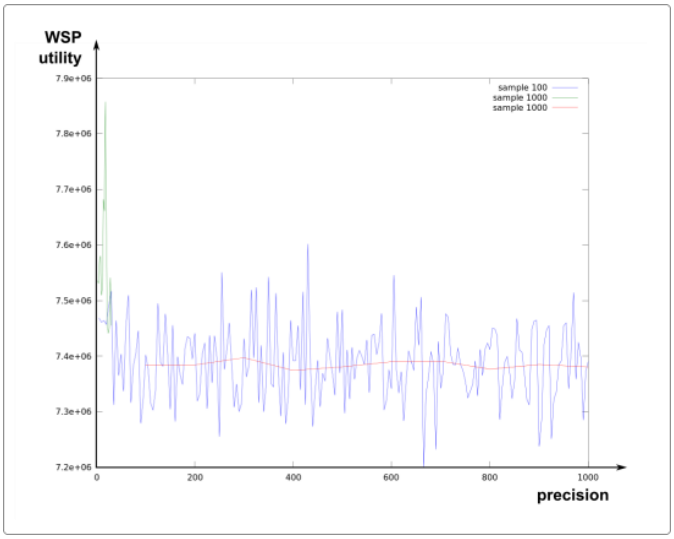


Figure 122: WSP's utility varying the precision

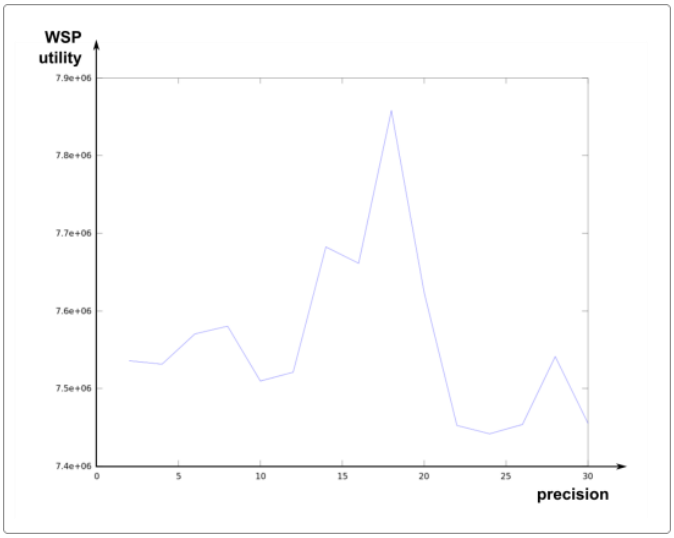


Figure 123: Close up for low values of precision

B.5.2 Summary

Parameter	Number of experiments	Optimal value
Precision	3	18
Max	2	0.6 (irrelevant)
Mean	2	0.6 (irrelevant)

Table 35: Summary table of reinforcement algorithm parameters for scenario E

B.6 Reinforcement F

This is the first scenario where the parameters importance seems different, as shown by the ANOVA analysis:

Parameter	Precision	Max coefficient	Mean coefficient
P-value	0,65865	0,037158	0,8928
Test Statistic F	0,4176	2,5533	0,2773

Table 36: ANOVA test values for the reinforcement algorithm on scenario F

B.6.1 Max coefficient

Even though the max coefficient seemed to have an important role in the revenue obtained by the WSP, posterior analysis show that it is not that meaningful:

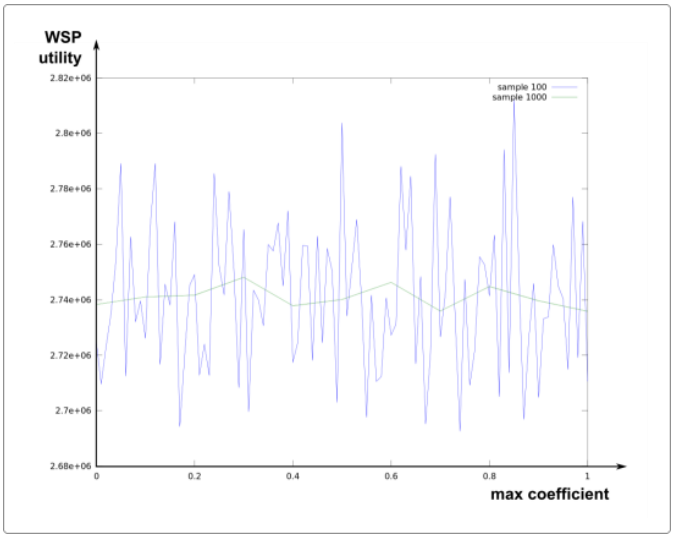


Figure 124: WSP's utility varying the max coefficient

As we can see in the ANOVA analysis, the influence of the parameters is almost unappreciable in this scenario:

Experiment	Precision 1	Precision 2	Precision 3	Max 1	Max 2	Mean 1	Mean 2
step	5	50	5	0,01	0,1	0,01	0,1
repeat	100	1000	1000	100	1000	100	1000
P-value	0,18016	0,99716	0,99569	0,05557	0,98472	0,05563	0,98472

B.6.2 Summary

Parameter	Number of experiments	Optimal value
Precision	3	20 (irrelevant)
Max	2	0.3 (irrelevant)
Mean	2	0.3 (irrelevant)

Table 37: Summary table of reinforcement algorithm parameters for scenario F

B.7 Scenario G

The ANOVA analysis over scenario G shows that the most influent parameter in this case is the precision, followed by the mean coefficient and finally the max coefficient.

Parameter	Precision	Max coefficient	Mean coefficient
P-value	0	0,78758	0,6884
Test Statistic F	2596,0227	0,4293	0,5645

Table 38: ANOVA test values for the reinforcement algorithm on scenario G

B.7.1 Precision

As can be seen in the graphs, higher gains are obtained when the precision takes low values.

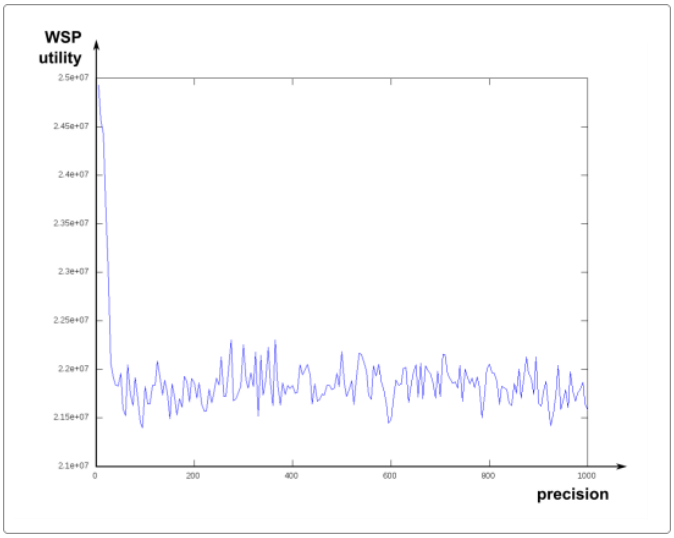


Figure 125: WSP's utility varying the precision

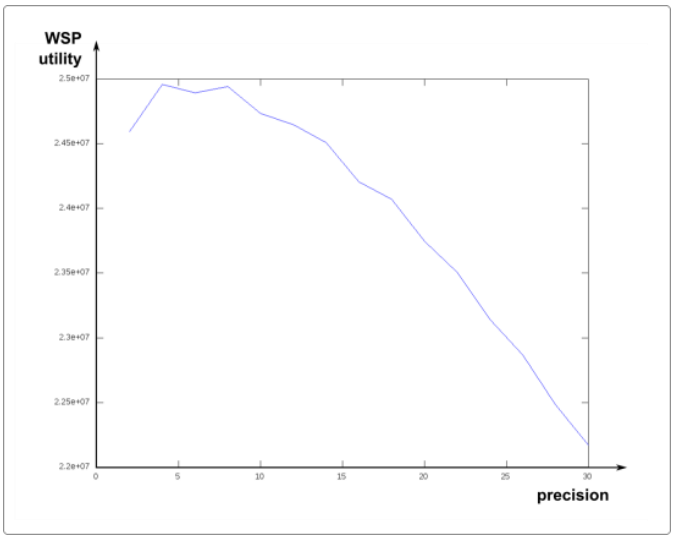


Figure 126: Close up for low values of the precision parameter

B.7.2 Mean coefficient

As regards to the mean coefficient, better revenues are obtained when it takes a higher value.

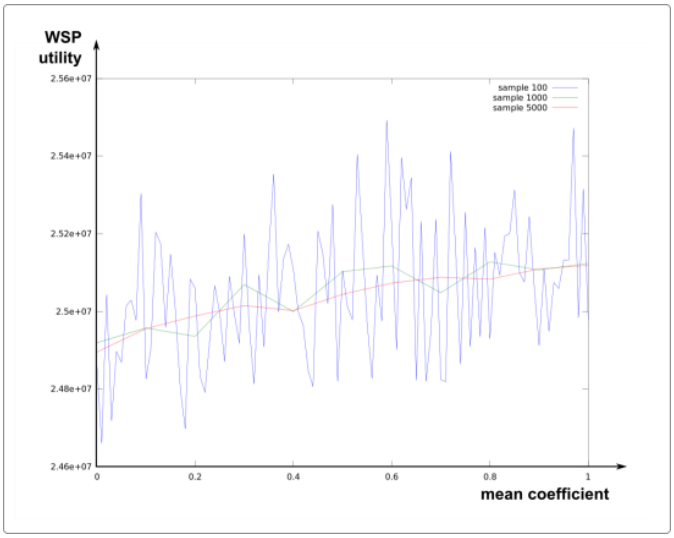


Figure 127: WSP's utility varying the mean coefficient

B.7.3 Max coefficient

On the other hand the max coefficient is better lower, which implies that the learning process is done over the mean revenue obtained for each strategy, rather than keeping track of the max revenue produced by a concrete strategy.

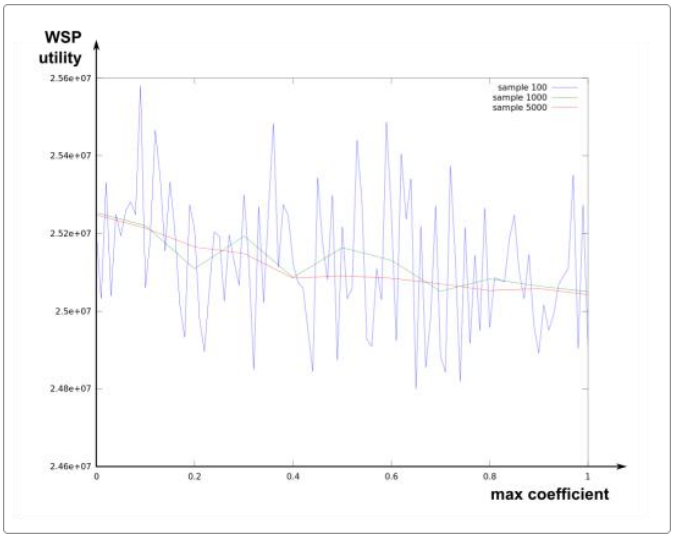


Figure 128: WSP's utility varying the max coefficient

B.7.4 Summary

Parameter	Number of experiments	Optimal value
Precision	3	4
Max	3	0
Mean	3	1

Table 39: Summary table of reinforcement algorithm parameters for scenario G

B.8 Scenario H

Also in scenario H, parameter precision takes the lead in the influence in the learning process, in this case followed by the max parameter and finally by the mean coefficient.

Parameter	Precision	Max coefficient	Mean coefficient
P-value	5,71E-014	0,41777	0,6566
Test Statistic F	335,9757	0,9787	0,6084

Table 40: ANOVA test values for the reinforcement algorithm on scenario H

B.8.1 Precision

As in previous scenario, lower values for precision provide higher revenues, concretely, 4 seems the optimal value.

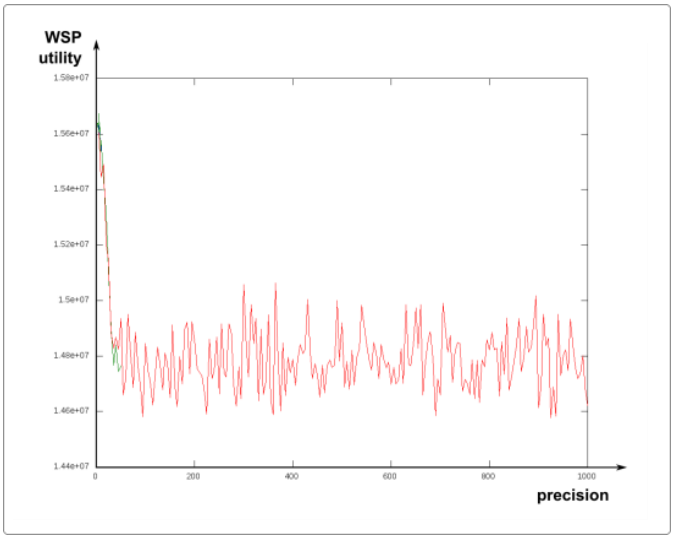


Figure 129: WSP's utility varying the precision

B.8.2 Summary

Parameter	Number of experiments	Optimal value
Precision	3	4
Max	2	0.3 (irrelevant)
Mean	2	0.3 (irrelevant)

Table 41: Summary table of reinforcement algorithm parameters for scenario H

B.9 Scenario I

Finally, scenario I also is mainly influenced by the parameter precision, followed by the max coefficient, and then the mean coefficient; as can be seen in the ANOVA table.

Parameter	Precision	Max coefficient	Mean coefficient
P-value	0,18797	0,33564	0,74053
Test Statistic F	1,6722	1,1401	0,4935

Table 42: ANOVA test values for the reinforcement algorithm on scenario I

However the next simulations shoed that there is no remarkable influence of the parameters in the WSP's utility:

#Experiment	Precision 1	Precision 2	Precision 3
step	5	50	5
repeat	100	1000	1000
P-value	0,007083	0,43721	0,42834

Table 43: ANOVA analysis of each experiment conducted for the precision parameter

B.9.1 Summary

Parameter	Number of experiments	Optimal value
Precision	3	900 (irrelevant)
Max	3	0 (irrelevant)
Mean	2	0.3 (irrelevant)

Table 44: Summary table of reinforcement algorithm parameters for scenario I

B.10 Conclusions

The most important parameter in the reinforcement learning is the precision or number of chunks each variable will be divided into. In cases with low competition, the service provider prefers low values of precision, which will allow the algorithm to be learn faster, as there are less elements to evaluate. With competition and low network load, however, it is preferred to have higher values for the precision.

Regarding the max and mean coefficient, they are not very important, but usually is better to give more relevance to the average revenue (mean coefficient = 1) than to the maximal revenue obtained by a strategy.

Appendix C Service Broker pricing function analysis

Next tables show the WSP's utility for each parameter of y and τ tested, and for each scenario so future users can carefully select its value. As we can see, as the number of users increase, the y and τ can take higher values without compromising the revenue of the WSPs. On the contrary, with higher competition (more service providers) the parameters value should be lower.

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	9,25E+05	7,05E+05	6,05E+05	5,56E+05	5,76E+05	7,36E+05	6,47E+05	5,45E+05
0,1	7,91E+05	7,49E+05	5,58E+05	5,83E+05	6,47E+05	6,45E+05	6,64E+05	7,28E+05
1	5,94E+05	7,29E+05	5,06E+05	5,43E+05	5,33E+05	6,65E+05	7,43E+05	-4,67E+06
2	8,68E+05	7,64E+05	7,23E+05	5,71E+05	6,06E+05	-7,54E+05	-1,13E+07	-1,16E+08
3	7,35E+05	4,39E+05	6,87E+05	5,36E+05	-2,28E+06	-2,63E+07	-2,67E+08	-2,67E+09
10	6,99E+05	-8,99E+12	-8,99E+13	-8,99E+14	-8,99E+15	-8,99E+16	-8,99E+17	-8,99E+18

Table 45: WSP's utility varying parameters y and τ in scenario A

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	1,63E+05	7,05E+05	6,05E+05	5,56E+05	5,76E+05	7,36E+05	6,47E+05	5,45E+05
0,1	7,91E+05	7,49E+05	5,58E+05	5,83E+05	6,47E+05	6,45E+05	6,64E+05	7,28E+05
1	5,94E+05	7,29E+05	5,06E+05	5,43E+05	5,33E+05	6,65E+05	7,43E+05	-4,67E+06
2	8,68E+05	7,64E+05	7,23E+05	5,71E+05	6,06E+05	-7,54E+05	-1,13E+07	-1,16E+08
3	7,35E+05	4,39E+05	6,87E+05	5,36E+05	-2,28E+06	-2,63E+07	-2,67E+08	-2,67E+09
10	6,99E+05	-8,99E+12	-8,99E+13	-8,99E+14	-8,99E+15	-8,99E+16	-8,99E+17	-8,99E+18

Table 46: WSP's utility varying parameters y and τ in scenario B

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	3,38E+03	3,52E+03	3,44E+03	2,95E+03	2,56E+03	3,37E+03	-2,33E+03	-4,73E+04
0,1	3,23E+03	3,72E+03	2,80E+03	3,10E+03	2,95E+03	2,82E+03	-1,57E+03	-4,66E+04
1	2,59E+03	3,06E+03	3,27E+03	2,89E+03	2,03E+03	-9,18E+03	-1,21E+05	-1,24E+06
2	3,90E+03	3,05E+03	3,27E+03	4,79E+01	-2,77E+04	-3,07E+05	-3,10E+06	-3,11E+07
3	3,70E+03	1,84E+03	-3,76E+03	-7,45E+04	-7,73E+05	-7,74E+06	-7,74E+07	-7,76E+08
10	3,63E+03	-4,73E+12	-4,70E+13	-4,70E+14	-4,70E+15	-4,73E+16	-4,70E+17	-4,73E+18

Table 47: WSP's utility varying parameters y and τ in scenario C

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	1,53E+07	1,25E+07	1,29E+07	1,14E+07	1,37E+07	1,36E+07	1,36E+07	1,31E+07
0,1	1,46E+07	1,47E+07	1,24E+07	1,36E+07	1,42E+07	1,44E+07	1,27E+07	1,40E+07
1	1,57E+07	1,41E+07	1,40E+07	1,36E+07	1,42E+07	1,47E+07	1,52E+07	1,31E+07
2	1,47E+07	1,59E+07	1,47E+07	1,40E+07	1,24E+07	1,31E+07	8,46E+06	-1,07E+08
3	1,54E+07	1,33E+07	1,59E+07	1,35E+07	1,15E+07	7,99E+06	-2,80E+08	-2,92E+09
10	1,48E+07	-1,24E+13	-1,24E+14	-1,24E+15	-1,24E+16	-1,24E+17	-1,24E+18	-1,24E+19

Table 48: WSP's utility varying parameters y and τ in scenario D

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	8,52E+06	6,87E+06	6,83E+06	6,16E+06	7,06E+06	7,13E+06	7,36E+06	7,19E+06
0,1	8,22E+06	7,99E+06	6,65E+06	6,96E+06	7,48E+06	7,82E+06	6,79E+06	7,30E+06

1	8,35E+06	7,81E+06	7,17E+06	7,33E+06	7,60E+06	7,75E+06	8,27E+06	5,55E+06
2	8,11E+06	8,63E+06	7,86E+06	7,57E+06	6,90E+06	6,59E+06	4,13E+06	-6,71E+07
3	8,18E+06	7,15E+06	8,55E+06	6,71E+06	5,36E+06	-1,29E+07	-1,81E+08	-1,88E+09
10	8,03E+06	-1,14E+13	-1,14E+14	-1,13E+15	-1,14E+16	-1,14E+17	-1,13E+18	-1,14E+19

Table 49: WSP's utility varying parameters y and τ in scenario E

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	3,14E+06	2,58E+06	2,41E+06	2,28E+06	2,52E+06	2,56E+06	2,68E+06	2,53E+06
0,1	3,03E+06	2,91E+06	2,47E+06	2,47E+06	2,76E+06	2,85E+06	2,48E+06	2,60E+06
1	2,97E+06	2,87E+06	2,48E+06	2,71E+06	2,80E+06	2,78E+06	2,98E+06	1,65E+06
2	3,00E+06	3,20E+06	2,74E+06	2,78E+06	2,55E+06	2,26E+06	-7,74E+05	-2,81E+07
3	2,99E+06	2,58E+06	3,06E+06	2,38E+06	1,78E+06	-5,59E+06	-7,54E+07	-7,76E+08
10	2,91E+06	-4,75E+12	-4,75E+13	-4,75E+14	-4,75E+15	-4,75E+16	-4,75E+17	-4,75E+18

Table 50: WSP's utility varying parameters y and τ in scenario F

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	2,51E+07	2,48E+07	2,36E+07	2,23E+07	2,52E+07	2,47E+07	2,31E+07	2,51E+07
0,1	2,76E+07	2,74E+07	2,69E+07	2,48E+07	2,58E+07	2,63E+07	2,58E+07	2,46E+07
1	2,62E+07	2,39E+07	2,48E+07	2,55E+07	2,53E+07	2,66E+07	2,59E+07	2,34E+07
2	2,36E+07	2,53E+07	2,64E+07	2,58E+07	2,32E+07	2,42E+07	1,87E+07	6,58E+06
3	2,63E+07	2,36E+07	2,73E+07	2,30E+07	2,15E+07	1,68E+07	-1,95E+08	-2,23E+09
10	2,57E+07	-5,42E+12	-5,42E+13	-5,42E+14	-5,42E+15	-5,42E+16	-5,42E+17	-5,42E+18

Table 51: WSP's utility varying parameters y and τ in scenario G

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	1,59E+07	1,55E+07	1,42E+07	1,33E+07	1,57E+07	1,53E+07	1,42E+07	1,53E+07
0,1	1,70E+07	1,72E+07	1,62E+07	1,51E+07	1,60E+07	1,63E+07	1,61E+07	1,46E+07
1	1,66E+07	1,45E+07	1,54E+07	1,61E+07	1,58E+07	1,60E+07	1,61E+07	1,34E+07
2	1,51E+07	1,56E+07	1,62E+07	1,57E+07	1,38E+07	1,42E+07	8,15E+06	-6,11E+07
3	1,62E+07	1,48E+07	1,74E+07	1,40E+07	1,23E+07	9,09E+06	-1,78E+08	-1,92E+09
10	1,62E+07	-1,18E+13	-1,18E+14	-1,18E+15	-1,18E+16	-1,18E+17	-1,18E+18	-1,18E+19

Table 52: WSP's utility varying parameters y and τ in scenario H

	y							
τ	0	0,00001	0,0001	0,001	0,01	0,1	1	10
0,01	6,47E+06	6,30E+06	5,69E+06	5,41E+06	6,30E+06	6,27E+06	5,80E+06	6,21E+06
0,1	7,03E+06	6,99E+06	6,56E+06	6,15E+06	6,51E+06	6,62E+06	6,62E+06	5,91E+06
1	6,82E+06	5,99E+06	6,26E+06	6,59E+06	6,48E+06	6,47E+06	6,60E+06	5,29E+06
2	6,19E+06	6,38E+06	6,54E+06	6,33E+06	5,64E+06	5,76E+06	2,65E+06	-2,45E+07
3	6,65E+06	6,06E+06	7,10E+06	5,58E+06	4,88E+06	-1,81E+06	-7,14E+07	-7,71E+08
10	6,63E+06	-4,72E+12	-4,72E+13	-4,71E+14	-4,72E+15	-4,72E+16	-4,71E+17	-4,73E+18

Table 53: WSP's utility varying parameters y and τ in scenario I

Appendix D User Manual

Our simulation model is programmed using the OMNeT++ framework; therefore, in a first instance, it would be convenient for the new user to become familiar with it. Loads of information, as well as manuals, can be found on their website[2]. Chapters 8 and 9 of their User Manual are a highly recommended lecture.

On this basic manual we will describe first the necessary inputs of the program, its outputs and finally how to run it. That means defining the parameters needed for the simulation; the files where the results are stored and its structure, and, finally, how to execute and configure the simulation.

D.1 Input Parameters

As described in chapter 5, to work properly, there are some parameters that must be set to run the simulation properly. These parameters can be volatile or not. If a parameter is volatile, it means that its value is re-read every time is needed during the simulation. Each module has a list of parameters that are described below.

D.1.1 EndUser

- *bitValue*: represents the value that has for the end user to transmit a basic bit of information. It is a volatile parameter.

D.1.2 SessionGenerator

- *datarate*: Is the data rate needed by the sessions generated by the session generator. It is a volatile parameter.
- *duration*: Is the duration of the sessions generated by the session generator. It is also a volatile parameter.
- *sessionArrivalTime*: Is the time since the last session finishes till the new session start. It is a volatile parameter.
- *filename*: Is the name of the file with the statistics of the multipliers that have to be applied to the base time.
- *dT*: Is the amount of time a coefficient from the file referenced by “filename” .

D.1.3 ServiceBroker

- *TTL*: time to live of a service request petition. Only the offers received within that time will be forwarded to the end user.

[2] http://omnetpp.org/documentation

D.1.4 SpectrumBroker

- x: parameter x of the pricing function.
- *y*: parameter y of the pricing function.
- *tau*: parameter τ of the pricing function.
- *TTL*: time to live since the Spectrum Broker notifies the WSPs till the last bandwidth request is admitted. Requests arriving later will be dismissed.
- *numCycles*: number of DSA cycles the simulation will last.
- *marketLifeCycle*: length of every DSA cycle.
- *bandwidth*: bandwidth available to the Spectrum Broker to allocate dynamically among the WSPs. It is a volatile parameter.
- *allocationAlgorithm*: it must be “fairness” or “proportional”. Selects the allocation algorithm that will be used.

D.1.5 WSP

- *learningModule*: it must be “Genetic” or “Reinforcement”, it selects the learning module that will be used.

D.1.6 WSPController

- *spectralEfficiency*: it is the spectral efficiency of the WSP.
- *minCost*: is the parameter Pricemin from its pricing function.
- *maxCost*: is the parameter Pricemax from its pricing function.
- *tau*: is the parameter τ from its pricing function.
- *minBW*: is the minimum amount of bandwidth the WSP will request to the Spectrum Broker.
- *maxBW*: is the maximum amount of bandwidth the WSP will request to the Spectrum Broker.

D.1.7 Genetic

- *populationSize*: is the number of strategies that will be considered simultaneously in the learning process.
- *mutationProbability*: is the probability that a strategy will mute.
- *increment*: percentage of the value a strategy is allowed to vary during its mutation.
- *crossoverProbability*: is the probability that two strategies will be combined to generate a new pair of strategies.
- *fading*: is the fading coefficient that will be used to update the fitness of a strategy.
- *minFitness*: is the minimum probability of a element from the population to be chosen by the selection algorithm.

D.1.8 Reinforcement

- *meanCoef* : is the weight factor that the mean utility obtained has in the valuation of a strategy.
- *maxCoef*: is the weight factor that the maximum utility obtained has in the valuation of a strategy.
- *precision*: is the precision that will be used to subdivide the variables of the learning process.

D.2 Output Results

OMNeT++ stores the results in two different files depending on their type. The results that are a vector of values are stored in a file whose extension is “vec”. On the other hand the values that are a scalar value are stored in a “sca” file. These files can be found in the results directory, which by default is “dsa/src/results”.

An example of a vec file could be Figure 130:

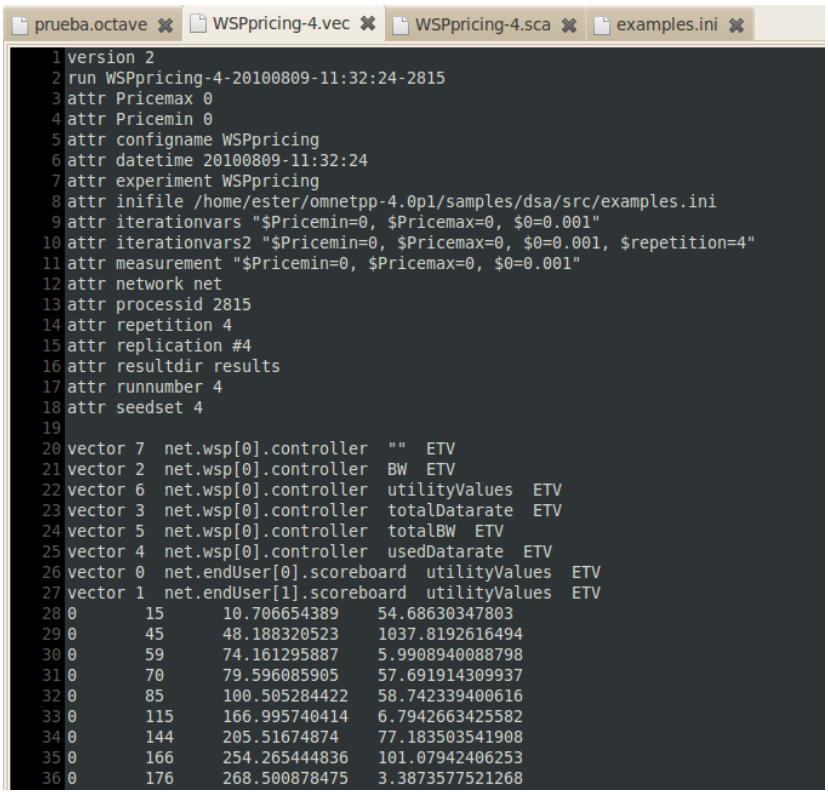


Figure 130: Example of vec file

As it can be seen, the file can be subdivided in three parts. The first part includes description of the parameters of the simulation. The second part describes the vectors that have been stored in the file; each entry is like the following:

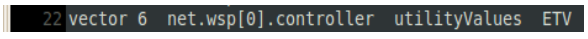


Figure 131: Line of a vec file defining a vector

First we have the keyword “vector”, implying that a vector is being defined. After, we have the index number that the simulation has assigned to that concrete vector. The next two parameters determine which vector is it, first it declares the module that has generated the vector (the controller of the WSP 0 in this case) and second the name of the vector. Finally we have some capital letters that define the format of every entry of the vector, in this case ETV, which stands for Event, Time, Value.

Finally we have the vector entries Figure 132.

```
31 0        70      79.596085905    57.691914309937
```

Figure 132: Line of a vec file with a vector entry.

First we have the index of the vector, which in this case is the vector 0. We can check on the previous section of the file that it corresponds to the utilityValues vector of user 0. After that, we have the Event, that is, the number of event that was being run when the value was stored. The third value is the simulation time, and finally the value that was stored.

To analyze the results obtained, there is the possibility of using the tool provided by OMNeT++: Scave. However, when dealing with a lot of large vec files, this tool is not the best solution, as it becomes quite slow. The solution we have used is based on shell scripting to extract the values and Octave to analyze and represent them.

For data extraction we have written a basic shell script:

```
#!/bin/bash
vector=$2
awk -v vector=$vector '{ if ($1==vector) print $3}' ../results/"$1" > ../results/$(echo "$1"|sed "s/\.vec/\.$2/")
```

Figure 133: getvector.sh

An example of how to use this shell script, parting from a vec file like Figure 134, to extract the vector 0, use the command:

```
ester@t0uchm3:~/omnetpp-4.0p1/samples/dsa/src/scripts$ ./getvector.sh WSPpricing-4.vec 0
```

Figure 134: How to invoke getvector.sh

It will generate a file called WSPpricing-4.0 Figure 135, where there are all extracted values one after the other. This file can be loaded into a Matlab/Octave array using the command load.

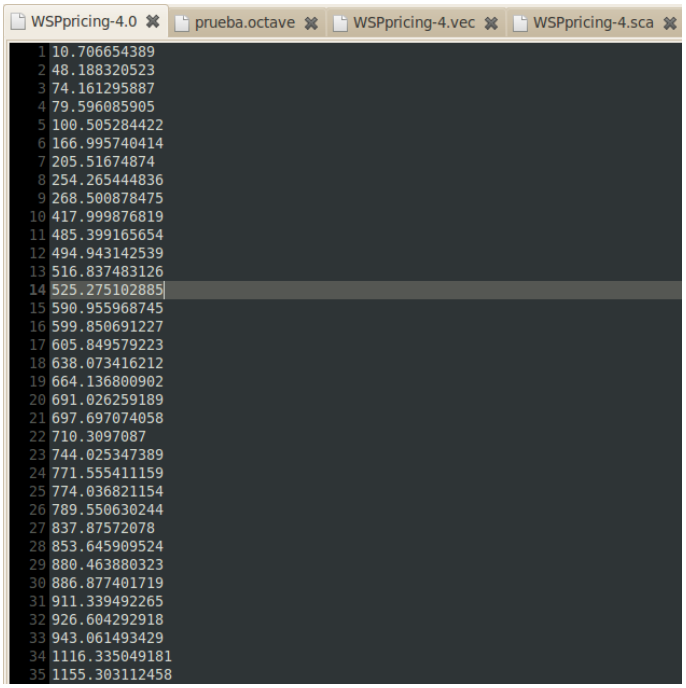


Figure 135: File generated by getvector.sh

Similarly for the cases where the time is important, you can use the getvector2.sh file, that extracts both the value and the time.

Other scripts have been developed to automate data extraction like runsimulation.sh, that runs several simulations, extracts some values and computes their means; reorderVector.m which loads the means generated by the previous and reorders it accordingly to the parameters studies; and so on.

The vectors that can be obtained from the simulation are:

- WSP
    - *totalDatarate*: is the allocated capacity to the WSP by the Spectrum Broker in bps.
    - *usedDatarate*: is the capacity that the WSP is currently using.
    - *totalBW*: is the allocated bandwidth by the Spectrum Broker in Hz.
    - *utilityValues*: is the utility obtained by the WSP every cycle.

  Moreover all the parameters included in the learning module are watched, and every cycle their values are stored. In this case the only parameter is the bandwidth requested to the Spectrum Broker (it does not have to be necessarily the same that is allocated afterwards), which is saved with the vector name BW.

- EndUser
    - *utilityValues*: is the utility obtained by every session served.

On the other hand we have the scalar values that are stored in the sca file.

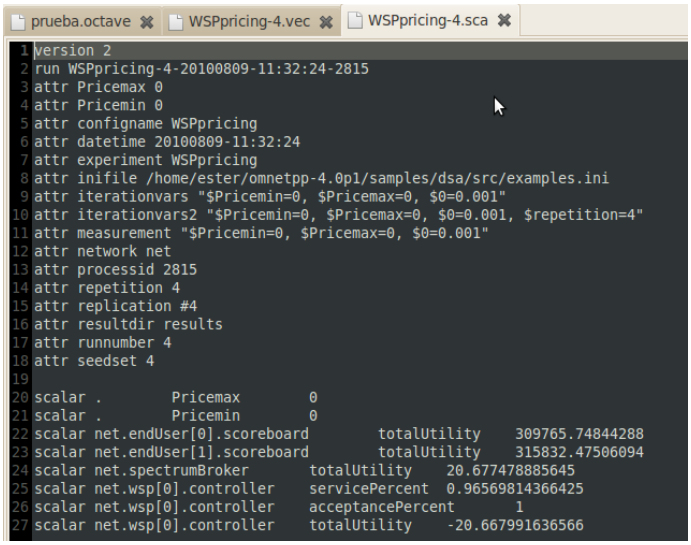


Figure 136: Example of a sca file

Like in the vec file we have first a description of the attributes of the simulation. Then there are the scalars values, preceded by the keyword “scalar”, followed by the module that stored the value and its name. Finally it is the value stored.

The scalar values that are stored by each simulation are:

- Spectrum Broker
  - *totalUtility*: is the total revenue obtained by the Spectrum Broker.
  - *totalBW*: is the total bandwidth allocated through the whole simulation (average allocated bandwidth can be obtained by dividing it by the number of cycles.
- WSP
  - *totalUtility*: is the total revenue obtained by the WSP.
  - *servicePercent*: is the quotient between the number of service requests received and the offers created.
  - *acceptancePercent*: is the quotient between the number of connections and the offers created.
- End User
  - *totalUtility*: is the total utility obtained by the end user.

D.3 Execution and Configuration File

When running the simulation, there is the possibility of running it with the Graphical User Interface (GUI), which is convenient for understanding the model and its mechanisms. However, the graphical simulation consumes a lot of resources. As a consequence, when running several simulations for parameters studies, it is preferred to use the command line environment, which runs the simulation without GUI.

The simulation can be run from the OMNeT++ IDE, from the menu Run -> Run Configurations… It will prompt a configuration dialog (Figure 137), where you can choose the configuration, the

run and the environment. Other options are the number of parallel runs and the configuration file to use (ini file).

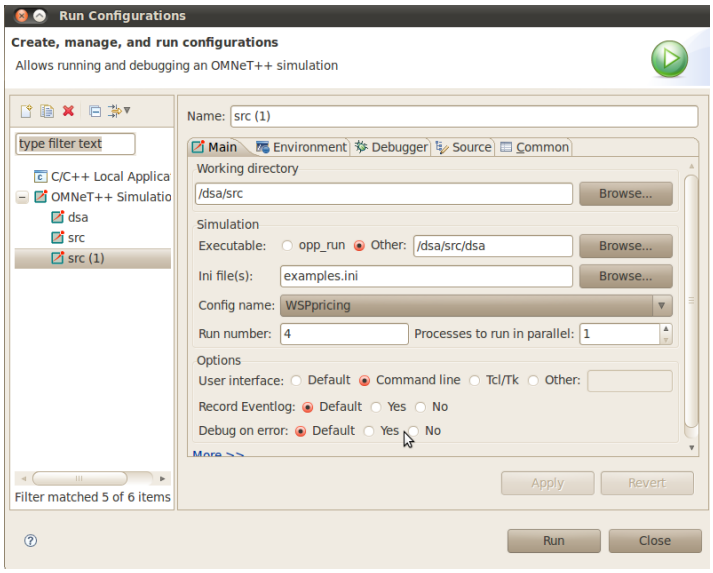


Figure 137: Run Configurations dialog

Otherwise you can run the simulation directly from the command line:

```
ester@t0uchm3: ~/omnetpp-4.0p1/samples/dsa/src
ester@t0uchm3:~/omnetpp-4.0p1/samples/dsa/src$ ./dsa -u Cmdenv -c WSPpricing -r 0 examples.ini
OMNeT++ Discrete Event Simulation  (C) 1992-2008 Andras Varga, OpenSim Ltd.
Version: 4.0p1, build: 090310-10709, edition: Academic Public License -- NOT FOR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer
Setting up Cmdenv...
Loading NED files from .: 14

Preparing for running configuration WSPpricing, run #0...
Scenario: $Pricemin=0, $Pricemax=0, $0=0.001, $repetition=0
Assigned runID=WSPpricing-0-20100809-11:55:45-3411
Setting up network `net'...
Initializing...
Initializing module net, stage 0
Initializing module net.endUser[0], stage 0
Initializing module net.endUser[0].controller, stage 0
Initializing module net.endUser[0].scoreboard, stage 0
Initializing module net.endUser[0].sessionGenerator, stage 0
Initializing module net.endUser[1], stage 0
Initializing module net.endUser[1].controller, stage 0
Initializing module net.endUser[1].scoreboard, stage 0
Initializing module net.endUser[1].sessionGenerator, stage 0
Initializing module net.serviceBroker, stage 0
Initializing module net.spectrumBroker, stage 0
Initializing module net.wsp[0], stage 0
Initializing module net.wsp[0].learning, stage 0
Initializing module net.wsp[0].controller, stage 0
Initializing module net.wsp[0].learning, stage 1

Running simulation...
** Event #1   T=0   Elapsed: 0.000s (0m 00s)
     Speed:     ev/sec=0   simsec/sec=0   ev/simsec=0
     Messages:  created: 5   present: 5   in FES: 5
** Event #58036   T=100000   Elapsed: 0.434s (0m 00s)
     Speed:     ev/sec=133721   simsec/sec=230415   ev/simsec=0.58035
     Messages:  created: 48275   present: 0   in FES: 0

<!> No more events -- simulation ended.


Calling finish() at end of Run #0...
```

Figure 138: Execution of the simulation from the command line

The command is:

```
ester@t0uchm3:~/omnetpp-4.0p1/samples/dsa/src$ ./dsa -u Cmdenv -c WSPpricing -r 0 examples.ini
```

Figure 139: Example of the command to run the simulation

The first option, “-u” selects the environment, type “Cmdenv” for the command line environment and “Tkenv” for the graphical one. “-c” selects the configuration that will be run, it must be a configuration defined in the configuration file. The run that will be executed can

be specified with the option “-r”, otherwise all the runs from the selected configuration will be run; to specify a range type “<initial_run>..<final_run>” after the run option (Figure 140). Finally you can select the configuration file that will be used. If no configuration file is specified, “omnetpp.ini” will be selected as default.

```
ester@t0uchm3:~/omnetpp-4.0p1/samples/dsa/src$ ./dsa -u Cmdenv -c WSPpricing -r 0..2 examples.ini
```

Figure 140: Another example of the simulation run command

To know the number of runs a configuration has you can use the command:

```
ester@t0uchm3:~/omnetpp-4.0p1/samples/dsa/src$ ./dsa -u Cmdenv -x WSPpricing examples.ini
OMNeT++ Discrete Event Simulation  (C) 1992-2008 Andras Varga, OpenSim Ltd.
Version: 4.0p1, build: 090310-10709, edition: Academic Public License -- NOT FOR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer
Setting up Cmdenv...

Config: WSPpricing
Number of runs: 40800

End.
```

Figure 141: Using the command line to obtain information about the simulation configurations

The configuration file is the file where all the input parameters are described. When running the simulation using the command line environment, all the parameters must be included in it, or it will stop, raising an error:

```
ester@t0uchm3:~/omnetpp-4.0p1/samples/dsa/src$ ./dsa -u Cmdenv -c WSPpricing -r 4 examples.ini
OMNeT++ Discrete Event Simulation  (C) 1992-2008 Andras Varga, OpenSim Ltd.
Version: 4.0p1, build: 090310-10709, edition: Academic Public License -- NOT FOR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer
Setting up Cmdenv...
Loading NED files from .: 14

Preparing for running configuration WSPpricing, run #4...
Scenario: $Pricemin=0, $Pricemax=0, $0=0.001, $repetition=4
Assigned runID=WSPpricing-4-20100809-11:36:39-3009
Setting up network `net'...

<!> Error in module (cCompoundModule) net (id=1): The simulation wanted to ask a question, set cmdenv-interactive=true to allow it: "Enter parameter `net.num
Users':".


End.
ester@t0uchm3:~/omnetpp-4.0p1/samples/dsa/src$
```

Figure 142: Undefined parameter error

On the Tk environment, however, if a parameter is not set on the configuration file, it will pop up a window asking for it:

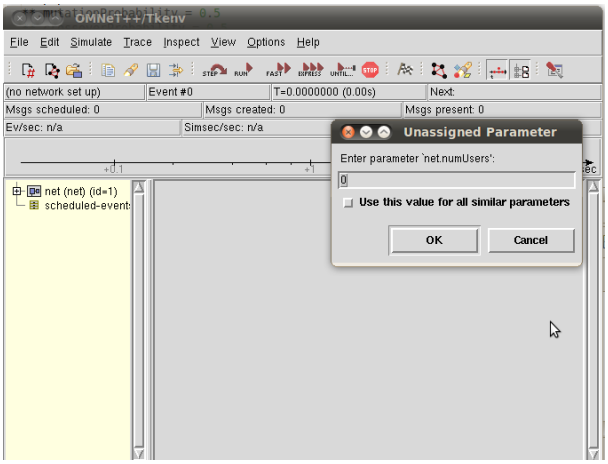


Figure 143: Dialog for unassigned parameters

On the same configuration file we can describe different situations to be simulated by the model, for instance, a case where there is only one WSP and another where there are 2. Each of these situations is called a configuration, and to describe them the configuration file is subdivided in sections.

The first section is the [General] section, which has all the common parameters that the other sections will inherit. After that section we have the named configurations, which are sections headed with the line [Config <name>]. Each of these sections corresponds to a different model situation as explained above, and it contains the set of parameters that describe this situation.

The syntax to assign a value to a parameter is the following:

```
**.<module_name>.<parameter_name>=<value>
```

If the module name does not need to be specified we can use simply:

```
**.<parameter_name>=<value>
```

To show the different possibilities, let's examine some examples:

```
**.bitValue=10
```

Will set the bitValue of all the users to 10.

```
**.endUser[0].bitValue=10
**.endUser[*].bitValue=15
```

Will set the bitValue of user 0 to 10, and all the others to 15.

```
**.endUser[1..10]=24
```

Will set the bitValue of users from 1 to 10 as 24.

Finally, we can use the random functions to define a value:

```
**.bitValue=uniform(0,15)
```

Which will make the bitValue vary during the simulation, taking values from 0 to 15, describing a uniform random variable.

If the parameter described with a random function is "volatile", then, the parameter value will change during the simulation, every moment it is consulted. If the parameter is not "volatile", then it will be fixed for every run, but it will vary among the different runs.

Some of different functions that can be used to describe random values are uniform, exponential, normal, etc.[3]

Finally to run parameters' studies you can tell the parameters to vary among a set of values:

```
**.bitValue=${0,10,15}
```

Will multiply per [check word] 3 the number of runs of the configuration, one with 0 as bitValue, the second with 10 and the third width 15.

```
**.bitValue = ${0..10 step 2}
```

Will generate runs with the bitValues 0, 2, 4, 6, 8 and 10.

If there are several parameters which values are multiple, the runs generated will have all the possible permutations.

An additional feature is the possibility to restrain the values to some constrain, for instance:

```
**.minCost=${Pricemin=0..15}
**.maxCost=${Pricemax=0..15}
constraint=$Pricemin<=$Pricemax
```

Will generate from all the possible permutations, only the ones where the minCost is less or equal than the maxCost.

Apart from the model parameters, there are other parameters about the simulation that can be set on the configuration file. The more important ones are:

- network: selects the model to be set up and run.
- repeat: establishes the number of repetitions each configuration will be executed.
- vector-recording: enables or disables the recording of a vector.
- result-dir: directory where the result files will be stored.

The easiest way to write a correct ini file is to take a correct one and modify the desired parameters.

[3] Check http://www.omnetpp.org/doc/omnetpp40/api/group__RandomNumbers.html for the whole list and especification.

Appendix E Programmer guide

To illustrate how to modify the simulation model, we will explain how to perform some simple changes in the model. The first example shows how to modify the model only changing the ned files. On the second example, the C++ code is modified as well, which implies changes on the behaviour of the model agents.

E.1 Example 1: Introducing 3 different types of traffic for the End User

The goal of this example is to be able to describe 3 totally different traffics for every EndUser. These traffic flows could represent (1) voice, (2) data and, (3) video, for instance, so their statistics are intrinsically different and must be defined separately. Therefore we need to include 3 session generators in the end user description, that is, the ned file. We can include two more modules through the graphic interface directly, and we must remove the old connections with the controller.

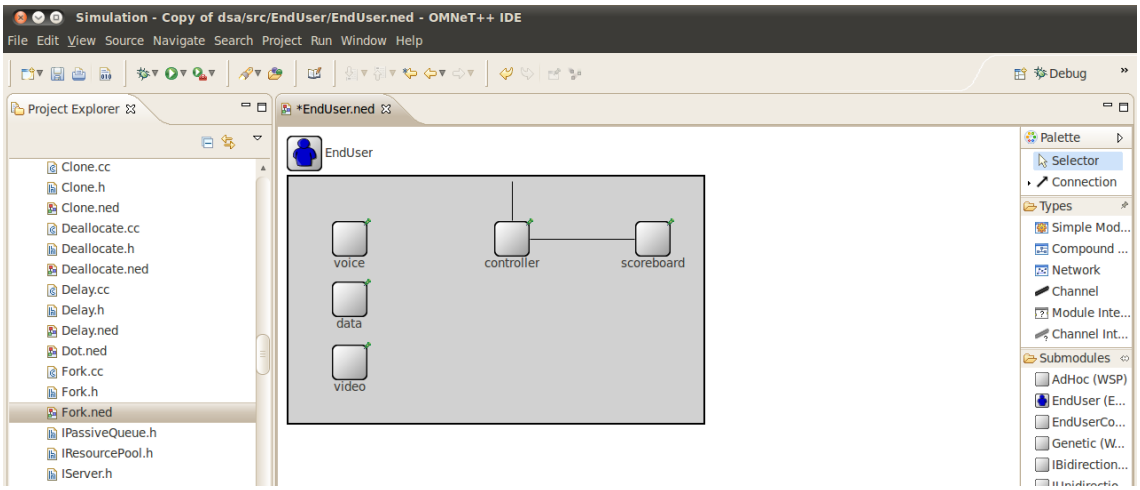


Figure 144: Modified EndUser.ned with 3 SessionGenerators

The next step is to connect the generators with the controller, however as the controller gate is not a vector gate but a scalar one; we cannot connect them directly. To connect them we are going to use two modules defined in the queueinglib library: (1) the Fork module and (2) the Merge module.

The Fork module has a scalar gate as input and a vector gate as output gate. All the messages that arrive through the input gate are replicated and sent forward through all the output gates that are connected. This will be useful to connect the output gate from the controller to the generators module, as all of them must receive the message that indicates the end of the simulation.

The Merge module has a vector gate as input and a scalar one as output gate. All the messages that arrive to the module are forwarded to the output gate. We will use this gate to merge all the messages generated by the generators and send them to the controller.

To be able to use these modules we have to include them into the project. To do so we need to first include them into the EndUser.ned file:

```
import org.omnetpp.queueing.Fork;
import org.omnetpp.queueing.Merge;
```

We also need to modify the configuration file so OMNeT++ knows the folder where it has to look for the ned files:

```
ned-path = .;../../queueinglib
```

Finally, to run the simulation we need to link the library so the object files are found. If we run the simulation from the IDE we can do so from the “Run Configurations” dialog; we have to add as a dynamic library the queueinglib.

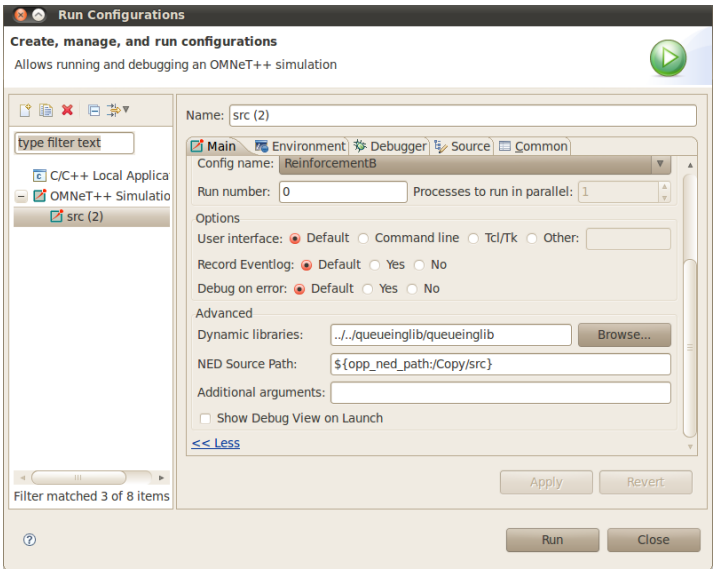


Figure 145: Run configurations dialog with queueinglib as a dynamic library

To run the simulation from the command line we can use the option “-l” to include a library:

```
./Copy -l../../queueinglib/queueinglib
```

We need first to interconnect the modules, though. To connect the modules we need to modify the EndUser.ned file again, this time in the “source” mode, as the “design” one would not allow us to do the connections. To address the output gate of an inout gate we use the suffix $o; similarly, we use $i to address the input gate. The connections are:

```
voice.controller$o --> joiner.in++;
data.controller$o --> joiner.in++;
video.controller$o --> joiner.in++;
joiner.out -->
controller.sessionGenerator$i;
voice.controller$i <-- splitter.out++;
data.controller$i <-- splitter.out++;
video.controller$i <-- splitter.out++;
splitter.in <--
controller.sessionGenerator$o;
```

E.2 Example 2: Modifying the spectrum transaction into an auction

This example shows the necessary changes in the code to modify the transaction between the SpectrumBroker and the WSPs, converting it into an auction. First of all, as the current transaction bid only informs about the desired bandwidth, we need to define a new type of bid that contains the desired bandwidth as well as the price the WSP is willing to pay for a unit of bandwidth. Therefore we create a new message definition with both parameters:

```
message BidMsg {
       //Bandwidth the WSP is willing to obtain
       double BW;
       //Price the WSP is willing to pay for a unit of bandwidth
       double price;
}
```

The next step is to modify the two modules that will use this message: (1) the WSPController and (2) the SpectrumBroker.

To allow the model to switch between the old behaviour and the new one, we will define a ned parameter to select between them. Therefore we include a new parameter in the WSP.ned file:

```
string transaction @enum("priceFunction", "auction")
```

Moreover as the WSP will need to know which value to choose as price, we decide to define it as a new variable of the learning module. This implies defining the range of values it can vary among. To define this range we use the same technique as with the bandwidth, that is, as two ned parameters.

```
//Minimum price the WSP is willing to pay for a unit of spectrum
double minPrice;
//Maximum price the WSP is willing to pay for a unit of spectrum
double maxPrice;
```

The new variable has to be defined in the WSPController initialization phase:

```
if(!strcmp(par("transaction"), "auction")) {
       v = new Variable("bidPrice",par("minPrice"),
par("minPrice"), par("maxPrice"), par("precision"),true);
       learning->addVariable(v);
}
```

Now we have to modify the code concerning to the creation of the message, switching based on the "transaction" parameter.

```
if(!strcmp(par("transaction"), "priceFunction")) {
       double BW = learning->getValue("BW");
       WSPBid* bid = new WSPBid();
       bid->setKind(Bid);
       send(bid, "spectrumBroker$o");
} else {
       double BW = learning->getValue("BW");
       double p = learning->getValue("bidPrice");
       BidMsg* bid = new BidMsg();
       bid->setBW(BW);
       bid->setPrice(p);
       bid->setKind(Bid);
       send(bid, "spectrumBroker$o");
}
```

Finally we have to include the message files, so the WSP will compile:

```
#include "BidMsg_m.h"
```

The changes on the SpectrumBroker module are quite similar. First we include the option to switch between the allocation algorithms; as we already have a parameter to select it, we only need to add the auction as an option:

```
string allocationAlgorithm @enum("fairness", "proportional", "auction");
```

Based on the allocateBandwidthFairness( ), we write a new function, allocateBandwidthAuction( ) , that allocates the bandwidth as a first price auction:

```
bool compareBids(BidMsg* m1, BidMsg* m2) {
	return m1->getPrice() > m2->getPrice();
}

void SpectrumBroker::allocateBandwidthAuction() {
	vector<BidMsg*> aux;
	int num=0;
	for (int i=0; i<numWSP; i++) {
		if (answers[i] != NULL) {
			aux.push_back(dynamic_cast<BidMsg*> (answers[i]));
			num++;
		}
	}
	sort(aux.begin(), aux.end(),compareBids);
	double available=par("bandwidth");
	int i=0;
	while (i<num && aux[i]->getBW()<=available){
		sendAllocation(aux[i]->getBW(),aux[i]->getPrice()*aux[i]-
>getBW(), aux[i]->getArrivalGate()->getIndex());
		available -= aux[i]->getBW();
		totalUtility += aux[i]->getPrice()*aux[i]->getBW();
		i++;
	}
	while (i<num){
		sendAllocation(0,0,aux[i]->getArrivalGate()->getIndex());
	}

}
```

Now we only need to modify the allocateBW() function to select among the 3 allocation functions available:

```
void SpectrumBroker::allocateBandwidth(){
	if(!strcmp(par("allocationAlgorithm"), "proportional")){
		allocateBandwidthProportional();
	} else if (!strcmp(par("allocationAlgorithm"),
"fairness")) {
		allocateBandwidthMaxMinFairness();
	} else {
		allocateBandwidthAuction();
	}
	[…]
}
```

Obviously we need to include the new function declared in the header file:

```
void allocateBandwidthAuction();
```

And finally it only remains to add the new parameter values into the omnetpp.ini file, recompile the source and run it.

Appendix F Cluster Simulation Description

The main goal of a simulation tool is usually to obtain results under a set of environment conditions. Unfortunately, to be able to obtain accurate conclusions under scenarios that include randomness, it is necessary to run numerous times each situation. The high number of simulations that need to be executed involve a high resource consumption, specially CPU consumption.

To address this problem, and consequently accelerate the obtaining of results, we have designed a cluster model, which is pretty simple but at the same time very efficient for the type of simulations that we must run. The basic idea is really simple, it is just distributing the simulations that must be run among several computers, each of them is a node of the cluster.

As the availability of each node is undetermined, some mechanisms to stop and restart the simulations have been implemented. The simulations can be stopped deliberately or due to technical reasons (such as the computer has been switched off or it has run out of disk space). Therefore we need a mechanism to know the progress of the simulations being run at every node. This knowledge will also simplify the planning and distribution for future simulations.

The nodes that make up the cluster have a heterogeneous hardware: different processors (AMD or Intel, one core or multiple cores), different architectures (686 or amd64). All the software used to configure and run the cluster is licensed under GPL terms and can be found in any GNU/Linux distribution, though.

F.1 Node Description

Currently the cluster is composed of eleven nodes, which are described in the following table:

Name	CPU	Score
m2	2xdual-core AMD Opteron 216	3049
t1000	AMD Opteron 256	2153
devel1	Intel Pentium Dual E2180 @2GHz	1128
lorena	Intel Pentium Dual E2180 @2GHz	1128
ch0k0late	AMD Athlon 64 X2 Dual Core 3800+	1045
e4200	Intel Core2 Duo U9400 @1.40GHz	956
t0uchm3	Intel Core2 Duo U9400 @1.40GHz	956
ronya	Intel Pentium D 3.00GHz	811
jorge	Intel Pentium D 3.00GHz	811
node1	Intel Pentium 4 2.80GHz	416
node0	AMD Athlon XP 1700+ 1.8GHz	310

The nodes of the cluster are located in three different networks. As only one of the networks uses public IP addresses; we have implemented a VPN to access the nodes from one of the private networks to the other one.

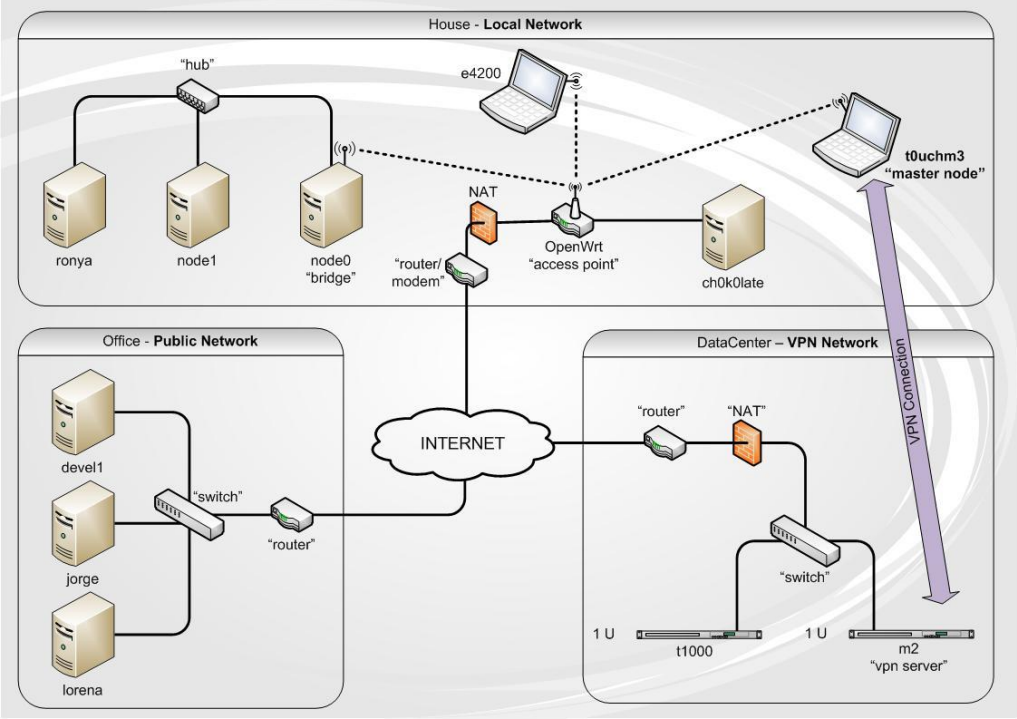


Figure 146: Cluster and networks

F.2 Software requirements

The required software to implement the cluster on each node is:

- OMNeT++ plus the simulation code: basic to be able to run the simulation on each node.
- Octave: computes and pre-analyzes the results obtained.
- Openssh-server: will be used for the communication between the master nodes and the slave nodes.
- Bash interpreter: to be able to run the scripts developed.
- Git: it is not necessary, but it simplifies the code update and download.

F.3 Cluster simulation parts

The cluster functionality is implemented through 2 shell scripts. The first script, runsimulation.sh, runs the simulations on the node it is executed and extracts and pre-analyzes the results obtained after a set of simulations is run. The second one, cluster_management.sh, manages all the simulations executed on each node from the master node.

F.4 runsimulation.sh

runsimulation.sh is the core of the simulation cluster. Its main purpose is to execute the simulations, extract some relevant data from the results and obtain their average for the runs with the same parameters. These average values are stored in a ".means" file, so they can be analysed a posteriori. At the same time it saves the current status, in the satus.sim file, so this process can be stopped at any time and restarted afterwards.

Anytime you want to run runsimulation.sh to execute a new simulation you must provide 3 parameters, (1) the name of the configuration you want to run, (2) the initial run and (3) the final run. If a simulation has been stopped previously you can resume it by invoking the script without any parameters.

```
./runsimulation.sh 0 10100 GeneticAincrement
./runsimulation.sh
```

The .means file generated by the script is named <Config>-<initial_run>-<final_run>.means, it is organised as a matrix, where the columns are the average values of the runs with the same parameters, the rows are different output values that we have considered of interest. For instance, if we have an omnetpp.ini file like:

```
[General]
 […]

[Config GeneticIfading2]
repeat = 1000
**.numUsers = 200
**.numWSP = 5
**.wsp[*].learningModule = "Genetic"
**.populationSize = 10
**.mutationProbability = 0.25
**.crossoverProbability = 0.5
**.increment = 0.25
**.fading = ${0..1 step 0.1}
[…]
```

We can see that the configuration "GeneticIfading" has 11*1000=1000 runs, which will have 11 different possible values for the fading parameter, each of which will be repeated 1000 times. Therefore run 0 to 999 have a fading = 0,for runs from 1000 to 1999 it is 0.1, etc. The GeneticIfading-0-10999.means file will have 11 columns, each with the average of the value for the case where fading is 0, 0.1, 0.2, etc.

```
6.37E+06    6.32E+06    6,40E+14    6,33E+14    …
9.82E+04    2.85E+05    3,27E+13    3,13E+13
9.88E-02    9.85E-02    9,90E+06    9,86E+06
3.90E-03    3.70E-03    4,13E+05    4,11E+05
1.06E-01    1,06E+07    1,06E+07    1,06E+07
1.08E-03    9,88E+04    9,60E+04    1,10E+05
8.71E+04    8.69E+04    8,73E+12    8,63E+12
1.06E+05    1,06E+13    1,06E+13    1,05E+13
6.25E+01    6,25E+09    6,25E+09    6,25E+09
1.06E+05    1,06E+13    1,06E+13    1,05E+13
```

The first row of the .means file corresponds to the average global utility of a WSP, the second one is its deviation. The third and forth column correspond to the average values of the servicePercentage and its deviation. Then there is the average acceptancePercentage and its deviation. Next couple of rows are the average and deviation of the SpectrumBroker utility and finally there is the end-user utility.

F.5 cluster_management.sh

cluster_management.sh is a shell script that allows us to control all the simulations being run in the cluster from the master node, as well as stop and restart them.

It works invoking through ssh the runsimulation.sh script. Other additional functionality that we have implemented on this script is the possibility of checking the current status of the simulation on each node, update their code to the latest version from the git server, download the means file and deleting the results.

It is an interactive script with a menu to select among the available options.

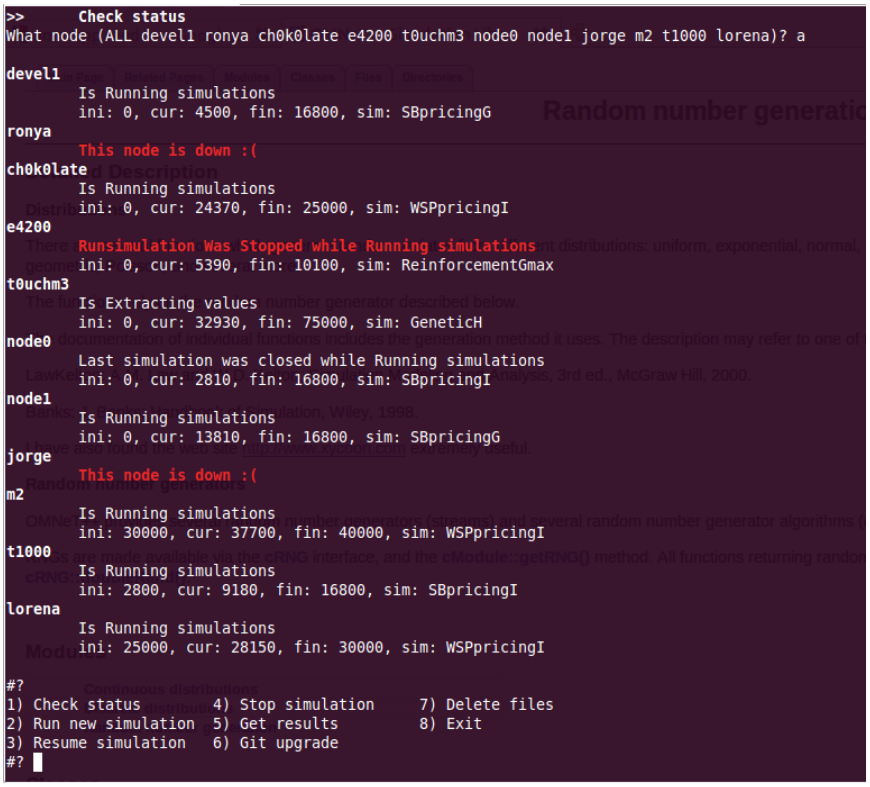


Figure 147: cluster_management.sh execution

F.6 How to install and configure the cluster

In this section the necessary steps to configure the simulation cluster will be explained, so anyone would be able to configure one on their own.

F.7 Common steps for all the nodes

These steps must be run in all the nodes that will make up the cluster.

To avoid troublesome authentication while running the cluster_management.sh script, we have decided to create a common user for all machines, all the steps are run logged as this user.

```
adduser omnetpp #or whatever other name you wish
```

First of all install OMNeT++, the installation instructions can be found on their web page, or type the following in a command window:

```
wget http://www.omnetpp.org/omnetpp/doc_download/2198-omnet-40p1-source--ide-tgz #obtain the source code from the web page, you may want to download the lattest version
tar xvfz omnetpp-4.0p1-src.tgz #decompress the file downloaded
apt-get install build-essential gcc g++ bison flex perl tcl8.4 tcl8.4-dev tk8.4 tk8.4-dev blt blt-dev libxml2 libxml2-dev zlib1g zlib1g-dev libx11-dev
cd omnetpp-4.0p1 #install the omnetpp deppendecies.
./configure
make
make install
echo "export PATH=$PATH:/home/omnetpp/omnetpp-4.0p1/bin" >> ~/.bashrc #export the folder where omnetpp is installed, make sure this path is the one where you have installed omnetpp (if your username is different it may change)
```

Now you should be able to run the OMNeT++ IDE by typing (You may need to restart the X server):

```
omnetpp
```

The next step is to obtain and compile the code of the simulation model. But before that, we will install all the cluster_management.sh dependencies:

```
apt-get install openssh-server screen git-core octave3.2
```

Now that we have git installed, we can use git to obtain the source code. Locate yourself in the samples folder of the omnetpp (or wherever you want to install the model simulation) and execute:

```
git clone git@ping.indefero.net:ping/dsa.git
```

Now you have the code in your computer, to compile it:

```
cd dsa/src
opp_makemake -f --deep #generates the makefile
make
```

At this point there should be an executable file to run the simulation:

```
./dsa
```

The scripts are downloaded simultaneously with the simulation model code. They are on the scripts folder, you can see them by:

```
cd scripts
ls
```

F.8 Master node steps

The following steps are the steps that must be run in the master node.

First of all we will make the nodes accessible by name, to save time whenever we need to access them. To do so we must map them in the /etc/hosts file:

```
127.0.0.1 localhost
127.0.1.1 t0uchm3
10.0.0.10 ronya
10.0.0.11 e4200
10.0.0.12 t0uchm3
10.0.0.5 ch0k0late
10.0.0.150 node0
10.0.0.151 node1
XXX.XXX.XXX.XXX  devel1
XXX.XXX.XXX.XXX jorge
192.168.0.101 m2
192.168.0.102   t1000
XXX.XXX.XXX.XXX lorena
```

Then we have to configure the paths of the cluster_management.sh, to do that edit your cluster_management.sh file so that their first lines are correct.

```
PATH_SCRIPTS="/home/omnetpp/omnetpp-4.0p1/samples/dsa/src/scripts"
PATH_RESULTS="/home/omnetpp/omnetpp-4.0p1/samples/dsa/src/results"
PATH_SRC="/home/omnetpp/omnetpp-4.0p1/samples/dsa/src"
PATH_MEANS="/home/omnetpp/Ubuntu One/PFC/results/means"
PATH_ANOVA="/home/omnetpp/Ubuntu One/PFC/results/anova"
PATH_TAR="/mnt/Ester/PFC tars"
nodes="devel1 ronya ch0k0late e4200 t0uchm3 node0 node1 jorge m2 t1000
lorena"
```

The first one should point to the scripts folder, the second one to the folder where the results are stored (by default will be the results folder), then the source folder and finally the folders where you want to store the means file, the anova file and the compressed file when you retrieve them from the others nodes. The last line is the set of nodes that make up the cluster, just as you named them in the /etc/hosts file. Similarly, the file extractVarValues.sh has to be configured.

Also, to avoid user authentication you must create your own ssh key:

```
ssh-keygen
```

This key will be added to the list of authorized keys on the slave nodes afterwards.

F.9 Slave nodes

The only thing left is to add in the slaves nodes the public key of the master node, to do so you can type:

```
ssh omnettp@IP_MASTER_NODE -C "cat ~/.ssh/id_rsa.pub" >> ~/.ssh/authorized_keys
```

Or simply paste the contents of id_rsa.pub file from the master node, into the authorized_keys file of the slave node.